HiLA: Historical-Layer Attention for Low-Rank Adaptation

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient fine-tuning (PEFT) has become an important paradigm for adapting large language models (LLMs) to downstream tasks. However, existing methods such as LoRA typically perform local low-rank updates conditioned only on the current-layer input, leaving task-relevant representations from preceding layers underutilized. Our analysis shows that shallow and intermediate layers already contain information that can influence final predictions, motivating the explicit use of cross-layer representations for low-rank adaptation. To this end, we propose HiLA (Historical Layer Attention), a PEFT method that dynamically integrates cross-layer representational changes into low-rank updates. HiLA constructs differences between the current and preceding-layer states and uses the current LoRA bottleneck representation as a query to retrieve relevant historical information through attention. We evaluate HiLA on mathematical reasoning, commonsense reasoning, and open-domain dialogue generation across four backbones with different architectures and scales. HiLA achieves the highest average commonsense accuracy on all four backbones and the best performance on five of six dialogue-generation metrics. Notably, on challenging mathematical reasoning tasks, HiLA consistently outperforms all low-rank baselines and even surpasses full fine-tuning on Llama-3.2-3B with only 1.03% trainable parameters. These results indicate that attention-based cross-layer representations effectively alleviate the limitations of existing PEFT methods. Our code is available at `https://anonymous.4open.science/r/HiLA-8209/`

1 Introduction

With the large language model (LLM) backbone parameters frozen, parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019) enables efficient adaptation to downstream tasks by updating only a small number of parameters. As a representative PEFT method, LoRA (Hu et al., 2022) introduces low-rank matrices to construct the weight update $\Delta\mathbf{W}$ and relies solely on the current-layer input representation $\mathbf{x}_\ell$ to compute the adapter update $\Delta\mathbf{W}_\ell\mathbf{x}_\ell$. However, this design isolates adapters across layers, preventing them from directly exploiting rich cross-layer information. Therefore, enabling low-rank branches to extract and integrate cross-layer information is a key step toward further extending the capability of LoRA.

Recent studies (Zhou et al., 2025; He et al., 2025b) show that introducing cross-layer connections within low-rank adapters can reduce inter-layer redundancy and improve collaboration among adapters at different depths. Meanwhile, the analysis in Layer by Layer (Skean et al., 2025) shows that shallow and middle layers already contain informative representations that can be directly used for task discrimination. Together, these findings indicate that preceding-layer representations provide useful information for improving final predictions. However, existing cross-layer methods primarily focus on parameter sharing or parameter reuse, while the explicit use of representations from preceding layers remains limited.

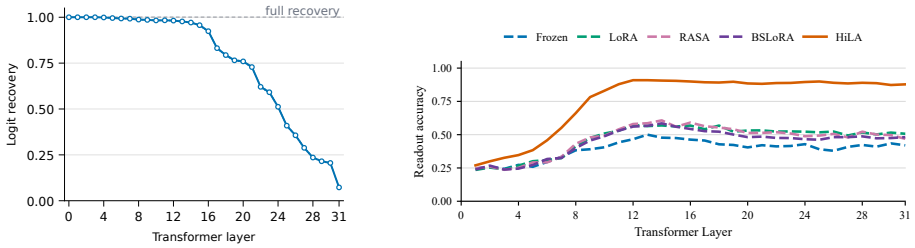
To substantiate the importance of preceding layers for the model's final predictions, we design an Activation Patching experiment inspired by Zhang and Nanda (Zhang & Nanda, 2024). Activation Patching introduces counterfactual interventions to test the causal link between intermediate representations and the final model output. Specifically, we construct a clean input that leads the model to the correct answer and a corrupted input that induces an incorrect prediction. During the forward

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.

pass on the corrupted input, for a specific Transformer layer, we replace only the hidden state of the critical token with the corresponding activation from the clean input at the same layer and position, while leaving all other token states and the subsequent computation path unchanged. The rationale is that if replacing only a local intermediate state substantially restores the logit margin for the correct answer, then the patched layer contains task-relevant information that can propagate through subsequent computation and influence the final prediction. As shown in Figure 1a, patching shallow and middle layers yields substantially greater prediction recovery than patching deeper layers. This result provides direct evidence that preceding layers play an important role in both the formation of task-relevant information and its contribution to subsequent predictions.

We further examine whether existing methods effectively retain and exploit these preceding-layer features. Following the paradigm of layerwise readout of intermediate representations (Alain & Bengio, 2016), we train a Linear Probe on the representation of the last valid token at each layer for each method. Linear-probe accuracy directly measures how much task-relevant information in a given layer can be extracted by a linear classifier. Under effective cross-layer information flow, if the model successfully extracts and preserves key information from preceding layers, the readoutability of such task information should be maintained or even enhanced in deeper layers. However, the probing results in Figure 1b reveal a clear disconnect. Although preceding layers already contain highly readable task-relevant information, LoRA and existing cross-layer variants fail to achieve comparable improvements in readoutability.

Taken together, the probing results and the Activation Patching results in Figure 1a support a clear inference: preceding layers already contain key task-relevant features that can influence the final prediction, yet the local update mechanisms of existing PEFT methods do not effectively exploit these historical features. As a result, critical information from preceding layers is not fully converted into readable task information as it propagates to deeper layers. This observation motivates our central research question: how can a low-rank adapter explicitly extract and integrate historical representation-evolution information according to the adaptation needs of the current layer, thereby overcoming the capability bottleneck of existing PEFT methods?



(a) Frozen-backbone activation patching. (b) Linear-probe readout accuracy.

Figure 1: Results of the activation-patching and layerwise linear-probe experiments.

Building on this insight, we propose HiLA, an attention-based method for utilizing cross-layer hidden states. HiLA incorporates the cross-layer state differences between preceding-layer outputs and the current-layer input state into the low-dimensional bottleneck representation of the LoRA branch, thereby improving its representational capacity. Specifically, HiLA maps the Historical Cross-Layer State Differences to keys and values, maps the current LoRA bottleneck representation to a query, and uses Historical-Layer Attention to compute the cross-layer representation most relevant to the current adapter. This design establishes a direct path for the current adapter to read changes in historical representations and substantially improves the layerwise readoutability of task-relevant information, as shown in Figure 1b.

In summary, our main contributions are threefold.

1 We combine Activation Patching with layerwise Linear Probing to show that hidden states in shallow and middle layers already contain task-relevant information that can influence final predictions, while existing PEFT methods do not fully exploit this information.

2 We propose HiLA, a Historical-Layer Attention adaptation method that explicitly uses Historical Cross-Layer State Differences through attention to enhance low-rank updates.

HiLA substantially improves the layerwise readoutability of task-relevant information and strengthens the model's representation capacity.

3 We systematically evaluate HiLA on mathematical reasoning, commonsense reasoning, and open-domain dialogue generation. With a comparable number of trainable parameters, HiLA consistently outperforms LoRA and other representative baselines across multiple models and task settings. On mathematical reasoning, HiLA even surpasses FFT on GSM8K for both backbones and on MATH for Llama-3.2-3B.

2 Related Work

LoRA and Its Low-Rank Adaptation Variants. LoRA (Low-Rank Adaptation) (Hu et al., 2022) enables efficient adaptation to downstream tasks by introducing a low-rank update branch alongside frozen pretrained weights. Specifically, for the layer ℓ, LoRA parameterizes the weight update using two low-rank matrices, $\mathbf{A}_\ell \in \mathbb{R}^{r\times d_{\text{in}}}$ and $\mathbf{B}_\ell \in \mathbb{R}^{d_{\text{out}}\times r}$, and computes the low-rank update as

$$\Delta\mathbf{W}_\ell\mathbf{x}_\ell = \frac{\alpha}{r}\mathbf{B}_\ell\mathbf{A}_\ell\mathbf{x}_\ell = \frac{\alpha}{r}\mathbf{B}_\ell\mathbf{z}_\ell, \qquad \mathbf{z}_\ell = \mathbf{A}_\ell\mathbf{x}_\ell. \tag{1}$$

Here, α is the scaling factor, and $\mathbf{z}_\ell \in \mathbb{R}^r$ denotes the low-dimensional bottleneck representation of the current LoRA branch. To further improve the capacity of LoRA, subsequent studies explore several directions, including initialization strategies, optimization spaces, and effective update rank. PiSSA (Meng et al., 2024), for example, initializes low-rank adapters using the principal singular values and corresponding singular vectors of pretrained weights, allowing low-rank updates to start from a more informative parameter subspace. DoRA (Liu et al., 2024) decomposes pretrained weights into magnitude and direction and uses the low-rank branch specifically to learn directional changes, thereby improving the dynamics of gradient updates. MoRA (Jiang et al., 2024) introduces invertible mappings and parameter sharing to increase the effective update rank of the low-rank branch without increasing the overall parameter budget. Despite these advances, these methods primarily improve low-rank adaptation within the framework of layer-local weight updates and do not explicitly exploit intermediate representations produced by preceding layers.

