DepthGraft: Structural Regularization through Hierarchical Cross-Layer KV Reconstruction

Xiaohan Qin[1,2,3], **Xiangdong Zhang**[1,3], **Yu Wang**[3], **Huaijin Wu**[1,3],
Zhuo Xia[1,3], **Yebin Yang**[1,3], **Junchi Yan**[1,†]
[1]School of AI, Shanghai Jiao Tong University, [2]Shanghai Innovation Institute
[3]Dots Studio, Xiaohongshu Inc.

Abstract

Scaling depth is a key route to stronger large language models, but Pre-LN architectures often suffer from depth-wise information dilution, limiting the effective use of deep layers. While cross-layer connectivity has proven effective at mitigating this problem, existing designs usually introduce additional computation or memory access. We revisit cross-layer KV sharing as an efficient form of such connectivity. Beyond cache compression, each KV reuse edge creates a direct gradient path from deep layers to source KV projections, turning KV sharing into an implicit structural regularizer for KV representation geometry. We provide a systematic analysis of this mechanism, showing that proper sharing topologies sharpen key retrieval directions while enriching the value content subspace, whereas existing adjacent or global-anchor designs do not fully exploit this effect. Motivated by this, we propose DepthGraft, a hierarchical KV-sharing architecture that reconstructs deep-layer KV from stride-aligned shallow–middle source pairs. DepthGraft distributes non-local gradient flow across source layers, while preserving prefill early completion and reducing KV cache by roughly one third. Experiments on dense and MoE models from 0.5B to 16B parameters show consistent downstream improvements under both GQA and MLA. Notably, on the 16B MoE model, DepthGraft achieves a **7.0%** absolute improvement on MMLU-Pro, along with gains of **6.6%** on CMMLU and **6.4%** on C3. These results validate the potential of DepthGraft as a scalable design for large-scale pretraining. The project page is available at https://github.com/jianke0604/DepthGraft.

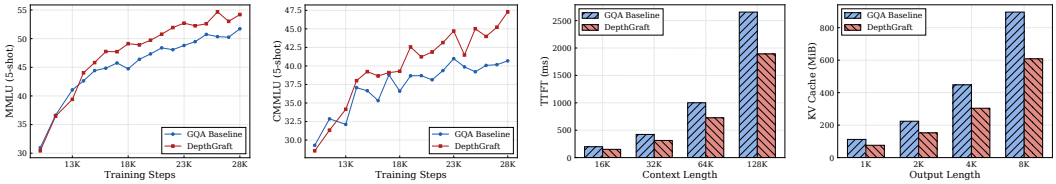


Figure 1: Results on the 16B MoE model. **Left two:** DepthGraft demonstrates a consistent advantage throughout training. **Right two:** DepthGraft reduces prefill time (TTFT) and KV cache usage by approximately one-third, while incurring almost no loss in decoding speed.

1 Introduction

Scaling depth has been a primary lever for improving Transformer-based large language models (LLMs) [47, 36, 24]. In modern Pre-Layer-Normalization Transformers [45], however, adding layers

[†]Corresponding author: yanjunchi@sjtu.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

does not always yield the expected improvements [13, 25]. The residual stream [14] aggregates all preceding computations through a single hidden-state trajectory with fixed additive updates; as the stack grows deeper, features formed in early layers can be progressively diluted and become difficult for later layers to retrieve. This depth-wise information dilution limits the effective use of additional layers, making depth scaling less efficient than its representational capacity would suggest [34, 11].

Recent work has validated *cross-layer connectivity* as an effective remedy for information dilution along depth. Instead of forcing all depth history through the residual stream, recent designs allow deep layers to directly read earlier representations: Attention Residuals [37] reweights preceding hidden states to replace fixed residual accumulation, while Mixture-of-Depths Attention (MoDA) [55] augments attention with key–value pairs from previous layers. Such connections give deep layers direct access to earlier features and create shorter gradient paths to shallow layers. However, dense cross-layer access also introduces additional computation, memory traffic, and system complexity, requiring careful performance–efficiency trade-offs and sophisticated systems-level engineering.

We note that cross-layer KV sharing can also be viewed as a form of such connectivity: it provides deep layers with a channel that *bypasses* information dilution in the residual stream by granting direct access to shallow-layer memory representations, while gradients flow from deep layers back through the shared KV to shallow layers, supplying shallow-layer KV with optimization signals driven by deep-layer task demands—thus holding the potential to improve depth utilization. Although cross-layer KV sharing has been explored in prior work [4, 56, 35], existing methods predominantly approach it from the perspective of cache compression and inference acceleration. No prior work has systematically studied the cross-layer gradient dynamics of KV sharing, especially how such connections reshape KV representation geometry during pretraining and how the sharing topology determines this effect.

In this paper, we propose that KV sharing is not merely a cache-saving mechanism, but also a form of *cross-layer gradient regularization*. When a source layer's KV is reused by a distant deep layer, the source KV receives training signals from multiple query spaces. This reshapes KV representation geometry in an asymmetric way: keys are encouraged to concentrate around shared discriminative retrieval directions, while values are pushed to cover a richer content subspace. With a proper sharing topology, this specialization sharpens memory retrieval while increasing the expressive capacity of the value content channel, yielding a useful structural bias for training. We use this lens to revisit existing KV-sharing methods and find that their topologies expose complementary limitations. Adjacent-layer sharing provides too little query-space separation, whereas global-anchor sharing concentrates heterogeneous deep-layer gradients on a few sources, preventing either design from fully exploiting the regularization effect.

Motivated by this analysis, we propose **DepthGraft**, a simple hierarchical KV-sharing design derived from the cross-layer regularization view. DepthGraft partitions the network into shallow, middle, and deep stages. The shallow and middle stages compute KV normally, while each deep layer reconstructs its KV from a stride-aligned shallow–middle source pair using learnable channel-wise coefficients. This construction creates non-local source–reuse connections with sufficient query-space separation, distributes cross-layer gradients across many source layers rather than a few global anchors, and gives each deep layer access to memory bases at two abstraction levels. As a result, DepthGraft preserves the efficiency benefits of KV sharing, including prefill early completion and KV-cache reduction, while turning the memory pathway into an effective regularized cross-layer connection. Training-dynamics analyses further support our theory, showing that DepthGraft reshapes KV representation geometry as predicted, and experiments across dense and MoE models demonstrate consistent performance gains at multiple scales.

Our main contributions are as follows.

1). KV sharing as cross-layer gradient regularization. We identify KV sharing as a form of cross-layer connectivity and provide a systematic analysis of how its gradient paths reshape KV geometry. Our analysis shows that proper KV sharing sharpens key retrieval directions while enriching the value content subspace, and that this effect is governed by the sharing topology.

2). Hierarchical cross-layer KV reconstruction. We propose DepthGraft, which uses a hierarchical stage layout to reconstruct deep-layer KV from stride-aligned shallow–middle source pairs. This yields a distributed non-local connectivity pattern with two abstraction-level memory bases, avoiding the concentrated-anchor structure of prior KV-sharing methods.

3). Efficient and scalable implementation. DepthGraft removes independent KV materialization from the deep stage, reducing KV cache by roughly $\frac{1}{3}$ and allowing KV construction to finish before the final stage during prefill. Its stride-aligned source assignment is engineering-friendly and naturally compatible with PP/VPP schedules, making the design suitable for large-scale distributed training.

4). Consistent empirical gains. We validate DepthGraft on both dense and MoE models, covering model scales from 0.5B to 16B total parameters. Across scales and attention variants, DepthGraft consistently improves downstream performance while preserving inference efficiency.

2 Related Work

Cross-layer connectivity. A broad line of work enriches depth-wise information flow by giving each layer direct access to earlier representations [17, 19, 20]. DenseNet [17] concatenates all preceding feature maps, establishing a canonical dense cross-layer design. In Transformers, DenseFormer [26] and ANCRe [52] introduce learned layer-wise connection weights, while Value Residual Learning [54] adds targeted residual access to earlier value states. Attention Residuals [37] replaces fixed residual accumulation with learned reweighting over preceding hidden states, and MoDA [55] augments sequence attention with key–value memories from previous layers. Together, these works show that cross-layer connectivity is an effective mechanism for improving deep Transformer architectures [50, 29].

KV sharing for efficient inference. KV cache is a major memory bottleneck during autoregressive inference, motivating many methods that reduce cache storage through KV sharing [42, 49, 41]. Cross-layer KV sharing is an important direction in this line. CLA [4] and MLKV [56] reuse KV within adjacent layer groups, reducing cache but often incurring performance degradation when the reuse constraint becomes too strong. YOCO [35] shares a global KV memory for all deeper layers, enabling prefill early completion and strong cache reduction. FusedKV [23] further reconstructs deep-layer KV from two global anchors, providing more flexibility than direct global reuse. Cross-layer KV sharing is becoming an increasingly important architectural choice for large-scale LLM pretraining, as illustrated by its adoption in DeepSeek-V4.1-Flash [46] and HySparse2 [39].

Positioning. DepthGraft systematically analyzes how cross-layer KV sharing alters training dynamics, showing that the induced cross-layer gradients can regularize KV representation geometry. Guided by this analysis, DepthGraft uses stride-aligned hierarchical reconstruction to improve KV representations and model quality while preserving the KV-cache savings of sharing-based inference.