Cross-Layer Parameter Sharing and Collaboration. To reduce the isolation among adapters at different depths, another line of work introduces cross-layer dependencies to improve parameter efficiency. The earlier COMPACTER (Mahabadi et al., 2021) constructs compact adapters using Kronecker products and balances flexibility with parameter cost by sharing slow weights across layers while retaining layer-specific fast rank-one matrices. TRAC (Ye et al., 2026) subsequently introduces Tensor-Train decomposition and models cross-layer structural correlations by freezing or sharing selected Tensor-Train cores across layers. More recently, cross-layer collaboration is integrated directly into the LoRA architecture. RaSA (He et al., 2025b) selects rank components from layerwise LoRA updates to construct a cross-layer shared rank pool and dynamically recomposes them using layer-specific weights. BSLoRA (Zhou et al., 2025) further decomposes the LoRA weight update into local parameters, intra-layer shared parameters, and inter-layer shared parameters to strengthen coupling among adapters at different depths. Although these methods establish connections across network layers of different depths, their cross-layer interactions remain primarily confined to the parameter space through weight sharing or feature-component reuse. They do not directly operate in the representation space and therefore lack an explicit cross-layer information pathway for reading, retrieving, and propagating the evolving representations produced by preceding layers. This limitation is precisely the central problem that HiLA aims to address.

3 Method

The HiLA workflow consists of three main steps: constructing Historical Cross-Layer State Differences, applying Historical-Layer Attention, and updating the History-Conditioned Bottleneck Representation. The overall architecture is illustrated in Figure 2.

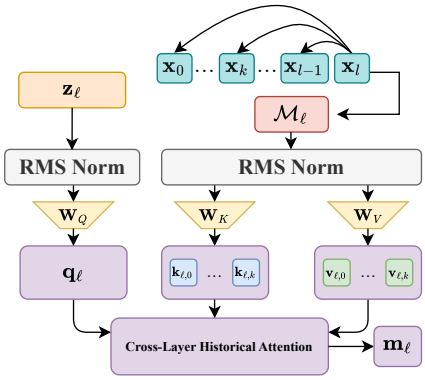


Figure 2: Overall architecture of HiLA.

Historical Cross-Layer State Differences. As shown by the mechanism analysis in Section 4.3, hidden states at different depths share substantial structural information. To emphasize the components of representations that evolve with network depth, HiLA adopts a relative formulation that uses the Current-Layer Input State as the reference to construct Cross-Layer State Differences while suppressing shared structural information.

Let $\mathbf{x}_\ell \in \mathbb{R}^d$ denote the input state of the ℓ-th Transformer block, where $\mathbf{x}_0$ is the input state of the first block and $\mathbf{x}_k$ is the input state of the k-th block. Using the current-layer input state $\mathbf{x}_\ell$ as the reference, the difference between $\mathbf{x}_\ell$ and a historical state $\mathbf{x}_k$ is defined as

$$\boldsymbol{\delta}_{\ell,k} = \mathbf{x}_\ell - \mathbf{x}_k, \qquad 0 \le k < \ell. \tag{2}$$

The difference vector $\boldsymbol{\delta}_{\ell,k}$ explicitly captures the representation change of the same token as it propagates from the k-th block to the current ℓ-th block. Accordingly, the set of historical state differences available to the current layer is defined as

$$\mathcal{M}_\ell = \{\boldsymbol{\delta}_{\ell,0}, \boldsymbol{\delta}_{\ell,1}, \ldots, \boldsymbol{\delta}_{\ell,\ell-1}\}. \tag{3}$$

To prevent scale differences among state differences at different depths from affecting the subsequent attention computation, HiLA applies non-affine RMSNorm (Zhang & Sennrich, 2019) separately to the query input $\mathbf{z}_\ell$ and each historical state difference $\boldsymbol{\delta}_{\ell,k}$.

Historical-Layer Attention Selection. Because adaptation requirements vary across samples and network depths, a static historical information fusion mechanism lacks sufficient flexibility. HiLA therefore uses the Low-Dimensional Bottleneck Representation of the current LoRA branch as the query and performs dynamic attention-based retrieval over the depth dimension of the set of Historical Cross-Layer State Differences.

Let the query-key matching dimension of the attention mechanism be r_h, the LoRA rank be r, and the hidden dimension be d. We introduce projection matrices $\mathbf{W}_Q \in \mathbb{R}^{r_h \times r}$, $\mathbf{W}_K \in \mathbb{R}^{r_h \times d}$, and $\mathbf{W}_V \in \mathbb{R}^{r \times d}$ to compute the query, key, and value vectors, respectively:

$$\mathbf{q}_\ell = \mathbf{W}_Q \mathbf{z}_\ell, \qquad \mathbf{k}_{\ell,k} = \mathbf{W}_K \boldsymbol{\delta}_{\ell,k}, \qquad \mathbf{v}_{\ell,k} = \mathbf{W}_V \boldsymbol{\delta}_{\ell,k}. \tag{4}$$

For any $\ell \ge 1$, HiLA stacks the key and value vectors according to the layer index $k = 0, \ldots, \ell-1$ into matrix form:

$$\mathbf{K}_\ell = \begin{bmatrix} \mathbf{k}_{\ell,0}^\top \\ \vdots \\ \mathbf{k}_{\ell,\ell-1}^\top \end{bmatrix} \in \mathbb{R}^{\ell \times r_h}, \qquad \mathbf{V}_\ell = \begin{bmatrix} \mathbf{v}_{\ell,0}^\top \\ \vdots \\ \mathbf{v}_{\ell,\ell-1}^\top \end{bmatrix} \in \mathbb{R}^{\ell \times r}.$$

Each row of $\mathbf{K}_\ell$ and $\mathbf{V}_\ell$ corresponds to the key and value of one historical state difference, respectively. HiLA then applies standard single-query scaled dot-product attention to measure the match between the current low-dimensional query and each historical state difference. The softmax function produces normalized attention weights, which are used to compute a weighted aggregation of

the value representations:

$$\mathbf{a}_\ell = \mathrm{softmax}\left(\frac{\mathbf{K}_\ell \mathbf{q}_\ell}{\sqrt{r_h}}\right) \in \mathbb{R}^{\ell \times 1}, \qquad \mathbf{m}_\ell = \mathbf{V}_\ell^\top \mathbf{a}_\ell \in \mathbb{R}^{r \times 1}. \tag{5}$$

The k-th element of $\mathbf{a}_\ell$ represents the attention weight assigned to the difference $\boldsymbol{\delta}_{\ell,k}$, while $\mathbf{m}_\ell$ denotes the aggregated historical-attention output vector.

History-Conditioned Low-Rank Update. A key advantage of HiLA is that it enhances representational capacity without altering the original architecture of LoRA. As shown in Figure 3, HiLA explicitly injects the historical-attention output $\mathbf{m}_\ell$ into the Low-Dimensional Bottleneck Representation before the LoRA up-projection (write-back) matrix $\mathbf{B}_\ell$:

$$\widetilde{\mathbf{z}}_\ell = \mathbf{z}_\ell + \mathbf{m}_\ell, \qquad \Delta \mathbf{W}_\ell \mathbf{x}_\ell = \frac{\alpha}{r} \mathbf{B}_\ell \widetilde{\mathbf{z}}_\ell. \tag{6}$$

With this design, the low-dimensional bottleneck representation of the adapter is extended from a local representation that depends only on the current-layer input to a conditioned representation that incorporates representation evolution from preceding layers. In particular, at the first network layer, where no historical states are available and $\mathbf{m}_0 = \mathbf{0}$, or when historical information is disabled, $\widetilde{\mathbf{z}}_\ell = \mathbf{z}_\ell$. In either case, the update in equation 6 reduces exactly to the standard LoRA update.

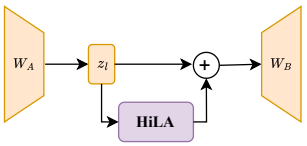


Figure 3: Insertion point of HiLA in the LoRA branch.

To limit the parameter and computational overhead introduced by historical retrieval, HiLA reuses hidden states from the forward pass and shares the projection matrices across layers and compatible modules. For each sharing group with hidden dimension d, the resulting additional parameter overhead is $P_{\text{hist}} = r_h r + r_h d + r d$, corresponding to $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$, respectively; this count is computed per shared projection group rather than per layer. The total trainable parameters remain approximately 1% of the frozen backbone parameters in our configuration. When retrieving from k preceding layers with $r_h \ll r$, the computational complexity is $\mathcal{O}(dkr)$. We further employ FlashAttention (Dao et al., 2022) to accelerate the attention computation over the projected historical representations.

4 Experiments

4.1 Experimental Setup

Datasets. We comprehensively evaluate HiLA on three representative tasks. The data sources, dataset sizes, and detailed configurations are provided in Appendix A.4.

- **Mathematical Reasoning:** We perform instruction tuning on MetaMathQA (Yu et al., 2023), which contains a large collection of high-quality examples with multi-step reasoning. For evaluation, we use the official test sets of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). GSM8K primarily evaluates the ability to solve math word problems, whereas MATH assesses advanced reasoning on more challenging mathematical problems.
- **Commonsense Reasoning:** We train the models on Commonsense-170K (zwhe99, 2024). To evaluate commonsense reasoning across multiple dimensions, including true-or-false judgment, physical reasoning, social reasoning, and scientific reasoning, we use the OpenCompass framework (Contributors, 2023) and conduct evaluations on eight widely used benchmarks: BoolQ, PIQA, SocialIQA, HellaSwag, WinoGrande, ARC-Challenge, ARC-Easy, and OpenBookQA.
- **Open-Domain Dialogue Generation:** We use ConvAI2 (Dinan et al., 2019) to evaluate multi-turn dialogue generation. Following the Self-Persona setting of Huang et al. (Huang et al., 2024), we retain only the profile of the current speaker in the prompt to better simulate personalized dialogue scenarios. The detailed evaluation protocol is provided in Appendix A.5.

Evaluation Metrics. For mathematical and commonsense reasoning, we use exact-match accuracy as the primary evaluation metric and additionally report the arithmetic mean accuracy across the eight commonsense reasoning benchmarks. For open-domain dialogue generation, we evaluate generation quality using multiple text generation metrics, including BLEU (Papineni et al., 2002), BERTScore-F1 (Zhang et al., 2020), METEOR (Banerjee & Lavie, 2005), and ROUGE-L (Lin, 2004). The prompt templates and answer extraction rules are provided in Appendix A.5.

Baselines and Backbone Selection. To evaluate the generalization ability of the proposed method, we select four pretrained models with different architectures and scales as frozen backbones: Qwen2.5-1.5B (Qwen Team, 2024), Qwen3-8B (Yang et al., 2025), Llama-3.2-3B (Meta AI, 2024), and Llama-3.1-8B (Llama Team, 2024). We compare HiLA with prompt-based methods, including Prompt Tuning and P-Tuning, as well as low-rank adaptation methods, including LoRA, DoRA (Liu et al., 2024), MoRA (Jiang et al., 2024), RaSA (He et al., 2025b), and PiSSA (Meng et al., 2024).

Training Protocol. All experiments are conducted on NVIDIA A100 GPUs. Mathematical and commonsense reasoning tasks use an Alpaca instruction template (Taori et al., 2023), while ConvAI2 directly takes structured dialogue contexts as input. We use a fixed configuration in all main experiments: the rank of all low-rank methods is $r = 32$, HiLA uses historical attention rank $r_h = 4$, and its historical readout projections are shared across modules with matching dimensions. Historical readout is applied to W_q, W_k, W_v, and W_{up}, while W_{down} retains the standard LoRA branch; all low-rank baselines adapt the same five target modules. The unified hyperparameters and implementation configurations are detailed in Appendix A.2–A.3.

4.2 Main Results and Analysis

Mathematical Reasoning Results. Table 1 reports the results on GSM8K and MATH. HiLA demonstrates a significant advantage over all reported competing methods across both backbone models, consistently outperforming LoRA, DoRA, PiSSA, and RaSA. Notably, on the Llama3.2-3B model, HiLA achieves accuracies of 68.84% and 17.82% on GSM8K and MATH, respectively, with only approximately 1.03% trainable parameters, surpassing the performance of full fine-tuning (FFT). On the Llama3.1-8B model, HiLA obtains accuracies of 79.15% and 33.24% on GSM8K and MATH, respectively. Specifically, it improves over FFT by 5.76% on GSM8K and surpasses the best low-rank baseline PiSSA by 3.12% on MATH. Particularly on GSM8K, HiLA achieves a substantial improvement of 10.01% over the strongest low-rank baseline DoRA, further demonstrating its superiority in complex mathematical reasoning tasks. These results indicate that the cross-layer historical information utilization mechanism introduced by HiLA provides significant advantages in handling challenging multi-step mathematical reasoning problems.

Table 1: Mathematical reasoning results on GSM8K and MATH (%). Bold and underlined values denote the best and second-best results, respectively. † marks the value reported by He et al. (He et al., 2025b).

Model	Method	Trainable (%)	GSM8K	MATH
Llama-3.2-3B	FFT	–	<u>65.13</u>	<u>16.84</u>
	LoRA	1.0281	50.95	10.78
	DoRA	1.0424	51.63	11.64
	PiSSA	1.0281	52.69	12.14
	RaSA	1.0287	53.15	12.56
	HiLA	1.0315	**68.84**	**17.82**
Llama-3.1-8B	FFT	–	<u>73.39</u>	**35.50**†
	LoRA	0.7051	67.70	28.94
	DoRA	0.7149	69.14	29.06
	PiSSA	0.7051	68.76	30.12
	RaSA	0.7054	68.69	29.80
	HiLA	0.7070	**79.15**	<u>33.24</u>

Commonsense Reasoning Results. As shown in Table 2, HiLA achieves the highest average accuracy on all four frozen backbones with different model scales and architectures. Specifically, on Qwen2.5-1.5B, HiLA reaches an average accuracy of 79.61%, while on Qwen3-8B, it achieves 88.17%, outperforming the strongest baseline, RaSA, by 0.26%. HiLA also achieves the high-

est average accuracy on both Llama backbones. Overall, these results further show that dynamically retrieving historical representations enables HiLA to provide richer and more precise context-dependent adaptation signals than methods that either rely only on the Current-Layer Input State or establish cross-layer connections in parameter space.

Table 2: Commonsense reasoning results for four frozen backbones (%). Bold and underlined values denote the best and second-best results. HS denotes HellaSwag, and WG denotes WinoGrande.

Model	Method	Trainable (%)	BoolQ	PIQA	SIQA	HS	WG	ARC-c	ARC-e	OBQA	Avg.
Qwen2.5-1.5B	LoRA	1.6047	67.43	82.48	74.87	85.32	72.85	75.00	88.55	82.96	78.68
	DoRA	1.6275	67.37	82.97	75.13	**86.15**	73.88	75.68	88.76	83.76	79.21
	MoRA	1.6046	66.48	82.75	75.44	83.91	**74.27**	73.29	87.46	81.68	78.16
	RaSA	1.6060	67.03	82.70	73.29	83.41	72.38	75.94	88.89	82.88	78.32
	HiLA	1.6083	**68.35**	**83.08**	**75.49**	85.94	74.19	**76.54**	**89.23**	**84.04**	**79.61**
Qwen3-8B	LoRA	0.7201	73.49	89.28	80.24	93.43	85.56	87.88	96.08	89.72	86.96
	DoRA	0.7300	73.33	89.39	80.50	94.38	**85.64**	88.82	95.75	91.16	87.37
	MoRA	0.7195	**73.98**	**90.32**	80.91	95.09	84.85	90.10	96.42	91.04	87.84
	RaSA	0.7205	73.94	89.93	81.22	95.07	83.90	90.78	96.97	91.44	87.91
	HiLA	0.7219	73.15	90.04	**81.83**	**95.22**	85.16	**91.38**	**97.10**	**91.48**	**88.17**
Llama-3.2-3B	LoRA	1.0281	70.09	85.04	78.92	90.66	82.00	74.74	87.12	82.88	81.43
	DoRA	1.0424	70.70	86.02	80.14	90.85	83.50	75.51	87.92	83.64	82.29
	MoRA	1.0262	**72.08**	85.80	79.58	**92.22**	84.21	76.28	88.51	83.40	82.76
	RaSA	1.0287	69.76	84.71	79.68	90.84	81.29	72.87	85.65	81.04	80.73
	HiLA	1.0315	70.55	**86.51**	**80.81**	90.80	**85.00**	**76.96**	**89.02**	**84.12**	**82.97**
Llama-3.1-8B	LoRA	0.7051	73.39	88.19	79.48	**95.13**	83.50	79.52	90.24	83.76	84.15
	DoRA	0.7149	74.95	89.06	77.74	94.13	86.82	80.72	91.04	**87.52**	85.25
	MoRA	0.7046	73.58	87.49	79.17	94.30	85.71	75.94	88.85	84.04	83.64
	RaSA	0.7054	74.43	89.01	**80.81**	94.23	86.11	**81.48**	90.78	86.40	85.41
	HiLA	0.7070	**75.84**	**89.99**	79.94	94.92	**86.90**	81.14	**91.37**	86.24	**85.79**

Open-Domain Dialogue Generation Results. Table 3 reports the generation quality on ConvAI2. Across the six evaluation metrics on Llama-3.2-3B, HiLA achieves the best performance on five. Although it trails DoRA by 0.11% on METEOR, it still outperforms LoRA and MoRA. Overall, these results indicate that Historical-Layer Attention improves semantic consistency while maintaining strong lexical and sequence-level matching, demonstrating the effectiveness of cross-layer historical representations for open-domain dialogue generation.

Table 3: Open-domain dialogue results for Llama-3.2-3B on ConvAI2.