3 Method

3.1 Preliminaries

Attention and KV cache. Consider an L-layer decoder-only Transformer. We index layers by $\ell \in \{0, \dots, L-1\}$ and use hidden states $\mathbf{X}_\ell \in \mathbb{R}^{T \times D}$ for a sequence of length T. Layer ℓ forms queries, keys, and values as

$$Q_\ell = \mathbf{X}_\ell W_\ell^Q, \qquad K_\ell = \mathbf{X}_\ell W_\ell^K, \qquad V_\ell = \mathbf{X}_\ell W_\ell^V, \tag{1}$$

where $W_\ell^Q \in \mathbb{R}^{D \times H_q d}$, $W_\ell^K, W_\ell^V \in \mathbb{R}^{D \times H_k d}$, with H_q query heads, H_k key-value heads, and per-head dimension d. Under grouped-query attention [1], the H_q query heads are assigned to H_k KV groups. For query head $h \in \{0, \dots, H_q - 1\}$, let $\phi(h) = \lfloor h H_k / H_q \rfloor$ denote its KV group. The attention output is

$$\mathrm{Attn}_\ell = \mathrm{Concat}_{h=0}^{H_q-1} \left(\mathrm{softmax}\!\left(\frac{Q_{\ell,h} K_{\ell,\phi(h)}^\top}{\sqrt{d}} \right) V_{\ell,\phi(h)} \right) W_\ell^O, \tag{2}$$

where $Q_{\ell,h}, K_{\ell,g}, V_{\ell,g} \in \mathbb{R}^{T \times d}$ and $W_\ell^O \in \mathbb{R}^{H_q d \times D}$. In a standard Transformer, each layer materializes its own (K_ℓ, V_ℓ), so autoregressive inference caches the full set $\{(K_\ell, V_\ell)\}_{\ell=0}^{L-1}$.

Cross-layer KV sharing as memory routing. KV sharing replaces some of these independent KV tensors with KV tensors produced at other layers. We write $(K_\ell^{\mathrm{use}}, V_\ell^{\mathrm{use}})$ for the pair actually used by layer ℓ in Eq. 2. This pair may be computed locally, $(K_\ell^{\mathrm{use}}, V_\ell^{\mathrm{use}}) = (K_\ell, V_\ell)$, or derived from a

set of source layers. Thus a KV-sharing scheme specifies a cross-layer memory-routing graph: source layers produce memory states, and reuse layers consume them. This view separates three design choices that matter for both efficiency and optimization: which layers serve as sources, how many reuse layers each source serves, and how the reused KV is constructed.

Adjacent-layer sharing. CLA [4] and MLKV [56] partition consecutive layers into groups of size N. Within each group, only the first layer computes KV and the remaining layers reuse it:

$$K_\ell^{\text{use}} = K_{\lfloor \ell/N \rfloor \cdot N}, \qquad V_\ell^{\text{use}} = V_{\lfloor \ell/N \rfloor \cdot N}. \tag{3}$$

Because source layers are interleaved throughout depth, this pattern reduces the number of cached KV tensors but does not make the cache complete before the final layer group is processed.

Global-anchor sharing. Global-anchor methods place all reusable memory before a boundary layer n. For layers $\ell > n$, KV is no longer computed independently but obtained from one or more anchors. YOCO [35] uses a single anchor: $K_\ell^{\text{use}} = K_n,\ V_\ell^{\text{use}} = V_n,\ \forall\, \ell > n$. FusedKV [23] instead reconstructs deep-layer KV from two anchors through channel-wise fusion:

$$K_\ell^{\text{use}} = \mathcal{F}_\ell^K(K_0, K_n), \qquad V_\ell^{\text{use}} = \mathcal{F}_\ell^V(V_0, V_n), \qquad \forall\, \ell > n, \tag{4}$$

where $\mathcal{F}_\ell^K$ and $\mathcal{F}_\ell^V$ are per-layer fusion functions. Since all source KV tensors are available once layer n completes, global-anchor designs already have all the KV states needed by the remaining layers, thereby enabling prefill early completion.

Forward reuse induces backward connectivity. Although KV sharing is usually introduced as a cache-saving mechanism, it also changes the training dynamics. If layer ℓ uses KV derived from a source layer s, gradients from layer ℓ backpropagate through the reused tensors to the source projections W_s^K and W_s^V. Thus every memory-routing edge is also a direct cross-layer gradient path. We next analyze how these cross-layer gradient signals change the representation geometry of keys and values during training.

3.2 Representation Geometry of Keys and Values Under Cross-Layer Sharing

We use *effective rank* [31] as a compact and widely used measure of representation dimensionality. For a centered representation matrix $M \in \mathbb{R}^{N\times d}$, we compute its feature covariance $C_M = M^\top M/N$ with eigenvalues $\{\lambda_i\}_{i=1}^d$, and define

$$\operatorname{erank}(C_M) = \exp\Big(-\sum_{i=1}^{d} p_i \log p_i\Big), \qquad p_i = \frac{\lambda_i}{\sum_{j=1}^d \lambda_j}. \tag{5}$$

Applied to K_ℓ and V_ℓ, effective rank measures how evenly activation energy is distributed across feature directions, reflecting the dimensionality of the key retrieval geometry and value content space.

K/V geometry in standard attention. Suppressing head indices, let $S_\ell = Q_\ell K_\ell^\top/\sqrt{d}$, $P_\ell = \operatorname{softmax}(S_\ell)$, $O_\ell = P_\ell V_\ell$, $G_\ell = \partial\mathcal{L}/\partial S_\ell$, and $E_\ell = \partial\mathcal{L}/\partial O_\ell$. In a standard layer,

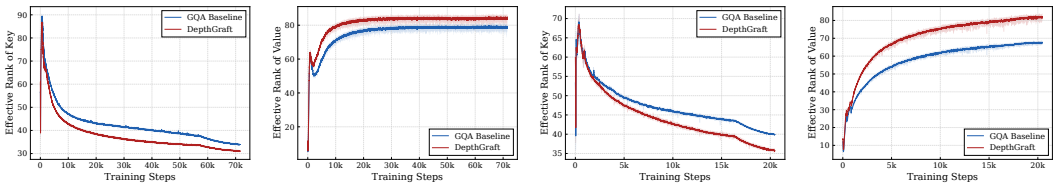
$$\nabla_{K_\ell} = \tfrac{1}{\sqrt{d}}\, G_\ell^\top Q_\ell, \qquad \nabla_{V_\ell} = P_\ell^\top E_\ell. \tag{6}$$

These two expressions expose the asymmetry between K and V. Keys are optimized through the softmax logits, where the competitive normalization makes retrieval decisions depend on relative token scores. During training, key directions that repeatedly create useful logit separations receive stronger reinforcement, while directions that do not affect the induced attention distribution are less consistently updated. This produces a natural tendency for the key spectrum to concentrate along a smaller set of discriminative directions. As shown by the baseline curves in Fig. 2, the effective rank of K_ℓ decreases during training, reflecting the formation of a sharper retrieval geometry.

Values enter attention through a different pathway. Once P_ℓ is fixed, the output is a linear readout $O_\ell = P_\ell V_\ell$. The value gradient $P_\ell^\top E_\ell$ routes downstream output gradients back to the value matrix according to the current attention pattern. As the model learns to transmit richer information through attention, this linear content channel is encouraged to use more independent directions. Correspondingly, the baseline in Fig. 2 shows that the effective rank of V_ℓ increases during training.

KV sharing amplifies K/V specialization through cross-layer gradients. Consider a source layer s whose KV is reused by a distant layer ℓ. The source KV receives both its local gradient and the reuse-layer gradient:

$$\nabla_{K_s}^{\text{share}} = \tfrac{1}{\sqrt{d}}\, G_s^\top Q_s + \tfrac{1}{\sqrt{d}}\, G_\ell^\top Q_\ell, \qquad \nabla_{V_s}^{\text{share}} = P_s^\top E_s + P_\ell^\top E_\ell. \tag{7}$$



(a) 1.5B Dense (b) 9B MoE

Figure 2: Evolution of KV representation geometry during training, visualized via effective rank (averaged over all layers).

The same source key now serves two query spaces, and the same source value supports two readout patterns. When s and ℓ are sufficiently separated, Q_s and Q_ℓ share task-relevant retrieval directions but also contain layer-specific components. For keys, shared discriminative directions receive coherent reinforcement, whereas layer-specific directions can interfere through the softmax competition; the source key is therefore biased toward directions useful across both layers, further sharpening the retrieval geometry. For values, the two readout terms do not compete through softmax. Different attention patterns instead route complementary gradients to the same value matrix, so the source value must cover a larger union of content directions. Thus cross-layer KV sharing amplifies the natural K/V specialization: keys become more compact retrieval indices, while values occupy richer content spaces, as observed in Fig. 2. In Appendix C, we provide a detailed theoretical analysis that further supports the mechanism above under explicit assumptions.

3.3 Connectivity Structure Determines Regularization Effectiveness

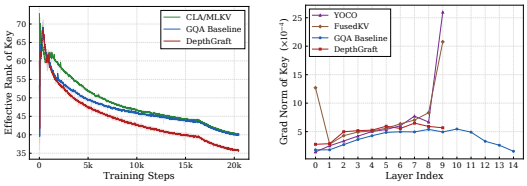
Our analysis suggests that cross-layer gradients can act as a structural regularizer for KV geometry, but the regularization effect is determined by the topology of the sharing graph. We now revisit existing KV-sharing designs and show that they expose complementary limitations.

Adjacent-layer sharing: insufficient query-space separation. In CLA [4] and MLKV [56], consecutive layers are grouped (typically groups of 2–3), and all layers within a group reuse the KV computed by the first layer. For a source layer s reused by its adjacent layer $s{+}1$, the source key receives

$$\nabla_{K_s}^{\text{adj}} = \tfrac{1}{\sqrt{d}}\, G_s^\top Q_s + \tfrac{1}{\sqrt{d}}\, G_{s+1}^\top Q_{s+1}. \tag{8}$$

This has the same form as Eq. 7, but the two query spaces are nearly adjacent in depth.

Because neighboring hidden states are highly correlated, Q_s and Q_{s+1} provide largely overlapping retrieval signals. The extra gradient path therefore adds little non-local pressure: it neither introduces enough query-space mismatch to strongly compress K toward shared discriminative directions nor provides sufficiently distinct attention patterns to expand V. Fig. 3(a) is consistent with this diagnosis: adjacent sharing does not produce the pronounced K-rank concentration predicted for useful cross-layer regularization. In this regime, the representational constraint can dominate the weak regularization benefit: multiple layers attend to identical memory, potentially reducing the model's effective depth.



(a) (b)

Figure 3: (a) Adjacent-layer sharing (CLA/MLKV) does not reduce K erank below the baseline. (b) Global-anchor sharing concentrates gradient mass on anchor layers.

Global-anchor sharing: gradient congestion and uneven regularization. In YOCO and FusedKV, a small number of early layers (one or two) serve as global anchors for all deep layers. For an anchor layer a shared by M deep layers, the gradient accumulates contributions from every reuse layer:

$$\nabla_{K_a}^{\text{global}} = \tfrac{1}{\sqrt{d}}\, G_a^\top Q_a \;+\; \sum_{j=1}^{M} \tfrac{1}{\sqrt{d}}\, \text{diag}(\alpha_j)\, G_{\ell_j}^\top Q_{\ell_j}, \tag{9}$$

where α_j are per-layer coefficients (all ones for YOCO; learnable for FusedKV). Compared with the two-way interaction in Eq. 7, the anchor must satisfy M heterogeneous query spaces at once. This

creates a high-fan-in multi-objective pressure on the same KV tensor: some cross-layer interference is useful, but concentrating all deep-layer objectives on one or two anchors can make the source representation an overloaded compromise. Fig. 3(b) shows this concentrated load directly: YOCO and FusedKV place most KV-gradient mass on their anchor layers.

Global-anchor sharing also makes the regularization structurally uneven. Only the anchor layers receive direct cross-layer KV gradients from deep layers; the remaining shallow and middle layers may still receive gradients through the residual stream, but they do not participate in the direct KV-sharing path analyzed in Sec. 3.2. Fig. 5 in the appendix illustrates this effect for FusedKV: the anchor layer exhibits elevated V effective rank, whereas nearby non-anchor layers show no clear improvement over the baseline.

3.4 DepthGraft: Hierarchical Cross-Layer KV Reconstruction

By viewing KV-sharing designs through the lens of cross-layer gradients as structural regularization, the analysis above motivates the following design principles:

1. Sufficient cross-layer distance. Source and reuse layers must be separated by enough depth that their query spaces meaningfully diverge, enabling the gradient interference that drives K/V specialization. Adjacent-layer sharing fails this condition.

2. Distributed gradient flow with full-layer participation. Each source layer should receive gradients from only a small number of reuse layers to avoid gradient congestion, and most layers should participate in the cross-layer connectivity graph so that the regularization benefit is not confined to a few anchors.

We instantiate these requirements under an efficiency constraint: all source KV tensors should be produced before the reconstructed layers, preserving prefill early completion. Based on this, we propose DepthGraft, which introduces a **hierarchical stage layout** that partitions the L-layer Transformer into shallow, middle, and deep stages, each containing roughly $L/3$ layers. The shallow and middle stages compute KV normally, while the deep stage reconstructs its KV from earlier stages. Rather than using global anchors, DepthGraft adopts **stride-aligned source assignment**, treating deep-layer memory as a locally reconstructed trajectory over depth. For each deep layer ℓ, DepthGraft selects one shallow source and one middle source at the same relative depth position:

$$s(\ell) = \ell - \tfrac{2L}{3}, \qquad m(\ell) = \ell - \tfrac{L}{3}. \tag{10}$$

This gives each deep layer two memory bases at different abstraction levels: shallow memory preserves lower-level lexical and token-local patterns, while middle-layer memory carries more abstract semantic states [2, 53, 30]. It then reconstructs KV from the corresponding shallow–middle source pair with learnable per-dimension coefficients:

$$K_\ell = a_{\ell,1} \odot K_{s(\ell)} + a_{\ell,2} \odot K_{m(\ell)}, \qquad V_\ell = b_{\ell,1} \odot V_{s(\ell)} + b_{\ell,2} \odot V_{m(\ell)}, \tag{11}$$

where $a_{\ell,i}, b_{\ell,i}$ are channel-wise coefficients for each KV group, providing higher flexibility and expressiveness than single-source reuse. For keys, the coefficients are tied across RoPE rotation pairs, so rotary position encoding and reconstruction commute, which preserves the standard KV-cache storage logic. The stride-aligned correspondence lets each deep layer receive shallow and middle memory from the same relative depth position, preserving the ordering of representation evolution across depth. Fig. 4 in the appendix provides a workflow illustration of DepthGraft.

This design naturally yields the following advantages.

(1) Inference efficiency. Because the deep stage does not materialize independent KV, DepthGraft reduces the KV cache and prefill cost by approximately $1/3$. Appendix F.1 describes the cache-saving and prefill-early-completion mechanism.

(2) Systems friendliness. The stride-aligned source mapping is simple to implement and naturally compatible with pipeline parallelism (PP) and virtual pipeline parallelism (VPP), avoiding additional cross-stage KV communication under an appropriate placement. Appendix F.3 provides the corresponding PP/VPP details.

(3) Regularized KV geometry. The design satisfies the two principles above: non-local source–target pairs provide query-space separation, and stride alignment distributes cross-layer gradients across source layers. As shown in Fig. 2 and Fig. 3(b), DepthGraft improves the geometry of KV

Table 1: Dense scaling results of baseline and DepthGraft on 0.5B–1.5B models. We report accuracy or pass rate on each benchmark and average performance over all benchmarks. The best result within each model size is shown in **bold**.

Model	Method	Knowledge & Understanding			Reasoning & Mathematics			Code			Chinese			Avg.
		MMLU	M-Pro	LAMB.	ARC-C	SIQA	GSM8K	HEval	MBPP	LCBench	C-Eval	CMMLU	C3	
0.5B	GQA	29.37	3.43	43.46	25.09	**36.08**	1.36	4.27	3.60	**1.22**	**30.81**	29.07	26.74	19.54
	DepthGraft	**30.07**	**4.86**	**43.66**	**28.18**	34.19	**1.97**	**6.71**	**5.20**	0.52	29.99	**30.35**	**27.73**	**20.29**
1B	GQA	30.35	7.86	52.58	**31.27**	35.26	**6.67**	9.15	9.40	2.09	**31.45**	30.23	31.67	23.17
	DepthGraft	**32.70**	**8.57**	**54.17**	27.84	**38.08**	6.07	**9.76**	**12.60**	**2.61**	31.43	**30.52**	**33.92**	**24.02**
1.5B	GQA	34.81	11.14	54.52	36.43	43.81	14.10	12.80	16.20	3.48	**34.64**	33.56	44.79	28.36
	DepthGraft	**38.14**	**12.57**	**58.25**	**37.11**	**46.67**	**15.01**	**14.63**	**21.40**	**3.83**	33.74	**33.81**	**54.25**	**30.78**

Table 2: MoE scaling results of baseline models and DepthGraft on 3B–16B models. We report accuracy or pass rate on each benchmark and average performance over all benchmarks. The best result within each model setting is shown in **bold**.

Model	Method	Knowledge & Understanding			Reasoning & Mathematics			Code			Chinese			Avg.
		MMLU	M-Pro	LAMB.	ARC-C	SIQA	GSM8K	HEval	MBPP	LCBench	C-Eval	CMMLU	C3	
3B A0.5B	GQA	31.44	**7.43**	51.42	26.46	35.62	6.22	12.80	**14.40**	1.22	32.29	28.19	28.78	23.02
	DepthGraft	**32.32**	6.29	**52.39**	**32.30**	**37.36**	**6.82**	**13.41**	13.00	**2.26**	**32.51**	**28.82**	**31.41**	**24.07**
9B A1.5B	GQA	44.18	**20.43**	62.36	48.11	47.54	26.99	**23.17**	**32.20**	7.65	40.00	36.69	55.12	37.04
	DepthGraft	**46.53**	19.71	**63.48**	**52.92**	**53.58**	**28.81**	22.56	29.60	**8.17**	**42.65**	**41.26**	**62.63**	**39.33**
9B A1.5B	MLA	43.30	18.29	**63.48**	46.74	47.75	27.52	22.56	**29.60**	4.00	37.85	35.02	55.95	36.01
	DepthGraft	**47.93**	**18.86**	63.33	**51.89**	**49.54**	**31.54**	**26.83**	27.40	**8.00**	**45.88**	**42.52**	**62.52**	**39.69**
16B A3B	GQA	51.72	23.86	66.43	62.54	55.89	39.50	22.56	32.40	7.22	47.60	40.68	70.74	43.43
	DepthGraft	**54.21**	**30.86**	**66.78**	**63.57**	**59.26**	**45.34**	**28.05**	**32.60**	**9.39**	**50.00**	**47.29**	**77.15**	**47.04**

representations while maintaining a healthy distributed gradient flow. The effective-rank advantage in Fig. 2 is observed across nearly all layers, rather than being confined to either source or reuse layers.

4 Experiments

We evaluate DepthGraft on both dense and sparse MoE Transformers, covering multiple model scales and attention variants. Our experiments are organized around three questions: (1) whether the empirical results support the analysis in Sec. 3.2, i.e., whether DepthGraft can use cross-layer gradients as a structural regularizer to improve downstream performance while preserving the KV-cache and prefill-efficiency benefits of KV sharing; (2) whether this effect scales to larger models and remains effective across different attention variants; and (3) whether the cross-layer connectivity and stage-layout choices in Sec. 3.4 are necessary, and how alternative sharing variants behave under the same training and evaluation protocol.

4.1 Experimental Setup

Training Settings. We conduct experiments on both dense and sparse MoE language models. For MoE models, we evaluate three scales of DeepSeekMoE-style Transformers [12] with SwiGLU feed-forward networks, ranging from 3B to 16B total parameters. For dense models, we train models from 0.5B to 1.5B parameters. The main comparisons use models trained from scratch with matched data and training recipes at each scale. Training details are provided in Appendix A, and additional experiments are provided in Appendix D.

Evaluation. We test few-shot performance on a broad downstream suite covering four capability groups: knowledge and language understanding with MMLU [15], MMLU-Pro [38], and LAMBADA [27]; reasoning and mathematics with ARC-Challenge [8], SIQA [32], and GSM8K [9]; code generation with HumanEval [7], MBPP [3], and OpenCompass LCBench [10]; and Chinese understanding with C-Eval [18], CMMLU [21], and C3 [33].