Method	Trainable (%)	BLEU	BERT F1	BERT-R	BERT-P	METEOR	ROUGE-L
Prompt Tuning	0.0019	1.54	82.76	83.96	81.65	7.52	9.94
P-Tuning	0.8834	12.33	86.96	86.52	87.47	18.04	22.85
LoRA	1.0281	15.87	87.43	87.16	87.75	21.12	26.06
DoRA	1.0424	16.03	87.46	87.20	87.78	**21.29**	26.22
MoRA	1.0262	15.00	87.28	87.07	87.56	20.53	25.46
HiLA	1.0315	**16.25**	**87.48**	**87.21**	**87.82**	21.18	**26.33**

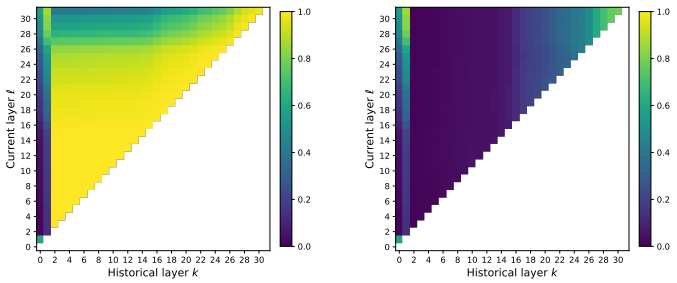
4.3 Mechanistic Analysis

Analysis of Cross-Layer Shared Components. Prior work (Kornblith et al., 2019; Phang et al., 2021) shows that Linear Centered Kernel Alignment (Linear CKA) provides an effective measure of shared representational structure across neural network layers. To investigate whether hidden states at different depths contain highly overlapping redundant information, we use Linear CKA to measure the relational structural similarity between layer representations. Let $\mathbf{H}_k, \mathbf{H}_\ell \in \mathbb{R}^{N\times d}$ denote the centered representations of a historical layer and the current layer, respectively, where $k < \ell$. Linear CKA is defined as:

$$\mathrm{CKA}(\mathbf{H}_k, \mathbf{H}_\ell) = \frac{\left\|\mathbf{H}_k^\top \mathbf{H}_\ell\right\|_F^2}{\left\|\mathbf{H}_k^\top \mathbf{H}_k\right\|_F \left\|\mathbf{H}_\ell^\top \mathbf{H}_\ell\right\|_F}, \tag{7}$$

where higher values indicate stronger similarity in the shared representation structure of the two layers.

To examine whether cross-layer differencing suppresses shared components, we compute Linear CKA for both the layer representations and the cross-layer difference representations over all



(a) $\mathrm{CKA}(\mathbf{H}_k, \mathbf{H}_\ell)$. (b) $\mathrm{CKA}(\mathbf{D}_{k,\ell}, \mathbf{H}_k)$.

Figure 4: Cross-layer structural similarity on Llama-3.1-8B. The left and right panels show $\mathrm{CKA}(\mathbf{H}_k, \mathbf{H}_\ell)$ and $\mathrm{CKA}(\mathbf{D}_{k,\ell}, \mathbf{H}_k)$, respectively.

historical-current layer pairs with $k < \ell$. Figure 4 shows that darker colors indicate lower representational similarity. The mean CKA between the current and historical layers is 0.809, whereas the mean CKA between their difference $\mathbf{D}_{k,\ell} = \mathbf{H}_\ell - \mathbf{H}_k$ and the corresponding historical layer is only 0.126. This substantial decrease indicates that cross-layer differencing effectively suppresses shared components.

Analysis of the Cross-Layer Difference Mechanism. To determine whether cross-layer differences contain critical task-relevant information, we reconstruct the shared component as a reference. Specifically, we introduce a ridge-regression-based cross-layer linear mapping to quantify the linearly predictable component shared between the current and historical layers:

$$\mathbf{W}^*_{k\to\ell} = \arg\min_{\mathbf{W}} \left\|\mathbf{H}_\ell - \mathbf{H}_k\mathbf{W}\right\|_F^2 + \lambda\left\|\mathbf{W}\right\|_F^2, \qquad \widehat{\mathbf{H}}_\ell = \mathbf{H}_k\mathbf{W}^*_{k\to\ell}. \tag{8}$$

To determine whether the evolving layer differences encapsulate crucial task-relevant signals rather than mere computational noise, we conduct a Linear Probing experiment on the HellaSwag dataset. We evaluate the test-set readout accuracy of three distinct representations: the full current state $\mathbf{H}_\ell$, the reconstructed shared component $\widehat{\mathbf{H}}_\ell$, and the cross-layer difference $\mathbf{H}_\ell - \mathbf{H}_k$. Higher probing accuracy strictly indicates a richer presence of task-discriminative information. As shown in Figure 5, the cross-layer difference yields substantially higher probing accuracy than the shared component in the middle layers, frequently approaching or even exceeding the performance of the full current state. Combined with the redundancy analysis in Figure 4, these results reveal a critical insight: while representations across depths share substantial predictable redundancies, their differences effectively distill the core task-discriminative signals. Consequently, instead of directly reusing raw historical states fraught with redundant components, HiLA leverages cross-layer differences to filter out this redundancy and precisely capture the task-relevant features dynamically evolving across depths.

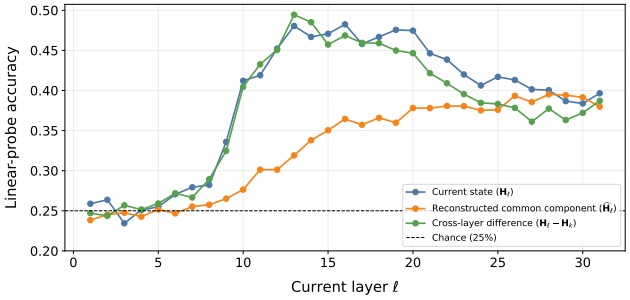


Figure 5: Linear-probe accuracy of the current state, reconstructed shared component, and same-token cross-layer difference. The dashed horizontal line denotes chance-level accuracy (25%) for the four-way classification task.

Validation on Historical Information. The performance gains of HiLA may arise from two possible mechanisms: the additional historical branch may simply increase the adaptation capacity, or the model may genuinely rely on historical representations aligned with the current sample.

To distinguish these explanations, we conduct a counterfactual intervention experiment. To test whether the gains depend on sample-aligned historical information rather than merely on added adaptation capacity, we disrupt the correspondence between samples and their historical states.

However, as shown in Table 4, removing historical information (Zero History) reduces GSM8K accuracy from 68.84% to 59.14%, while replacing it with mismatched historical information (Shuffled History) further reduces accuracy to 41.93%. These results strongly confirm that the performance gains of HiLA depend on sample-aligned historical representation-evolution information rather than merely on the additional adaptation capacity introduced by the historical branch.

Table 4: Counterfactual interventions on sample-aligned historical representations for the GSM8K task.

Intervention	**Acc.**	**Δ Acc.**
HiLA	68.84	–
Zero history	59.14	−9.70
Shuffled history	41.93	−26.91

4.4 Ablation Experiments

We ablate the historical information extraction and selection modules on the Qwen2.5-1.5B backbone to examine HiLA's core design. For feature aggregation, we compare mean aggregation (mean) with query-conditioned attention (attn). For feature construction, we consider the raw historical state (s_k), the adjacent-layer difference ($s_{k+1} - s_k$), and the Cross-Layer State Difference ($s_\ell - s_k$). The accuracy comparison is based on the mean accuracy across the eight commonsense reasoning tasks for seed 42.

The results in Table 5 indicate that historical representations from different depths contribute unequally to the current low-rank update, making dynamic and task-relevant Historical-Layer Attention retrieval conditioned on the current-layer query essential for effective historical information utilization. Moreover, when the attention aggregation mechanism is fixed, the Cross-Layer State Difference performs better than directly using the raw historical state. This finding is consistent with the preceding mechanism analysis: rather than directly reusing historical representations that contain substantial redundant information, the difference-based construction provides the adapter with a more focused representation-evolution signal that better matches the needs of the current update. Further ablation studies are provided in Appendix B.

Table 5: Component ablations for historical-information integration on the commonsense reasoning task.

Variant	**Avg. (%)**
LoRA	78.69
+ mean s_k	78.37
+ attn. over s_k	78.95
+ mean $(s_{k+1} - s_k)$	77.04
+ attn. over $s_{k+1} - s_k$	79.08
+ attn. over $s_\ell - s_k$	79.60

5 Conclusion

To address the isolation of layer-wise adapters in PEFT and their limited use of historical representations, we propose HiLA, a low-rank adaptation method based on Historical-Layer Attention. HiLA constructs cross-layer state differences and uses the current adapter's low-dimensional bottleneck representation as a query to dynamically retrieve and inject representation-evolution information from preceding layers.

With a trainable parameter count nearly identical to standard LoRA, HiLA consistently improves performance across mathematical reasoning, commonsense reasoning, and open-domain dialogue generation. On challenging multi-step mathematical reasoning tasks, it significantly outperforms existing low-rank baselines while tuning only about 1% of the parameters and surpasses full fine-tuning on some evaluations. These results highlight the value of cross-layer historical information for PEFT and establish a representation-level approach to cross-layer collaboration.

In future work, we will explore sparse historical-layer selection and efficient key-value cache mechanisms to further reduce computational and memory overhead. We will also investigate the scalability of HiLA to larger backbone models and long-context tasks.

AI USE STATEMENT

Generative AI tools were used to assist with research ideation, literature retrieval and discovery, implementation and debugging, experimental scripting, figure and table formatting, and manuscript polishing to improve the clarity, readability, and linguistic presentation of the paper.