4.2 Main Results

Dense models. We first evaluate DepthGraft on dense Transformers at 0.5B, 1B, and 1.5B parameters (Table 1). DepthGraft improves the average score at all three scales, by +0.75, +0.85, and +2.42 points, respectively. The gains are not confined to a single benchmark family. At 1.5B, for example, DepthGraft improves knowledge understanding (MMLU +3.33, MMLU-Pro +1.43), reasoning

Table 3: Long-context inference efficiency of baseline models and DepthGraft. We report time-to-first-token (TTFT) and time-per-output-token (TPOT) from 8K to 128K context lengths.

Model	Method	TTFT (ms)					TPOT (ms)				
		8K	16K	32K	64K	128K	8K	16K	32K	64K	128K
9B MLA	Baseline	52.22	89.05	161.75	314.09	611.19	3.39	3.54	3.84	4.09	4.59
	DepthGraft	43.82	69.89	122.77	233.31	447.51	3.42	3.57	3.87	4.13	4.62
16B GQA	Baseline	102.26	198.39	422.94	1000.15	2655.90	5.07	5.25	5.60	6.35	7.79
	DepthGraft	84.44	151.13	313.31	726.11	1891.80	5.12	5.30	5.64	6.40	7.84

and mathematics (SIQA +2.86, GSM8K +0.91), code generation (MBPP +5.20), and Chinese comprehension (C3 +9.46). This consistent improvement on dense models indicates that constraining only the KV memory pathway does not reduce model capacity; instead, it provides a useful inductive bias for deep-layer memory representations.

MoE models. We next scale DepthGraft to MoE architectures from 3B to 16B total parameters (Table 2). Across GQA-based MoE models, DepthGraft consistently outperforms the corresponding baseline, improving the average score by +1.05, +2.29, and +3.61 points at 3B, 9B, and 16B, respectively. The 16B model shows especially strong gains, including MMLU-Pro (+7.00), CMMLU (+6.61), C3 (+6.41), and GSM8K (+5.84). Furthermore, the consistent gains on both 9B GQA (+2.29 Avg.) and 9B MLA (+3.68 Avg.) further show that DepthGraft is not tied to a particular attention implementation.

Taken together, the main results support the mechanism developed in Sec. 3.2: cross-layer KV reconstruction improves performance not merely by saving cache, but by reshaping the geometry of KV representations through direct cross-layer gradient regularization. The fact that the gains appear across dense models, sparse MoE models, different attention variants, and a wide range of scales suggests that the effect is a stable architectural bias rather than a scale-specific artifact. During training, we also observe that the cross-layer gradient regularization introduced by DepthGraft can weaken attention-sink behavior, further supporting the view that DepthGraft improves the organization of the memory pathway. Detailed visualization and analysis are provided in Appendix E.2.

Efficiency Analysis. As discussed in Sec. 3.4, DepthGraft does not materialize independent KV tensors in the deep stage, yielding a fixed KV-cache reduction of roughly one third. Table 3 further reports end-to-end TTFT and TPOT on 9B MLA and 16B GQA models with a fixed 2K generation length. DepthGraft substantially reduces TTFT, and the gain becomes larger at longer contexts where prefill is increasingly compute-bound. At 128K context length, TTFT is reduced by 26.8% on 9B MLA and 28.8% on 16B GQA. In contrast, TPOT is nearly unchanged: the overhead remains below 1% across all tested context lengths. This suggests that DepthGraft preserves decoding efficiency while substantially reducing prefill latency, and the gap to the theoretical speedup can be further reduced with specialized kernel optimizations.

4.3 Ablation Studies

We conduct ablations on the 16B A3B MoE model to examine the roles of cross-layer gradient flow, connectivity pattern, and stage layout in DepthGraft. We further compare DepthGraft with existing KV-sharing methods under the same training and evaluation protocol.

Cross-layer gradients are the key source of the gain. Variants (2)–(4) all use the same three-stage layout, allowing us to isolate the effect of the cross-layer gradient path. *Shallow-Copy* directly reuses the stride-aligned shallow source from S_1, while *Shallow-Copy (detach)* stops gradients through the reused KV. The detached variant does not degrade relative to the GQA baseline (42.86 vs. 42.38 Avg.), indicating redundancy in deep-layer KV. Once the backward path is enabled, however, the average score increases markedly to 45.30. This controlled comparison supports our central motivation: KV sharing is useful not only as a forward memory reuse mechanism, but also because cross-layer gradients regularize earlier KV representations through deep-layer training signals.

Hierarchical reconstruction is more effective than single-source copy. Variants (3)–(5) compare single-source copy with hierarchical reconstruction under active cross-layer gradients. *Shallow-Copy* in Variant (3) only connects S_1 to S_3, leaving the middle stage outside the sharing graph; the 2-stage variant (5) uses full one-to-one copy, but each reuse layer still receives only a single memory source. DepthGraft instead reconstructs each deep-layer KV from a stride-aligned S_1–S_2 source pair, improving the average score over both single-source alternatives (47.56 vs. 45.30 and 43.58). The

Table 4: Ablation study on the 16B A3B MoE model. We decompose DepthGraft along two axes: cross-layer gradient connectivity and stage layout. Best results within each group are in **bold**.

Method	Grad. Flow	Stage Layout	MMLU	M-Pro	CMMLU	C3	C-Eval	GSM8K	HEval	Avg.
(1) GQA Baseline	–	–	51.72	23.86	40.68	70.74	47.60	39.50	22.56	42.38
Gradient Connectivity & Hierarchical Reconstruction Variants										
(2) Shallow-Copy (detach)	✗	$S_1 \to S_3$	49.72	24.14	42.08	70.96	44.76	41.55	26.83	42.86
(3) Shallow-Copy	Partial	$S_1 \to S_3$	52.53	29.43	46.72	70.96	45.50	43.29	**28.66**	45.30
(4) DepthGraft	Full	$S_1 \oplus S_2 \to S_3$	**54.21**	**30.86**	**47.29**	**77.15**	**50.00**	**45.34**	28.05	**47.56**
Stage Layout Variants										
(5) DepthGraft (2-stage)	Full	$S_1 \to S_2$	52.11	22.71	**47.82**	70.25	46.56	40.64	25.00	43.58
(6) DepthGraft (4-stage, 3-src)	Full	$S_1 \oplus S_2 \oplus S_3 \to S_4$	51.65	25.71	43.96	75.40	45.36	**47.69**	**28.05**	45.40
(7) DepthGraft (4-stage, 2-src)	Full	$S_1 \oplus S_2 \to S_3, S_4$	53.26	27.14	47.43	72.05	**50.48**	43.14	23.78	45.33
(8) DepthGraft	Full	$S_1 \oplus S_2 \to S_3$	**54.21**	**30.86**	47.29	**77.15**	50.00	45.34	**28.05**	**47.56**

Table 5: Comparison with prior KV sharing methods on the 16B A3B MoE model. *Type* indicates the sharing topology. Best results are in **bold**.

Method	Type	MMLU	M-Pro	CMMLU	C3	C-Eval	GSM8K	HEval	Avg.
GQA Baseline	–	51.72	23.86	40.68	70.74	47.60	39.50	22.56	42.38
CLA / MLKV	Adjacent	48.81	20.14	40.64	58.25	40.50	37.91	20.12	38.05
YOCO	Global-1	50.28	27.57	43.87	67.73	46.59	42.91	24.39	43.33
FusedKV	Global-2	53.86	27.71	46.48	70.90	**50.66**	44.88	25.00	45.64
DepthGraft	Hierarchical	**54.21**	**30.86**	**47.29**	**77.15**	50.00	**45.34**	**28.05**	**47.56**

gains on representative benchmarks such as M-Pro, C3, C-Eval, and GSM8K suggest that the benefit comes from combining broad source participation with a more flexible hierarchical memory basis, rather than from cross-layer gradient flow alone.

The three-stage layout gives the best overall trade-off. Variants (5)–(8) compare different stage partitions under full gradient flow. The reconstructed layers take stride-aligned sources and obtain their KV by either direct copy or hierarchical reconstruction. The 2-stage variant is too restrictive, while the 4-stage variants test whether using more source stages or more reuse stages is beneficial. The results suggest that reconstruction sources are not simply "the more the better": reconstructing the last stage from three source stages underperforms the proposed two-source DepthGraft layout, and the 4-stage two-source variant also remains below the three-stage design. Thus, a three-stage partition provides a stronger operating point, balancing two complementary source abstraction levels, distributed gradient flow, and the intended final-stage cache-saving region.

DepthGraft outperforms prior KV-sharing topologies. Table 5 compares DepthGraft with existing KV-sharing designs under the same 16B A3B MoE setting. The results further support the topology analysis in Sec. 3.3. Adjacent sharing substantially underperforms the baseline, supporting the view that reusing nearby KV reduces the model's effective depth. Global-anchor sharing performs better than adjacent sharing and can improve over the baseline, but shows limited gains because the gradient regularization is concentrated on a small set of anchors. By distributing non-local source–reuse connections across layers, DepthGraft achieves the best overall performance.

4.4 Long-Context Evaluation

We further evaluate whether cross-layer KV reconstruction preserves long-context retrieval. Both the baseline and DepthGraft 16B A3B models undergo continued training at a 32K context length, with the RoPE base adjusted for length extension. Table 6 reports ten RULER [16] tasks at context lengths from 4K to 32K, covering needle-in-a-haystack retrieval, variable tracking, frequent-word extraction, and question answering.

DepthGraft achieves a higher average at every evaluated context length, including 94.8 vs. 86.8 at 4K and 82.4 vs. 76.2 at 32K. The gains extend beyond simple retrieval: at 32K, variable tracking improves from 37.8 to 76.6 and frequent-word extraction from 64.0 to 74.7. It also matches or exceeds the baseline on most reported NIAH subtasks. These results show that reconstructing deep-layer KV can retain and improve long-context capabilities without storing independent KV at every layer.

Table 6: Long-context performance on RULER after context extension. NIAH denotes needle-in-a-haystack retrieval; VT, FWE, and QA denote variable tracking, frequent-word extraction, and SQuAD-based question answering.