Generative AI tools were not used to invent, fabricate, or manually alter any experimental numerical results. All quantitative results reported in this paper were obtained from actual experimental runs and evaluation procedures. AI-assisted outputs related to code, data processing, experimental procedures, and citations were reviewed and verified by the authors.

The authors take full responsibility for the implementation, experiments, reported results, and final manuscript, including all content prepared or refined with the assistance of generative AI tools.

REPRODUCIBILITY STATEMENT

We provide detailed training configurations, initialization schemes, and implementation environments in Appendix A. Its subsections on dataset statistics (Appendix A.4) and evaluation protocol (Appendix A.5) provide the corresponding data statistics and deterministic evaluation procedures. Complete implementation details of HiLA, including cross-layer state-difference construction, historical rank, RMSNorm placement, projection sharing, and injection modules, are described in the Method section and Appendix A. Additional ablation experiments on signal sources, injection locations, RMSNorm placement, and the LoRA–historical-rank configuration are reported in Appendix B. Additional mechanistic experiments and analyses of parameter count, latency, throughput, and GPU memory usage are provided in Appendices C and D. Anonymous source code is available at `https://anonymous.4open.science/r/HiLA-8209/` or in the supplementary material.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. URL https://arxiv.org/abs/1610.01644.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72, 2005. URL https://aclanthology.org/W05-0909/.

Yonatan Bisk, Rowan Zellers, Jianbang Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3074–3085, 2019. URL https://aclanthology.org/N19-1300/.

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. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Rei Nakano, et al. Training verifiers to solve math word problems. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2110.14168.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass, 2023.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander H. Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur D. Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander I. Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. The second conversational intelligence challenge (ConvAI2). *arXiv preprint arXiv:1902.00098*, 2019. URL https://arxiv.org/abs/1902.00098.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. Gora: Gradient-driven adaptive low rank adaptation. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-3805. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a5e4907a40c0dcb8433a35c714ba9d79-Abstract-Conference.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123. URL https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html.

Zhiwei He, Zhaopeng Tu, Xing Wang, Xingyu Chen, Zhijie Wang, Jiahao Xu, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, and Rui Wang. RaSA: Rank-sharing low-rank adaptation. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=GdXI5zCoAt.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2103.03874.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qiushi Huang, Xubo Liu, Tom Ko, Bo Wu, Wenwu Wang, Yu Zhang, and Lilian Tang. Selective prompting tuning for personalized conversations with LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 16212–16226, 2024. doi: 10.18653/v1/2024.findings-acl.959. URL https://aclanthology.org/2024.findings-acl.959/.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024. doi: 10.48550/arXiv.2405.12130. URL https://arxiv.org/abs/2405.12130.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, 2004. URL https://aclanthology.org/W04-1013/.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Danny Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024. doi: 10.48550/arXiv.2402.09353. URL https://arxiv.org/abs/2402.09353.

Llama Team. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/081be9fdff07f3bc808f935906ef70c0-Abstract.html.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper/2024/hash/db36f4d603cc9e3a2a5e10b93e6428f2-Abstract-Conference.html.

Meta AI. Llama 3.2 model card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

MLCommons. Llama 2 70B: An MLPerf inference benchmark for large language models. https://mlcommons.org/2024/03/mlperf-llama2-70b/, 2024.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002. URL https://aclanthology.org/P02-1040/.

Jason Phang, Haokun Liu, and Samuel R. Bowman. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 529–538, Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.blackboxnlp-1.42. URL https://aclanthology.org/2021.blackboxnlp-1.42/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, et al. Mlperf inference benchmark. *arXiv preprint arXiv:1911.02549*, 2019. doi: 10.48550/arXiv.1911.02549. URL https://arxiv.org/abs/1911.02549.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019. URL https://aclanthology.org/D19-1454/.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/skean25a.html.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

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

Bangguo Ye, Yuanwei Zhang, and Xiaoqun Zhang. TRAC: Tensor-train based across-layer compression for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tz5yPWZp9W.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023. URL https://arxiv.org/abs/2309.12284.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/06a52a54c8ee03cd86771136bc91eb1f-Abstract-Conference.html.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://iclr.cc/virtual_2020/poster_SkeHuCVFDr.html.

Yuhua Zhou, Ruifeng Li, Changhai Zhou, Fei Yang, and Aimin Pan. BSLoRA: Enhancing the parameter efficiency of LoRA with intra-layer and inter-layer sharing. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78883–78902. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhou25k.html.

zwhe99. Commonsense-170k. Hugging Face Datasets, 2024. URL https://huggingface.co/datasets/zwhe99/commonsense_170k.

A TRAINING DETAILS

The following sections first provide the training, implementation, and evaluation details that support the main experiments, and then present supplementary ablations, mechanistic diagnostics, and resource measurements.

A.1 TASK SELECTION LOGIC

We select three complementary task families to evaluate distinct aspects of parameter-efficient adaptation:

- **Mathematical reasoning:** This task tests whether explicitly leveraging historical representation changes that emerge with network depth improves adaptation in complex multi-step reasoning scenarios.
- **Commonsense reasoning:** This task tests stable generalization across different task types and knowledge dimensions. The eight benchmarks used in the paper cover multiple types of commonsense reasoning, including true-or-false judgment, physical reasoning, social reasoning, and scientific reasoning.
- **Open-domain dialogue generation:** This task tests the applicability of the method to non-answer-oriented, free-form, autoregressive, sequence-level text generation scenarios.

A.2 TRAINING HYPERPARAMETERS

All PEFT methods freeze the base model and update only the adapter parameters. Unless otherwise specified, the target modules, training budget, and evaluation protocol follow Table 6, with task-specific settings in Table 7; method-specific parameterization and initialization remain as described above.

FFT under the unified training configuration. For a controlled comparison, the Full Fine-Tuning (FFT) baseline follows the unified training configuration used by the PEFT baselines, except for its learning rate. Because FFT updates all pretrained backbone parameters, it uses the smaller learning rate of 5×10^{-5} to avoid disrupting pretrained representations, following GoRA (He et al., 2025a). The optimizer, scheduler, warm-up strategy, batch size, gradient accumulation steps, number of training epochs, checkpoint-selection rule, decoding configuration, and evaluation scripts remain unchanged.

Table 6: Unified training configuration for PEFT methods and FFT.

Configuration	Setting
Optimizer	AdamW
Learning rate	PEFT: 2×10^{-4}; FFT: 5×10^{-5}; cosine decay
Weight decay	Commonsense 0.01; Math/Dialogue 0
Adam parameters	$\beta_1 = 0.9,\ \beta_2 = 0.95,\ \epsilon = 10^{-8}$
Gradient clipping	Maximum norm 1.0
Precision	bfloat16
Effective batch size	32
Target modules	$\mathbf{W}_Q,\ \mathbf{W}_K,\ \mathbf{W}_V,\ \mathbf{W}_{\text{up}},\ \mathbf{W}_{\text{down}}$
LoRA configuration	$r = 32,\ \alpha = 64$, dropout 0.05
HiLA history rank	$r_h = 4$
History normalization	Non-affine RMSNorm on query/key/value branches
Projection sharing	Share historical projections across shape-matched branches
Training objective	Teacher-forced causal language modeling; no decoding temperature is used in the loss
Checkpoint selection	Held-out validation loss every 100 optimization steps; retain the lowest-loss checkpoint
Training-time free generation	Diagnostic snapshot only: `do_sample=false`, $T = 1.0$, `num_beams=1`; not used for reported metrics

Table 7: Task-specific training settings.

Task	Training set	Epochs	Max length	Eff. batch	Warmup	Seeds
Commonsense	Commonsense-170K	2	256	32	10%	41–45
Math	MetaMathQA	2	512	32	10%	42
Dialogue	ConvAI2	1	1,024	32	100 steps	41–45

A.3 Implementation Environment

LoRA initializes $\mathbf{A}$ with Kaiming-uniform initialization (He et al., 2015) and $\mathbf{B}$ with zeros. HiLA applies the same initialization to the historical query and key projections, while initializing the value projection to zero. RMSNorm contains no learnable affine parameters. All runs are conducted on NVIDIA A100 80GB GPUs with Python 3.10 and PyTorch 2.1.1 (CUDA 12.1). Each random seed is assigned to a single GPU. Checkpoint selection follows the preceding unified configuration. Training cost is measured using the actual runtime, excluding queueing and data download time.

A.4 Dataset Statistics and Evaluation

Table 8 reports dataset scale and evaluation coverage for the three task families. Commonsense reasoning evaluation uses the publicly available OpenCompass framework (Contributors, 2023) and covers eight benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), ARC-Challenge and ARC-Easy (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). This table provides their aggregate evaluation count together with the corresponding training and evaluation data for the remaining task families.

Table 8: Dataset statistics and evaluation scope.