Length	Method	NIAH							VT	FWE	QA	Avg.
		s1	s2	s3	mk1	mk3	mv	mq				
4K	Baseline	100	100	75.0	100	99.0	90.5	95.5	43.2	80.7	84.0	86.8
	DepthGraft	100	100	100	100	99.0	98.3	98.5	81.6	84.3	86.0	**94.8**
8K	Baseline	100	100	70.0	97.0	96.0	58.3	95.3	44.0	65.0	66.0	79.2
	DepthGraft	100	100	100	94.0	100	98.0	94.5	80.0	78.7	61.0	**90.6**
16K	Baseline	100	100	82.0	96.0	93.0	48.3	91.5	45.6	66.0	63.0	78.5
	DepthGraft	100	100	96.0	92.0	96.0	87.5	86.0	77.6	78.3	63.0	**87.6**
32K	Baseline	100	100	99.0	95.0	61.0	66.3	87.8	37.8	64.0	51.0	76.2
	DepthGraft	100	100	100	94.0	76.0	73.5	76.5	76.6	74.7	53.0	**82.4**

5 Conclusion and Future Work

We presented DepthGraft, a hierarchical cross-layer KV reconstruction method that turns KV sharing from a cache-saving mechanism into a regularized memory pathway. Our analysis shows that KV reuse creates direct cross-layer gradient paths that reshape KV representation geometry, and that the effectiveness of this regularization depends on the topology of the sharing graph. Motivated by this view, DepthGraft reconstructs deep-layer KV from stride-aligned shallow–middle source pairs, preserving the efficiency benefits of KV sharing while distributing non-local gradient flow across source layers. Experiments on dense and MoE models from 0.5B to 16B show consistent downstream gains, substantial reductions in KV cache and prefill cost, and negligible overhead during decoding. Future work can develop a more rigorous theoretical understanding of how cross-layer KV sharing improves representation geometry, explore more effective sharing topologies, and further scale this direction to larger pretraining regimes.

Acknowledgments and Disclosure of Funding

This work was supported in part by the Scientific Research Innovation Capability Support Project for Young Faculty (U40) of the Ministry of Education of China (SRICSPYF-ZY2025019) and Xiaohongshu Inc.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

[2] Nura Aljaafari, Danilo S Carvalho, and André Freitas. The mechanics of conceptual interpretation in gpt models: Interpretative insights. *arXiv preprint arXiv:2408.11827*, 2024.

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

[4] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. *Advances in Neural Information Processing Systems*, 37:86927–86957, 2024.

[5] Shuochen Chang, Tong Bai, Xiaofeng Zhang, Qianli Ma, Qingyang Liu, Zhaohe Liao, Yibo Miao, and Li Niu. Unlocking the black box of latent reasoning: An interpretability-guided approach to intervention. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34019–34032, 2026.

[6] Shuochen Chang, Qingyang Liu, Shaobo Wang, Bingjie Gao, Qianli Ma, Haonan Zhao, Yibo Miao, Yulin Sun, Zelin Peng, Jiangtong Li, et al. Why struggle with continuous latents? interpretable discrete latent reasoning via rendered compression. *arXiv preprint arXiv:2606.29712*, 2026.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[8] 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.

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

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

[11] Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? *arXiv preprint arXiv:2505.13898*, 2025.

[12] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, 2024.

[13] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

[14] 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*, pages 770–778, 2016.

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[16] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[17] 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*, pages 4700–4708, 2017.

[18] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010, 2023.

[19] Suzana Ilic, Edison Marrese-Taylor, Jorge Balazs, and Yutaka Matsuo. Deep contextualized word representations for detecting sarcasm and irony. In *Proceedings of the 9th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 2–7, 2018.

[20] Jonas Knupp, Jan Hendrik Metzen, Jeremias Bohn, Georg Groh, and Kristian Kersting. Depth-recurrent attention mixtures: Giving latent reasoning the attention it deserves. *arXiv preprint arXiv:2601.21582*, 2026.

[21] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285, 2024.

[22] Ning Liao, Xiaoxing Wang, Zehao Lin, Weiyang Guo, Feng Hong, Shixiang Song, Geng Yu, Zihua Zhao, Sitao Xie, Longxuan Wei, et al. Innovator: Scientific continued pretraining with fine-grained moe upcycling. *arXiv preprint arXiv:2507.18671*, 2025.

[23] Hongzhan Lin, Zhiqi Bai, Xinmiao Zhang, Sen Yang, Xiang Li, Siran Yang, Yunlong Xu, Jiaheng Liu, Yongchi Zhao, Jiamang Wang, et al. Reconstructing kv caches with cross-layer fusion for enhanced transformers. *arXiv preprint arXiv:2512.03870*, 2025.

[24] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[25] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, 2025.

[26] Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, 2024.

[27] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1525–1534, 2016.

[28] Xiaohan Qin, Victor Wang, Ning Liao, Cancheng Zhang, Xiangdong Zhang, Mingquan Feng, Jingzhi Wang, and Junchi Yan. sstoken: Self-modulated and semantic-aware token selection for llm fine-tuning. In *International Conference on Learning Representations*, volume 2026, pages 79347–79368, 2026.

[29] Xiaohan Qin, Xiangdong Zhang, Tuo Dai, Sunan Zou, Huaijin Wu, Yebin Yang, Zhuo Xia, Lin Yao, Yuliang Liu, and Junchi Yan. Multi-stream attention residuals. *arXiv preprint arXiv:2610.13630*, 2026.

[30] Elisabetta Rocchetti and Alfio Ferrara. How instruction-tuning imparts length control: A cross-lingual mechanistic analysis. *arXiv preprint arXiv:2509.02075*, 2025.

[31] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

[32] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, 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 (EMNLP-IJCNLP)*, pages 4463–4473, 2019.

[33] Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8: 141–155, 2020.

[34] Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

[35] Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361, 2024.

[36] Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[37] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

[38] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[39] Jianyu Wei, Yizhao Gao, Qihao Zhang, Shimao Chen, Zhengju Tang, Yu Cheng, Shengjie Zhou, Zihan Jiang, Yifan Song, Hailin Zhang, et al. HySparse2: Hybrid sparse attention with two-level KV sharing. *arXiv preprint arXiv:2609.26368*, 2026.

[40] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024. URL https://arxiv.org/abs/2410.05192.

[41] Haoyi Wu and Kewei Tu. Layer-condensed kv cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11175–11188, 2024.

[42] Huaijin Wu, Lianqiang Li, Hantao Huang, Yi Tu, Jihang Zhang, Minghui Yu, and Junchi Yan. HShare: Fast LLM decoding by hierarchical key-value sharing. In *International Conference on Learning Representations*, volume 2025, pages 89412–89428, 2025.

[43] Huaijin Wu, Bingrui Li, Yebin Yang, Yi Tu, Zhanpeng Zhou, Jianfei Chen, and Junchi Yan. Achieving low-bit Muon through subspace preservation and grid quantization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=g2l9bg9DWx.

[44] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[45] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

[46] Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

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

[48] Yebin Yang, Huaijin Wu, Fu Guo, Lin Yao, Xiaohan Qin, Jingzhi Wang, Debing Zhang, and Junchi Yan. Jtok: On token embedding as another axis of scaling law via joint token self-modulation. *arXiv preprint arXiv:2602.00800*, 2026.

[49] Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing. *arXiv preprint arXiv:2410.18517*, 2024.

[50] Xiangdong Zhang, Xiaohan Qin, Sunan Zou, Tuo Dai, Xiaoming Shi, Huaijin Wu, Yebin Yang, Zhuo Xia, Shaofeng Zhang, Lin Yao, et al. xhc: Expanded hyper-connections. *arXiv preprint arXiv:2607.14530*, 2026.

[51] Xiangdong Zhang, Debing Zhang, Shaofeng Zhang, Xiaohan Qin, Yu Cheng, and Junchi Yan. Nitp: Next implicit token prediction for llm pre-training. *arXiv preprint arXiv:2605.24956*, 2026.

[52] Yilang Zhang et al. Ancre: Adaptive neural connection reassignment for efficient depth scaling. *arXiv preprint arXiv:2602.09009*, 2026.

[53] Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Attention heads of large language models: A survey. *arXiv preprint arXiv:2409.03752*, 2024.

[54] Zhanchao Zhou et al. Value residual learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 28341–28356, 2025.

[55] Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, et al. Mixture-of-depths attention. *arXiv preprint arXiv:2603.15619*, 2026.

[56] Zayd Muhammad Kawakibi Zuhri, Muhammad Farid Adilazuarda, Ayu Purwarianti, and Alham Fikri Aji. Mlkv: Multi-layer key-value heads for memory efficient transformer decoding. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5516–5525, 2025.

A Experimental Details

Within the broader landscape of LLM pretraining [51, 48, 22, 43] and post-training adaptation and inference [28, 6, 5, 42], our experiments focus on architectural improvements during pretraining.

Model configurations. Tables 7 and 8 summarize the model configurations and training hyperparameters used in our dense and MoE experiments. All models are decoder-only Transformers with SwiGLU feed-forward layers, RMSNorm, and RoPE positional embeddings. The dense models use GQA throughout, while the MoE experiments cover both GQA and MLA variants. All MoE backbones follow a DeepSeekMoE-style layout [12]: the first Transformer layer is dense and the remaining layers are MoE layers.

Table 7: Hyperparameters and model configurations for dense-model pretraining.

Group	Parameter	0.5B	1B	1.5B
Training	lr schedule	WSD	WSD	WSD
	lr	8.0×10^{-4}	6.0×10^{-4}	6.0×10^{-4}
	warmup steps	2,000	2,000	2,000
	decay ratio	0.2	0.2	0.2
	AdamW-β	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
	weight decay	0.1	0.1	0.1
	grad clip	1.0	1.0	1.0
	global batch size	512	512	–
Model	hidden dim.	1024	1024	1024
	# layers	9	18	28
	head dim.	128	128	128
	# query heads	8	8	8
	# KV heads	4	4	4
	context length	8192	8192	8192
	dense FFN size	10944	10944	10944
	DepthGraft reuse layers	3/9	6/18	9/28
	Normalization	RMSNorm ($\epsilon = 10^{-5}$)		
	Position Embedding	RoPE ($\theta = 50000$)		
Data	tokenizer	qwen2	qwen2	qwen2
	# tokens	–	–	–

Training hyperparameters. All comparisons at the same model scale use identical training recipes and token budgets, so that differences can be attributed to the KV-sharing design rather than optimization settings. For the prior KV-sharing baselines in Table 5, CLA/MLKV use three-layer sharing groups, while YOCO and FusedKV start reusing KV after the middle stage, i.e., after the first $2L/3$ layers, as illustrated in Fig. 3(b). This matches the reuse ratio of DepthGraft for YOCO and FusedKV, allowing us to compare performance under different connectivity patterns. For our 28-layer models, we leave layer 0 unshared and apply the three-stage reconstruction to the remaining 27 layers, resulting in three 9-layer stages. We use AdamW with β_1=0.9, β_2=0.95, weight decay 0.1, and gradient clipping at 1.0. Models smaller than 16B use a warmup–stable–decay schedule [40]: the final 20% of training tokens use exponential decay to reduce the learning rate to 0.1 of its peak value. For the 16B MoE model, we use a warmup–stable schedule without LR decay and with batch-size ramp-up. All models use the qwen2 tokenizer with a vocabulary size of 152,064 and are trained with a context length of 8192. Our pretraining corpus is primarily proprietary and private, built from web text, code, math, and multilingual sources, with exact/fuzzy deduplication, rule-based filtering, and LLM-based quality filtering.

Evaluation protocol. We evaluate all models with the same few-shot protocol under OpenCompass. The benchmark suite covers knowledge and language understanding with MMLU (5-shot), MMLU-Pro (5-shot), and LAMBADA (0-shot); reasoning and mathematics with ARC-Challenge (25-shot), SIQA (5-shot), and GSM8K (4-shot); code generation with HumanEval (0-shot), MBPP (3-shot), and OpenCompass LCBench2023 (5-shot); and Chinese understanding with C-Eval (5-shot), CMMLU (5-shot), and C3 (3-shot).

Table 8: Hyperparameters and model configurations for MoE-model pretraining.

Group	Parameter	3B A0.5B GQA	9B A1.5B GQA	9B A1.5B MLA	16B A3B GQA
Training	lr schedule	WSD	WSD	WSD	WS
	lr	4.82×10^{-4}	4.82×10^{-4}	4.82×10^{-4}	4.2×10^{-4}
	warmup steps	2,000	2,000	2,000	750
	decay ratio	0.2	0.2	0.2	–
	AdamW-β	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
	weight decay	0.1	0.1	0.1	0.1
	grad clip	1.0	1.0	1.0	1.0
	global batch size	512	–	–	–
Model	hidden dim.	768	2080	2080	2048
	# layers	15	15	15	28
	# dense layers	1	1	1	1
	# MoE layers	14	14	14	27
	head dim.	128	128	–	128
	# query heads	8	16	16	16
	# KV heads	4	8	–	8
	MLA q-lora rank	–	–	1536	–
	MLA kv-lora rank	–	–	384	–
	MLA qk head dim.	–	–	80+48	–
	MLA v head dim.	–	–	128	–
	context length	8192	8192	8192	8192
	dense FFN size	10944	6144	6144	10944
	# routed experts	144	144	144	64
	topK route	8	8	8	6
	MoE FFN size	544	704	704	1408
	shared exp. size	544	704	704	2816
	DepthGraft reuse layers	5/15	5/15	5/15	9/28
	Normalization	RMSNorm ($\epsilon = 10^{-5}$)			
	Position Embedding	RoPE ($\theta = 50000$)			
Data	tokenizer	qwen2	qwen2	qwen2	qwen2
	# tokens	–	–	–	–

B Method Details and Additional Illustrations

RoPE-pair tying for K coefficients. Rotary position embeddings apply a block-diagonal rotation to each two-dimensional channel pair of the query and key vectors. For the NeoX-style layout used in our implementation, dimensions j and $j + d/2$ form one RoPE pair. If a channel-wise coefficient assigns different values to the two dimensions in the same pair, then applying RoPE before and after reconstruction would no longer be equivalent. Consistent with FusedKV [23], we therefore tie the K reconstruction coefficients within each RoPE pair. Concretely, for each source coefficient $a_{\ell,i}$, we store a half-sized tensor $a_{\ell,i} \in \mathbb{R}^{H_k \times d/2}$ and expand it to $\tilde{a}_{\ell,i} \in \mathbb{R}^{H_k \times d}$ by

$$\tilde{a}_{\ell,i}[h, j] = \tilde{a}_{\ell,i}[h, j + d/2] = a_{\ell,i}[h, j], \qquad j = 1, \ldots, d/2. \tag{12}$$

Let R_p denote the RoPE rotation matrix at position p. Because each expanded coefficient is constant within every rotation pair, it commutes with the RoPE rotation:

$$R_p(\tilde{a}_{\ell,i} \odot K) = \tilde{a}_{\ell,i} \odot R_p(K). \tag{13}$$

Therefore, for the reconstructed key,

$$R_p\big(\tilde{a}_{\ell,1} \odot K_{s(\ell)} + \tilde{a}_{\ell,2} \odot K_{m(\ell)}\big) = \tilde{a}_{\ell,1} \odot R_p(K_{s(\ell)}) + \tilde{a}_{\ell,2} \odot R_p(K_{m(\ell)}). \tag{14}$$

This means that key reconstruction can be performed either before or after RoPE without changing the result, preserving the standard KV-cache storage logic. Value vectors do not use RoPE, so the V coefficients $b_{\ell,i}$ are stored at full channel dimension and do not require pair tying.

Coefficient initialization. All reconstruction coefficients $a_{\ell,1}, a_{\ell,2}, b_{\ell,1}, b_{\ell,2}$ are initialized to 0.5. At the start of training, each deep layer therefore receives an equal-weight reconstruction from its

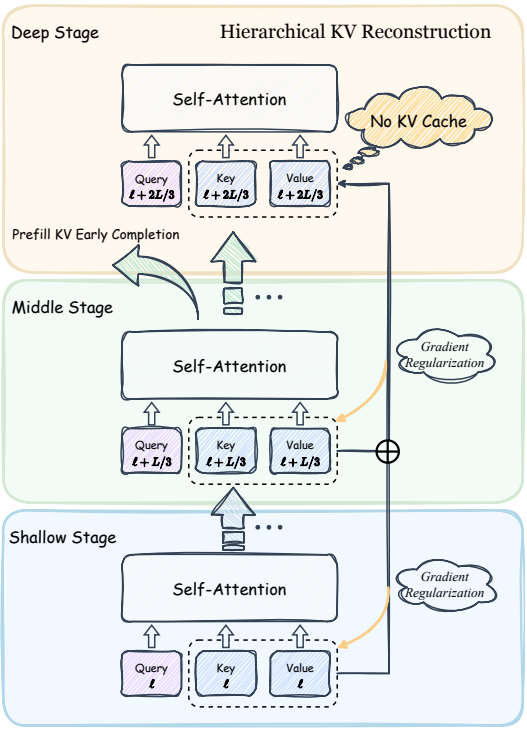


Figure 4: DepthGraft workflow. The shallow and middle stages compute and cache KV normally, while deep-stage KV is reconstructed from stride-aligned shallow–middle source pairs. The deep stage therefore does not materialize independent KV cache, while gradients from deep-layer attention flow back to the source KV projections through the reconstruction paths.

shallow and middle sources. This provides a stable interpolation between the two memory bases before the model learns layer- and channel-specific preferences. As shown in Appendix E.4, the coefficients later deviate from this symmetric initialization and learn different source preferences for K and V.

C Theoretical Analysis

A conditional spectral view of cross-layer KV regularization. We provide a local spectral analysis to support the effective-rank diagnostic in Eq. 5. For a centered key or value activation matrix M, denote its feature covariance by $C_M = M^\top M/N$. The effective rank is high when the covariance spectrum is close to uniform, and low when activation energy is concentrated in a few feature directions. Our goal is not to prove that cross-layer KV sharing always changes effective rank in a fixed direction. Instead, we show that the observed K/V effective-rank trends follow under a natural spectral-alignment condition on the cross-layer update covariance.

Lemma 1 (spectral injection and effective rank). Consider a local update of the activation matrix

$$M^+ = M + \eta\Delta, \tag{15}$$

where Δ is the induced activation-space update and η is the learning rate. The covariance update admits the exact decomposition

$$\begin{aligned} C_{M^+} &= \frac{1}{N}(M+\eta\Delta)^\top(M+\eta\Delta) \\ &= C_M + \eta A_{M,\Delta} + \eta^2\Sigma_\Delta, \end{aligned} \tag{16}$$

where

$$A_{M,\Delta} = \frac{1}{N}(M^\top\Delta + \Delta^\top M), \qquad \Sigma_\Delta = \frac{1}{N}\Delta^\top\Delta. \tag{17}$$

We analyze the directional trend over a short training window, where stochastic mini-batch fluctuations make the first-order alignment term $A_{M,\Delta}$ average to a small or low-rank correction that does not reverse the spectral ordering induced by Σ_Δ. Equivalently, for the purpose of identifying which feature directions receive persistent update energy, we approximate

$$\mathbb{E}[C_{M^+}] \approx C_M + \eta^2 \mathbb{E}[\Sigma_\Delta]. \tag{18}$$

Let $C_M = U \operatorname{diag}(\lambda_1, \ldots, \lambda_d) U^\top$ with $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$. Assume that the update covariance is approximately aligned with the current representation geometry:

$$U^\top \mathbb{E}[\Sigma_\Delta] U = \operatorname{diag}(\mu_1, \ldots, \mu_d) + R, \qquad \|R\|_2 \ll \|\operatorname{diag}(\mu)\|_2. \tag{19}$$

Ignoring the small off-diagonal residual R, the eigenvalues evolve as

$$\lambda_i^+ = \lambda_i + \eta^2 \mu_i = \lambda_i (1 + \eta^2 \rho_i), \qquad \rho_i = \mu_i / \lambda_i. \tag{20}$$

After normalization, the effective-rank weights become

$$p_i^+ = \frac{\lambda_i^+}{\sum_j \lambda_j^+} = \frac{\lambda_i (1 + \eta^2 \rho_i)}{\sum_j \lambda_j (1 + \eta^2 \rho_j)}. \tag{21}$$

If the relative injection ρ_i is larger on the leading eigenspace than on the trailing eigenspace, then probability mass shifts toward already-dominant components, so p^+ majorizes p. Since Shannon entropy is Schur-concave, $H(p^+) < H(p)$ whenever the shift is nonzero, and therefore

$$\operatorname{erank}(C_{M^+}) < \operatorname{erank}(C_M). \tag{22}$$

Conversely, if ρ_i is larger on trailing or previously underused directions, the normalized spectrum becomes more uniform and

$$\operatorname{erank}(C_{M^+}) > \operatorname{erank}(C_M). \tag{23}$$

This establishes that effective-rank changes are controlled by *relative spectral injection*, not merely by the absolute magnitude of the update covariance.