Task family	Training data	Evaluation data	Metric
Commonsense	Commonsense-170K ($\approx$ 170K)	Eight official suites; 22,419 examples in total	Per-suite accuracy and macro average
Math	MetaMathQA ($\approx$ 395K)	GSM8K (1,319) and MATH (5,000)	Exact-match accuracy
Dialogue	ConvAI2, 33,518 turns	1,000 dialogs, 13,716 turns	BLEU, BERTScore, METEOR, ROUGE-L

A.5 Evaluation Protocol

All tasks use fixed data manifests and deterministic evaluation configurations. For every run, checkpoints are selected according to validation loss: validation loss is evaluated every 100 optimization steps, and the checkpoint with the lowest observed validation loss is retained. The complete training objectives, decoding strategies, sampling settings, temperature, and length configurations are provided in Table 9. Commonsense results are obtained by matching the official answer format and taking the unweighted mean over the eight subsets. Mathematical reasoning results use exact match after unified numerical or expression normalization, while dialogue results are computed using the metrics reported in the main text.

Prompt Construction. The models for mathematical reasoning and commonsense reasoning are trained on MetaMathQA (Yu et al., 2023) and Commonsense-170K (zwhe99, 2024), respectively, and use Alpaca-style instruction wrappers on the corresponding evaluation sets. When additional context is available, the prompt concatenates, in order, "Below is an instruction that describes a task, paired with an input that provides further context.", `### Instruction:`, the question, `### Input:`, the context, and `### Response:`. Otherwise, the Input section is omitted. For GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), `Let's think step by step.` is appended after the Response marker. Neither training answers nor gold labels are appended to the evaluation prompts.

For ConvAI2 (Dinan et al., 2019), each turn to be predicted uses a structured dialogue context as the input text. The input contains the current speaker, only that speaker's persona, at most the eight most recent dialogue turns, and the instruction `Respond as [speaker] with one natural conversational reply.`. To mark the boundary between the input and the generated response, both training and evaluation retain only the generic `### Instruction:` and `### Response:` markers. The system prompt is empty, and the other speaker's persona is excluded from the input.

Answer Extraction and Evaluation. For commonsense evaluation, we first retain the text after the final `### Response:` marker and then use case-sensitive regular expressions to extract the first valid label. BoolQ uses `true/false`, PIQA uses `solution1/solution2`, SocialIQA, ARC-Challenge, ARC-Easy, and OpenBookQA use `answer1–answer5`, HellaSwag uses `ending1–ending4`, and WinoGrande uses `option1/option2`. A prediction is counted as correct only when the extracted label exactly matches the official answer string in the data file.

For GSM8K, both predictions and reference answers first attempt to match `#### <number>`. If no such match is found, the extractor sequentially searches for `the answer is <number>` and `answer: <number>`. The matched integer may contain a negative sign and thousands separators, and commas are removed before comparison. For MATH, both predictions and reference answers preferentially use the content of the final $\boxed{\cdot}$ or `fbox` expression. If a prediction does not contain a boxed answer, the extractor falls back to the segment containing `final answer` or `answer is`, and then to the first sentence of the generated text. Spaces, dollar signs, and common LaTeX wrappers are then removed, while fractions, square roots, commas, and units are normalized before string-equivalence comparison. Each MATH reference solution is required to contain a boxed final answer.

ConvAI2 does not use discrete answer extraction. Instead, the complete generated response is retained and compared with the reference response using BLEU, BERTScore-F1, METEOR, and ROUGE-L. Therefore, dialogue evaluation operates through metric-level text comparison rather than the exact label matching used for commonsense reasoning or the normalized exact match used for mathematical reasoning.

Table 9: Decoding and temperature settings for training and evaluation. Training uses teacher forcing without autoregressive sampling; evaluation follows the reported scripts.

Task	Training objective	Evaluation decoding	Generation settings
All tasks	Teacher-forced causal LM; temperature is not applicable to the loss	–	No autoregressive sampling is used during optimization
Commonsense	Same as above	Deterministic 4-beam search	`do_sample=false`, `num_beams=4`, $T = 0.1$, top-$p = 0.75$, top-$k = 40$, max new tokens $= 32$; T/top-p/top-k are non-operative when sampling is disabled
GSM8K	Same as above	Greedy decoding	`do_sample=false`, implicit `num_beams=1`, temperature not set at runtime, max input length $= 1{,}024$, max new tokens $= 512$
MATH	Same as above	Greedy decoding	`do_sample=false`, implicit `num_beams=1`, temperature not set at runtime, max input length $= 1{,}024$, max new tokens $= 2{,}048$
Dialogue	Same as above	Greedy decoding	`do_sample=false`, `num_beams=1`, temperature/top-p/top-k not set, max input length $= 1{,}024$, max new tokens $= 64$

The temperature field in the table distinguishes between the value explicitly recorded in the configuration and the temperature that actually affects sampling. Although the commonsense evaluation script records $T = 0.1$, top-$p = 0.75$, and top-$k = 40$, it also fixes `do_sample=false`; therefore, these parameters do not affect the generation distribution. The GSM8K, MATH, and ConvAI2 scripts likewise do not pass a temperature parameter to the generator and instead perform deterministic greedy decoding. Free-generation snapshots produced by the training script are used only for diagnostics, and neither the main training procedure nor the metrics reported in the paper depend on these snapshots.

A.6 BSLoRA Supplementary Comparison

To further evaluate a cross-layer baseline based on parameter sharing, we include a task-wise comparison between BSLoRA and HiLA using the same column order and official test sets as the main commonsense reasoning table. The HiLA row reuses the results reported in the main table, while the BSLoRA row uses results obtained with the current official PEFT implementation under its fixed configuration ($r = 32$, $r_{\text{local}} = 2$, $r_{\text{intra}} = 4$, $r_{\text{inter}} = 16$, with Kronecker sharing). The final column reports the percentage-point difference between BSLoRA and HiLA, defined as $\Delta = \text{BSLoRA} - \text{HiLA}$, where a negative value indicates that BSLoRA performs worse than HiLA.

Table 10: Supplementary comparison on commonsense reasoning between HiLA and BSLoRA (%). Δ denotes the percentage-point difference of BSLoRA relative to HiLA.

Model	Method	Trainable (%)	BoolQ	PIQA	SIQA	HS	WG	ARC-c	ARC-e	OBQA	Avg.	Δ Avg.
Qwen2.5-1.5B	HiLA	1.6083	68.35	83.08	75.49	85.94	74.19	76.54	89.23	84.04	79.61	–
	BSLoRA	0.1049	62.17	76.22	70.16	82.70	72.77	74.32	88.85	79.00	75.77	−3.84
Qwen3-8B	HiLA	0.7219	73.15	90.04	81.83	95.22	85.16	91.38	97.10	91.48	88.17	–
	BSLoRA	0.0462	71.87	89.39	80.09	93.98	83.50	91.13	96.80	90.00	87.10	−1.07
Llama-3.2-3B	HiLA	1.0315	70.55	86.51	80.81	90.80	85.00	76.96	89.02	84.12	82.97	–
	BSLoRA	0.0665	66.79	83.62	74.56	86.45	81.14	68.77	83.75	76.40	77.69	−5.28
Llama-3.1-8B	HiLA	0.7070	75.84	89.99	79.94	94.92	86.90	81.14	91.37	86.24	85.79	–
	BSLoRA	0.0452	70.95	87.21	79.22	90.87	84.21	79.52	90.70	83.20	83.24	−2.55

B Additional Ablation Experiments

This section provides additional ablations on the signal source, RMSNorm placement, and the grid of LoRA and historical ranks to characterize the fixed main configuration. Unless otherwise noted, all non-ablated settings and checkpoint selection follow the unified protocol in Appendix A.2.

B.1 Ablation on the Signal Source

To determine the representation from which the historical readout should extract its signal, we compare two sources: the intermediate representation $\mathbf{z}$ of the low-rank branch and the direct input $\mathbf{h}$ to the target linear transformation. All other architectural settings, training budgets, and evaluation protocols remain unchanged. As shown in Table 11, using $\mathbf{z}$ yields an average ConvAI2 score of 54.38%, whereas using the direct input $\mathbf{h}$ reduces it to 52.74%. This result indicates that retrieving historical information in the low-rank space better aligns the historical signal with the update direction of the adaptation branch.

Table 11: Historical signal-source ablation on Llama-3.2-3B and ConvAI2.

Historical signal source	Avg. (%)
$\mathbf{z}$ (low-rank representation)	**54.38**
$\mathbf{h}$ (direct linear-layer input)	52.74

B.2 Ablation on the Injection Module

Using the preceding main configuration, we independently control whether the q, k, v, o, g, and up branches receive historical signals to identify the target linear transformations for which historical readout is effective. Because the input dimension of W_{down} does not match the dimension required for historical readout injection, W_{down} retains the standard LoRA branch in all settings and is not included as an injection variable. Checkmarks in Table 12 indicate that historical readout is enabled for the corresponding branches.

The results show that the effectiveness of historical information is strongly module-dependent. Enabling historical readout only for the $q/k/v$ and up branches achieves the highest average accuracy of 79.60%. When historical readout is applied only to the $q/k/v$ branches while up retains standard LoRA, the average accuracy decreases to 78.42%. This result suggests that cross-layer information

benefits not only representation selection and interaction within the attention mechanism, but also provides complementary representation updates through the up-projection in the feed-forward network. In contrast, extending historical readout to the o or g branch reduces the average accuracy to 77.84% and 77.21%, respectively, while enabling it for all six injectable branches further decreases the accuracy to 76.63%. These results indicate that historical information is not equally effective across all linear transformations, and excessive propagation of historical signals may interfere with the local transformations performed by certain branches. Overall, the $q/k/v + up$ configuration provides a better balance between cross-layer information utilization and preservation of local representations, and is therefore adopted as the default injection configuration for HiLA.