Cross-layer KV sharing. We now apply the lemma to a source layer s whose KV activations are reused by a non-adjacent layer ℓ. Suppressing head indices and reconstruction coefficients for clarity, the source KV gradients take the form

$$\nabla_{K_s}^{\text{share}} = \frac{1}{\sqrt{d}} \left(G_s^\top Q_s + G_\ell^\top Q_\ell \right), \qquad \nabla_{V_s}^{\text{share}} = P_s^\top E_s + P_\ell^\top E_\ell, \tag{24}$$

where

$$G_t = \frac{\partial \mathcal{L}}{\partial (Q_t K_s^\top / \sqrt{d})}, \qquad E_t = \frac{\partial \mathcal{L}}{\partial (P_t V_s)}, \qquad t \in \{s, \ell\}. \tag{25}$$

Under a local linearization of the projection update, we assume that the induced activation update covariance follows the covariance of these activation gradients. The key-side covariance contains local and cross-layer terms:

$$\begin{aligned} \Sigma_K^{\text{share}} \propto \mathbb{E}\Big[& Q_s^\top G_s G_s^\top Q_s + Q_\ell^\top G_\ell G_\ell^\top Q_\ell \\ & + Q_s^\top G_s G_\ell^\top Q_\ell + Q_\ell^\top G_\ell G_s^\top Q_s \Big]. \end{aligned} \tag{26}$$

The value-side covariance similarly expands as

$$\begin{aligned} \Sigma_V^{\text{share}} \propto \mathbb{E}\Big[& E_s^\top P_s P_s^\top E_s + E_\ell^\top P_\ell P_\ell^\top E_\ell \\ & + E_s^\top P_s P_\ell^\top E_\ell + E_\ell^\top P_\ell P_s^\top E_s \Big]. \end{aligned} \tag{27}$$

Proposition 1 (asymmetric effective-rank pressure from cross-layer KV sharing). Suppose that (i) cross-layer key gradients share a low-dimensional task-relevant retrieval subspace $\mathcal{U}_K$ aligned with the leading eigenspace of C_{K_s}, and their cross-covariance is coherent inside $\mathcal{U}_K$ but weak outside it; and (ii) value readout gradients from the source and reuse layers contain complementary content directions with non-negligible projection onto trailing or underused eigendirections of C_{V_s}. Then, under the spectral approximation in Eqs. 18–19, cross-layer KV sharing locally induces

$$\operatorname{erank}(C_{K_s^+}) < \operatorname{erank}(C_{K_s}), \qquad \operatorname{erank}(C_{V_s^+}) > \operatorname{erank}(C_{V_s}). \tag{28}$$

Proof. For keys, Eq. 26 shows that cross-layer sharing adds covariance terms coupling $G_s^\top Q_s$ and $G_\ell^\top Q_\ell$. By assumption, these terms reinforce one another inside $\mathcal{U}_K$, producing a positive spectral increment on directions shared by the two query spaces. Outside $\mathcal{U}_K$, the cross-layer components are weakly correlated, so their contribution to trailing eigenvalues is small relative to the coherent leading injection. Therefore the relative increments satisfy

$$\frac{\mu_i^K}{\lambda_i^K} > \frac{\mu_j^K}{\lambda_j^K}, \qquad i \in \mathcal{I}_{\text{lead}}(K_s),\ j \in \mathcal{I}_{\text{trail}}(K_s), \tag{29}$$

where μ_i^K denotes the key update-covariance eigenvalue in the approximate eigenbasis of C_{K_s}. By Lemma 1, the normalized key covariance spectrum becomes more concentrated, yielding

$$\text{erank}(C_{K_s^+}) < \text{erank}(C_{K_s}). \tag{30}$$

For values, Eq. 27 shows that the reuse layer routes an additional readout gradient $P_\ell^\top E_\ell$ into the same source value matrix. Unlike keys, values are read out linearly once the attention pattern is fixed; distinct attention patterns can therefore provide complementary content gradients rather than competing through the softmax logits. By assumption, these additional content gradients have non-negligible projection onto underused value directions, so

$$\frac{\mu_i^V}{\lambda_i^V} < \frac{\mu_j^V}{\lambda_j^V}, \qquad i \in \mathcal{I}_{\text{lead}}(V_s),\ j \in \mathcal{I}_{\text{trail}}(V_s). \tag{31}$$

Lemma 1 then implies that the normalized value covariance spectrum becomes more uniform:

$$\text{erank}(C_{V_s^+}) > \text{erank}(C_{V_s}). \tag{32}$$

Combining the two parts proves Eq. 28. □

Interpretation and scope. The proposition is conditional: it does not claim that effective rank must change monotonically under arbitrary training dynamics. It states that the K/V effective-rank trends follow when the induced update covariance is organized as suggested by the attention mechanism: keys receive coherent reinforcement in shared retrieval directions, while values receive complementary content gradients. The topology of the sharing graph affects whether these assumptions are plausible. Adjacent sharing may make $Q_s \approx Q_\ell$ and $P_s \approx P_\ell$, yielding mostly redundant gradients. Global-anchor sharing can overload a few sources with many heterogeneous objectives. Stride-aligned non-local sharing is more likely to satisfy the assumptions because it provides sufficient cross-layer distance while distributing the gradient load across source layers.

D Additional Experimental Results

D.1 Retention of Gains after Supervised Fine-tuning

We apply supervised fine-tuning (SFT) to the baseline and DepthGraft 16B A3B pretrained models using the same data, learning-rate schedule, and update counts, and evaluate both with the same downstream protocol. Table 9 shows that the pretraining advantage persists after SFT: DepthGraft achieves an average score of 52.00, compared with 48.52 for the baseline. It improves on 11 benchmarks and matches the baseline on LCBench, with notable gains on MBPP (+7.00), HumanEval (+5.49), ARC-C (+4.81), and GSM8K (+2.96). The gains from KV reconstruction therefore carry through supervised adaptation rather than being erased by post-training.

Table 9: Downstream performance after SFT on the 16B A3B models. Both models use the same fine-tuning recipe.

Method	MMLU	M-Pro	LAMB.	ARC-C	SIQA	GSM8K	HEval	MBPP	LCBench	C-Eval	CMMLU	C3	Avg.
Baseline + SFT	53.72	35.00	60.77	67.01	59.83	52.08	37.80	35.80	**14.96**	49.46	43.47	72.33	48.52
DepthGraft + SFT	**56.84**	**37.57**	**62.28**	**71.82**	**62.54**	**55.04**	**43.29**	**42.80**	**14.96**	**52.03**	**47.28**	**77.59**	**52.00**

D.2 Compatibility with Sliding-Window Attention

DepthGraft changes the source of deep-layer KV, while sliding-window attention (SWA) restricts the positions that each query can attend to. We test their combination on a 12-layer MoE model with 8B total and approximately 1B activated parameters, using a 3:1 ratio of SWA to full-attention layers and a window size of 512. KV reuse is restricted to layers of the same attention type: SWA layers reuse SWA sources, and full-attention layers reuse full-attention sources. This avoids mixing the two types, whose KV-gradient norms differ in our experiments, while preserving prefill early completion because all sources remain in the first two stages.

Table 10: Compatibility with SWA on an 8B MoE model. The two variants share the same attention-mask pattern and training recipe.

Method	MMLU	M-Pro	LAMB.	ARC-C	SIQA	GSM8K	HEval	MBPP	LCBench	C-Eval	CMMLU	C3	Avg.
SWA baseline	37.05	11.71	60.26	21.62	**43.67**	16.38	**21.34**	22.20	**3.13**	**34.34**	34.32	31.45	28.12
SWA + DepthGraft	**38.25**	**13.57**	**61.43**	**33.33**	40.31	**19.50**	18.90	**23.00**	2.57	33.57	**36.54**	**35.63**	**29.72**

As shown in Table 10, DepthGraft improves the average score from 28.12 to 29.72, with gains on ARC-C, GSM8K, and C3 among other tasks. These results support combining cross-layer KV reconstruction with local attention, extending its applicability beyond full-attention backbones. An alternative extension is a causal encoder–decoder design inspired by DeepSeek-V4.1-Flash [46], applying KV reconstruction only to full-attention layers while allowing SWA layers to retain their own KV. This gives local representations greater freedom to evolve across layers and could preserve the benefits of prefill early completion through SWA bounded replay over a short suffix of the prompt.

E Additional Training Dynamics and Representation Analysis

E.1 Effective Rank Dynamics: Per-Layer Analysis

Sec. 3.3 argues that global-anchor sharing, such as FusedKV, induces structurally uneven regularization: only the designated anchor layers participate in cross-layer gradient flow, while nearby non-anchor layers receive no direct gradient benefit from KV reuse. We provide per-layer empirical evidence for this effect by tracking the value effective rank $\mathrm{erank}(V_\ell)$, defined in Eq. 5, at individual layers throughout training.