Table 12: Injection-module ablation on Qwen2.5-1.5B.

q	k	v	o	g	up	**Avg. (%)**
✓	✓	✓	×	×	✓	**79.60**
✓	✓	✓	✓	×	✓	77.84
✓	✓	✓	×	✓	✓	77.21
✓	✓	✓	✓	✓	✓	76.63
✓	✓	✓	×	×	×	78.42
×	×	×	×	×	✓	76.97

B.3 Ablation on RMSNorm Placement

To analyze the role of RMSNorm in historical attention, we control whether RMSNorm is applied to the query, key, and value branches. Table 13 summarizes the performance of Llama-3.2-3B on ConvAI2 open-domain dialogue generation. The results show that applying RMSNorm to all three branches achieves the best performance. Disabling RMSNorm on any branch leads to varying degrees of degradation, with the largest drop occurring when RMSNorm is removed from the value branch. These results suggest that branch normalization helps stabilize the numerical range of the historical readout.

Table 13: RMSNorm placement ablation.

q	k	v	**Avg. (%)**
✓	✓	✓	**54.38**
×	✓	✓	<u>54.24</u>
✓	×	✓	54.17
✓	✓	×	53.20

B.4 Rank Analysis

Table 14 summarizes the trainable parameter ratios and average commonsense reasoning accuracy under different combinations of LoRA rank and historical attention rank.

Table 14: Supplementary ablation of LoRA rank r and historical attention rank r_h on Qwen2.5-1.5B.

LoRA rank (r)	**Historical rank** (r_h)	**Trainable (%)**	**Avg. (%)**
16	4	0.8044	76.65
	8	0.8048	77.10
	16	0.8056	**77.35**
	32	0.8072	<u>77.29</u>
32	4	1.6083	**79.60**
	8	1.6087	<u>79.41</u>
	16	1.6095	79.34
	32	1.6112	79.29
64	4	3.2163	**81.25**
	8	3.2167	<u>80.95</u>
	16	3.2175	80.45
	32	3.2192	80.30

Increasing r improves the average accuracy, whereas further increasing r_h with r fixed does not yield consistent gains and introduces additional parameters. To balance parameter efficiency and performance, we adopt $r = 32$ and $r_h = 4$ as the unified configuration for the main experiments.

C Additional Mechanistic Analysis

This section supplements the diagnostic evidence summarized in the main text by detailing the Activation Patching, layerwise Linear Probing, and shared-component analysis protocols. All diagnostics are conducted in inference mode, with no updates to the base model or adapter parameters.

C.1 Activation Patching

Prediction-recovery activation patching. Activation Patching is a counterfactual intervention method for testing whether an intermediate representation lies on an effective information pathway that can influence the final prediction. While keeping the model parameters, all other input positions, and subsequent computation unchanged, it replaces the activation produced by a corrupted input at a specific intermediate position with the corresponding activation from the clean input, and then examines whether the final output is restored. Because Activation Patching directly intervenes on a specific intermediate state during the forward pass, it provides a targeted measure of the transmissible contribution of a representation at a given depth to the final prediction.

Let the clean and corrupted inputs differ at position t^*, and let the input states of the ℓ-th Transformer block be $\mathbf{h}_{\ell,t^*}^{\text{clean}}$ and $\mathbf{h}_{\ell,t^*}^{\text{corr}}$, respectively. At layer ℓ, we replace the state at the corresponding position on the corrupted path with

$$\widetilde{\mathbf{h}}_{\ell,t^*} = \mathbf{h}_{\ell,t^*}^{\text{clean}}, \tag{9}$$

while the states at all other positions remain on the corrupted path. If this replacement restores the logit margin of the correct answer over the alternative answer, the state at that layer contains information that can propagate through subsequent computation and influence the final prediction. We quantify recovery using

$$R_\ell = \frac{m_\ell - m_{\text{corr}}}{\max(m_{\text{clean}} - m_{\text{corr}}, 10^{-6})} \tag{10}$$

where m denotes the logit margin between the correct token and the alternative token, m_ℓ is the margin after patching the state at layer ℓ, and m_{clean} and m_{corr} denote the clean and corrupted baselines, respectively.

We perform inference on a frozen Llama-3.1-8B. For each Transformer block, we apply the intervention independently by replacing only the block-input hidden state at the differing token position on the corrupted path. All other token positions, the attention mask, model weights, and subsequent

computation remain unchanged. The recovery rate for each layer is computed according to equation 10, and uncertainty is reported using bootstrap confidence intervals. This experiment therefore quantifies the causal transmissibility of the representation at a specified depth to the final prediction.

Layer-local matched-state activation patching. To exclude confounding from upstream-error propagation in the recovery experiment when comparing the *layer-local downstream* influence of representations at different depths, we conduct a matched-state intervention using the same clean/corrupted pairs, intervention position, and frozen Llama-3.1-8B. For every example, we first record both the clean and corrupted trajectories. Then, for every Transformer block ℓ, we re-run the same clean trajectory and replace only the block-input hidden state of the critical token t^* with its matched corrupted state $\mathbf{h}_{\ell,t^*}^{\text{corr}}$. All other token states, the attention mask, model weights, and all subsequent host computation remain on the clean path. Restoring $\mathbf{h}_{\ell,t^*}^{\text{clean}}$ at the same site exactly restores the clean margin, which serves as an implementation check.

Let $m_\ell^{\text{local-corr}}$ denote the logit margin of the correct token over the alternative token after this single-layer replacement. We define layer-local damage as

$$D_\ell = m_{\text{clean}} - m_\ell^{\text{local-corr}}. \tag{11}$$

Figure 6 shows that local damage is largest in shallow and middle layers and declines substantially with depth: the mean damage is 12.35 at layer 0 and 0.94 at layer 31, a 92.4% reduction. Because every intervention begins from the same clean host trajectory, this fixed-path counterfactual intervention provides strong causal evidence for the claim that early critical-token representations exert greater local downstream influence on the final answer preference than deep representations; the depth trend is not merely produced by shallow corruption propagating earlier.

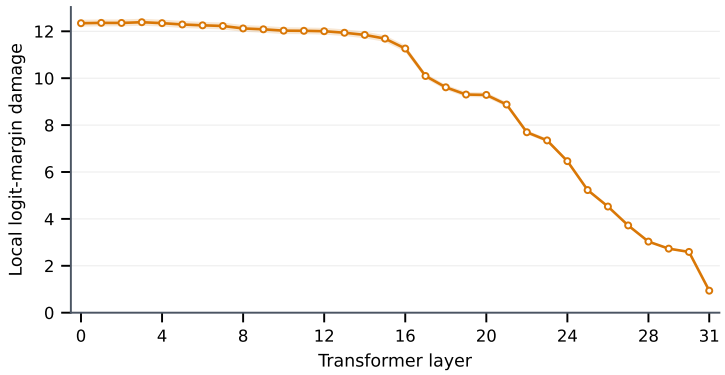


Figure 6: Layer-local matched-state activation intervention on frozen Llama-3.1-8B ($4{,}096$ secret-token copy pairs). Local logit-margin damage declines with depth; shading denotes 95% bootstrap confidence intervals.

C.2 Layerwise Linear Probing

Layerwise Linear Readout analyzes how task-relevant information emerges and becomes readable across Transformer depth, following the intermediate-layer Linear Probe paradigm (Alain & Bengio, 2016). For Frozen, LoRA, RaSA, BSLoRA, and HiLA, we extract features from the hidden state of the last valid token at each layer and independently train a single-logit Linear Probe without hidden layers for that layer to score each of the four candidate endings. Before training, features are standardized using statistics computed only from the training set. At test time, the probe score is computed for each of the four candidate endings, and the candidate with the highest score is selected as the prediction. We compare all layers of each model and report four-way selection accuracy on the held-out test set.

C.3 Additional Analysis of Cross-Layer Shared Components

To complement the shared-component analysis in Section 4.3, we use the cross-layer difference $\mathbf{D}_{k,\ell} = \mathbf{H}_\ell - \mathbf{H}_k$ to quantify the structural overlap that remains after differencing. Specifically, we

aggregate each vector-valued difference into the scalar

$$S_{k,\ell}^{\text{hist}} = \text{CKA}\left(\mathbf{D}_{k,\ell}, \mathbf{H}_k\right). \tag{12}$$

Lower values of $S_{k,\ell}^{\text{hist}}$ indicate that the difference retains less of the historical representation's shared structure. Figure 4 reports this diagnostic alongside the baseline CKA; the observed reduction is consistent with cross-layer differencing suppressing structurally redundant components while retaining a depth-specific signal for historical readout.

Figure 7 reports the original layer-pair diagnostics on Llama-3.1-8B. For every historical–current pair with $k < \ell$, the left panel reports the Linear CKA between the centered hidden-state matrices $\mathbf{H}_k$ and $\mathbf{H}_\ell$. The middle and right panels report the improvement in reconstruction over a shuffled control and the held-out reconstruction R^2, respectively. Together, these diagnostics show that adjacent and intermediate layers share substantial structure and that part of the current representation is linearly predictable from historical states.

We further evaluate how much of a current representation can be recovered by a linear transformation of a historical representation. For each pair (k, ℓ), we fit the ridge-regression map $\mathbf{W}_{k\to\ell}^*$ defined in Section 4.3 and form $\widehat{\mathbf{H}}_\ell = \mathbf{H}_k \mathbf{W}_{k\to\ell}^*$. We quantify reconstruction quality using the coefficient of determination

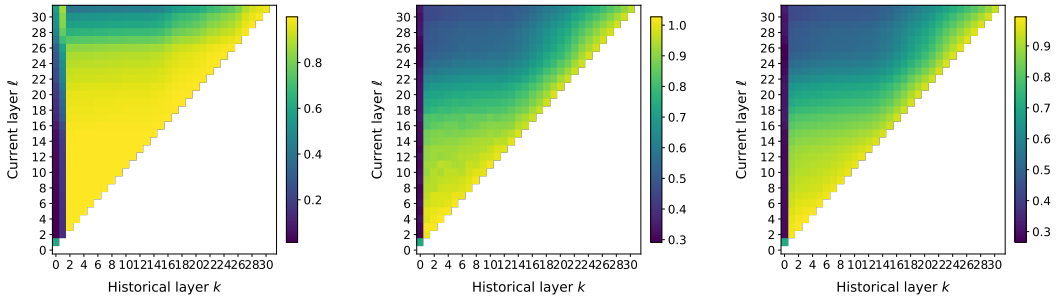
$$R_{\ell,k}^2 = 1 - \frac{\left\|\mathbf{H}_\ell - \widehat{\mathbf{H}}_\ell\right\|_F^2}{\left\|\mathbf{H}_\ell - \overline{\mathbf{H}}_\ell\right\|_F^2}, \tag{13}$$

where $\overline{\mathbf{H}}_\ell$ is the row-wise mean baseline for the current-layer representations. The right panel of Figure 7 reports this linear R^2 across layer pairs. High reconstruction scores for nearby layers show that a substantial fraction of the current representation is linearly predictable from historical states. This result is consistent with the interpretation that cross-layer differencing reduces the contribution of predictable shared content and directs historical readout toward representation changes rather than redundant state content.

To distinguish genuine cross-layer predictability from the effect of fitting a high-dimensional linear map, we construct a shuffled control by randomly permuting the historical-state rows before fitting the same ridge-regression procedure. Let $R_{\ell,k}^{2,\text{shuffle}}$ denote its held-out reconstruction score. The middle panel reports the improvement over this control,

$$\Delta R_{\ell,k}^2 = R_{\ell,k}^2 - R_{\ell,k}^{2,\text{shuffle}}. \tag{14}$$

A positive $\Delta R_{\ell,k}^2$ therefore shows that the historical–current correspondence, rather than an arbitrary pairing of samples, accounts for the reconstruction quality.



(a) Layerwise Linear CKA. (b) Reconstruction above shuffled control (ΔR^2). (c) Cross-layer reconstruction R^2.

Figure 7: Supplementary shared-component diagnostics on Llama-3.1-8B. Rows index the current layer ℓ and columns index the historical layer k. The three panels report layerwise Linear CKA, reconstruction improvement above a shuffled historical-state control (ΔR^2), and the held-out R^2 of ridge-regression reconstruction of the current representation from the historical representation, respectively.

D Resource Analysis

D.1 Efficiency Benefits of HiLA Optimization

To characterize the practical impact of our HiLA implementation optimizations, we compare the pre-optimization implementation with the optimized implementation on Llama-3.2-3B. Both implementations use the default injection configuration specified in Appendix B.2. The pre-optimization implementation instantiates independent historical projections for compatible layers and branches, whereas the optimized implementation incorporates the optimizations described below. The backbone contains 28 Transformer layers with hidden dimension $d = 3072$.

Under this configuration, the pre-optimization implementation contains $45{,}430{,}784$ trainable adapter parameters, whereas the optimized implementation contains $33{,}140{,}864$.

Table 15: Impact of HiLA implementation optimizations on Llama-3.2-3B. Both implementations use the default injection configuration from Appendix B.2. Additional parameters are relative to the default LoRA configuration; speed-up is measured relative to the pre-optimization implementation.

Implementation	Additional parameters	Trainable (%)	E2E latency (ms)	Speed-up
$\text{HiLA}_{\text{previous}}$	12.40M	1.4141	1,984.78	$1.00\times$
$\text{HiLA}_{\text{optimized}}$	0.111M	1.0315	455.66	$\mathbf{4.35}\times$

Relative to the pre-optimization implementation, optimized HiLA reduces the number of additional parameters by approximately 99.1% and trainable adapter parameters by approximately 27.1%. Its end-to-end latency decreases from $1{,}984$ ms to 455.66 ms, corresponding to a $4.35\times$ speed-up and a 77.0% reduction in latency.

The optimized implementation incorporates three practical optimizations:

1. Historical query, key, and value projections are shared across shape-compatible adapter modules and branches, substantially reducing duplicated parameters.
2. Hidden states already computed during the backbone forward pass are reused for historical retrieval, avoiding redundant hidden-state computation.
3. HiLA applies a structure-specific FlashAttention (Dao et al., 2022) mechanism to its historical-layer attention, fusing query–key matching, normalization, and weighted aggregation over historical layers to reduce intermediate memory traffic and runtime overhead.

D.2 Parameter Count and Latency

Given that introducing attention and cross-layer aggregation increases both the parameter count and computational overhead, HiLA controls the additional parameters by sharing the historical readout projection matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ across modules with the same input dimension, LoRA rank, and historical rank. Accordingly, for a sharing group with input dimension d, the additional parameter count is $r_h r + r_h d + rd$, which does not grow linearly with the number of Transformer layers or injected modules. As shown in Table 16, HiLA has a trainable parameter percentage of 1.03% on Llama-3.2-3B, comparable to those of LoRA, MoRA, and DoRA. This result indicates that the performance gains of HiLA do not arise from a substantially larger parameter budget.

The latency measurements were obtained on an NVIDIA A100 80GB using MLPerf Inference LoadGen in the SingleStream, PerformanceOnly scenario. We evaluated 1,024 timed queries from the ConvAI2 test split after two warm-up queries, with an effective batch size of one (sequential requests), BF16 computation, SDPA for the base attention, a fixed seed of 41, and at most 64 generated tokens per query. The HiLA run used shared historical projections and FlashAttention for historical-layer attention, with `torch.compile` disabled.

For inference efficiency, we follow the general benchmarking framework of MLPerf Inference (Reddi et al., 2019) and the LLM-specific latency and throughput reporting convention subsequently adopted by MLPerf (MLCommons, 2024). We report TTFT, TPOT, end-to-end latency, request throughput, and token throughput. HiLA achieves TTFT, TPOT, and end-to-end latency of

$49.56\,\mathrm{ms}$, $47.75\,\mathrm{ms}$, and $455.66\,\mathrm{ms}$, respectively, all lower than those of DoRA, while improving request throughput and token throughput by $0.42\,\mathrm{req/s}$ and $4.18\,\mathrm{tok/s}$, respectively. Although HiLA incurs higher latency than LoRA and MoRA, its efficiency advantage lies in achieving stronger task performance under a parameter budget close to that of LoRA while maintaining lower overall inference overhead than DoRA despite the additional historical readout. During gradient-free inference, HiLA reuses hidden states, shares historical key and value projections across adapter branches, and employs FlashAttention (Dao et al., 2022) to reduce high-bandwidth memory access.

Table 16: Trainable-parameter ratio, peak GPU memory, and latency on Llama-3.2-3B.

Method	Trainable (%)	Peak GPU Mem. (GB)	TTFT (ms)	TPOT (ms)	Latency (ms)	$\mathrm{TP}_{\mathrm{req}}$ (req/s)	$\mathrm{TP}_{\mathrm{tok}}$ (tok/s)
LoRA	1.0281%	**6.19**	**25.16**	**24.42**	**242.33**	**4.13**	**40.81**
MoRA	1.0262%	<u>6.23</u>	<u>33.87</u>	<u>33.50</u>	<u>329.55</u>	<u>3.03</u>	<u>29.94</u>
DoRA	1.0424%	6.35	52.26	59.37	566.10	1.77	16.65
HiLA	1.0315%	6.30	49.56	47.75	455.66	2.19	20.83

D.3 Peak GPU Memory

Table 17 reports the peak forward-pass GPU memory for different frozen backbones and adaptation methods, providing a complementary measure of resource overhead alongside the parameter count and latency analyses.

Table 17: Peak forward-pass GPU memory for frozen backbones and adaptation methods (GB).

Frozen backbone	LoRA	DoRA	MoRA	PiSSA	HiLA
Qwen2.5-1.5B	3.09	3.12	3.08	3.09	3.11
Qwen3-8B	15.60	15.91	15.58	15.60	15.83
Llama-3.2-3B	6.19	6.35	6.23	6.18	6.30
Llama-3.1-8B	15.26	15.65	15.29	15.26	15.62