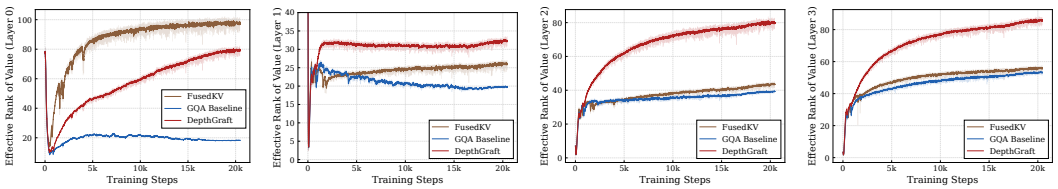
Figure 5 shows V effective-rank trajectories for layers 0–3 under the 9B GQA baseline, FusedKV, and DepthGraft. For FusedKV, only layer 0, the bottom anchor, exhibits a clear increase above the baseline. Layers 1–3 do not serve as anchors and therefore receive no direct cross-layer gradient through reused KV tensors; correspondingly, their V effective ranks show no clear improvement over the baseline. This supports the topology analysis in Sec. 3.3: global-anchor sharing confines the representation benefit to a small set of source layers.

In contrast, DepthGraft elevates V effective rank across the shown source layers. Because its stride-aligned source assignment distributes reuse edges across the shallow and middle stages, the gradient-mediated regularization described in Sec. 3.2 is spread over many source layers rather than concentrated on a few anchors. The per-layer effective-rank measurements therefore provide additional evidence that distributed cross-layer connectivity is important for turning KV sharing into an effective representation regularizer.

E.2 Attention Sink Analysis

A common pathology in deep Transformers is the *attention sink* phenomenon, where a disproportionate amount of attention mass is assigned to the first token regardless of its semantic relevance [44]. In our setting, this behavior provides an additional diagnostic of key geometry: if a layer develops extreme key directions, the softmax distribution can become dominated by a small number of sink positions.

DepthGraft mitigates this tendency through cross-layer gradient regularization. Each shallow or middle source serves its own attention layer while also being reused by a stride-aligned deep layer, so its KV projections receive optimization signals from multiple query spaces. This structure can regularize the source key geometry and make it less likely for a layer to develop sink-prone key

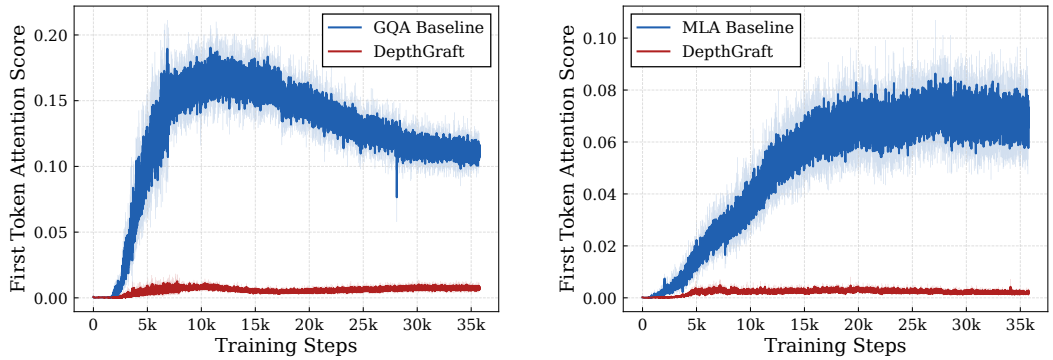


(a) Layer 0 (b) Layer 1 (c) Layer 2 (d) Layer 3

Figure 5: Per-layer V effective rank during training for the 9B GQA baseline, FusedKV, and DepthGraft. FusedKV improves the anchor layer but leaves nearby non-anchor layers close to the baseline, whereas DepthGraft yields consistent rank elevation across the shown source layers due to distributed stride-aligned connectivity.

directions that dominate the softmax. This is consistent with the analysis in Sec. 3.2, where useful cross-layer KV sharing makes keys form a more compact retrieval geometry.

We verify this empirically by tracking the average attention score assigned to the first token throughout training. Figure 6 reports this metric for both 9B GQA and 9B MLA models. In both cases, DepthGraft consistently lowers first-token attention concentration relative to the baseline, indicating reduced attention-sink behavior.



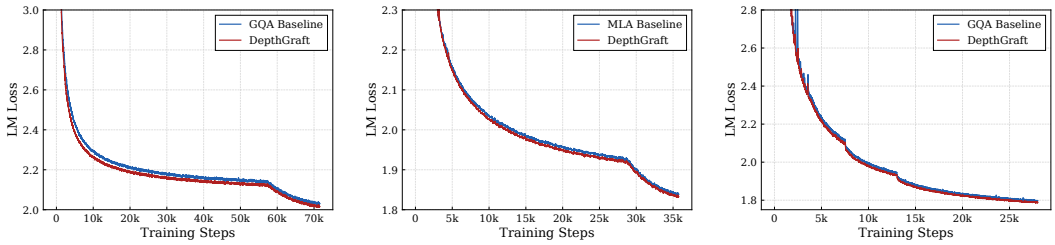
(a) 9B GQA model. (b) 9B MLA model.

Figure 6: Average attention score assigned to the first token over training steps. DepthGraft consistently reduces first-token attention concentration compared with the baseline on both GQA and MLA models.

E.3 Training Loss Curves

Figure 7 compares the training loss curves of DepthGraft and the corresponding baselines across representative dense and MoE settings. DepthGraft consistently achieves lower training loss across model scales and attention variants, indicating that the proposed cross-layer KV reconstruction does not introduce optimization instability. The magnitude of the loss improvement, however, is moderate: the relative loss reduction is roughly 0.5%.

This behavior is consistent with the role of DepthGraft as a structural regularizer. Unlike cross-layer designs whose primary effect is to substantially alter the residual computation pathway, DepthGraft constrains only the KV memory pathway and even removes independent deep-layer KV projections. Consequently, its training-loss advantage is not as large as that reported for some broader cross-layer connectivity mechanisms, but it yields much larger downstream gains than the loss difference alone would suggest. This supports our interpretation that DepthGraft improves the geometry and generalization of KV representations, rather than merely optimizing the pretraining loss more aggressively.



(a) 1.5B Dense. (b) 9B MoE MLA. (c) 16B MoE GQA.

Figure 7: Training loss curves across representative model scales and attention variants. DepthGraft consistently improves training loss, but the gain is modest relative to its downstream benchmark improvements, supporting its interpretation as a KV-geometry regularizer.

E.4 Reconstruction Coefficient Dynamics

We further inspect the learned channel-wise reconstruction coefficients. For each reused layer, DepthGraft reconstructs deep-layer KV from a shallow source (src1) and a middle source (src2). Table 11 reports the final coefficient norm normalized by its initialization norm; values larger than 1 indicate that the corresponding source receives increased weight during training.

Table 11: Final reconstruction-coefficient norm relative to initialization. Src1 denotes the shallow source and src2 denotes the middle source. Values larger than 1 indicate increased reliance on the corresponding source after training.

Layer	K src1	K src2	V src1	V src2
10	0.6905	1.1955	2.1655	0.7594
11	0.4738	1.1936	1.6323	1.4713
12	0.5894	1.2603	2.4572	1.2290
13	0.7178	1.0534	2.6295	0.8892
14	0.7208	0.9557	1.8670	0.6671

The learned coefficients reveal a consistent K/V asymmetry. For keys, the middle source is generally upweighted relative to the shallow source, suggesting that deep-layer retrieval benefits more from the more abstract middle-stage key geometry. For values, the shallow source is more strongly upweighted, indicating that lower-level content features remain useful when reconstructing the deep-layer memory content. This qualitative pattern is consistent with the observation behind FusedKV. However, the comparison in Table 5 shows that FusedKV still underperforms DepthGraft under the same 16B A3B MoE setting. This indicates that the benefit does not come only from two-source reconstruction; distributing the source–reuse connections across layers, and thereby spreading the cross-layer gradient regularization over many participating source layers, is also important.

F Efficiency and System Analysis

F.1 KV Cache and Prefill Cost

DepthGraft partitions the network into shallow, middle, and deep stages of approximately equal depth. The shallow and middle stages materialize KV normally, while the deep stage reconstructs its KV from stride-aligned shallow–middle sources. Therefore, no independent KV cache needs to be stored for the deep stage. For an L-layer model with identical KV dimensionality across layers, the baseline stores KV for all L layers, whereas DepthGraft stores KV only for roughly $2L/3$ source layers:

$$\frac{\mathrm{KVCache}_{DepthGraft}}{\mathrm{KVCache}_{\mathrm{baseline}}} \approx \frac{2L/3}{L} = \frac{2}{3}. \tag{33}$$

This gives an approximately one-third reduction in KV-cache memory during inference.

The same structure also enables prefill early completion. In standard autoregressive inference, the prefill phase processes the full prompt through all layers to produce the KV cache and the hidden

state needed to generate the first response token. In DepthGraft, the purpose of prefill with respect to KV construction is completed once the shallow and middle stages have processed the full prompt, because all KV tensors needed by future decoding steps have already been materialized. The deep stage does not need to process the entire prompt to create additional KV cache. Instead, after the shallow and middle KV caches are available, we only forward the final prompt token through the deep stage, using the stored shallow–middle KV caches to compute attention and produce the first response token.

Thus, the prefill saving comes from avoiding full-sequence computation in the deep stage, while retaining the necessary computation for the last prompt token that initializes decoding. As context length and model size increase, the full-prompt prefill computation becomes increasingly dominant, and the cost of forwarding only the final token through the deep stage becomes a smaller fraction of total prefill cost. This explains the trend in Table 3: the measured TTFT reduction grows at longer context lengths and approaches the theoretical one-third saving.

F.2 Training FLOPs and Parameter Count

DepthGraft does not introduce additional training FLOPs in the way that many explicit cross-layer connectivity mechanisms do. The shallow and middle stages are computed normally, and the deep-stage KV reconstruction only requires element-wise linear combinations of two cached source tensors. This mixing cost is negligible compared with the matrix multiplications in attention and FFN layers. Moreover, because the deep stage does not require independent K/V projections, an implementation can skip the corresponding K/V projection computation in reused layers, slightly reducing training FLOPs. In our large-scale runs, the observed training throughput is essentially identical to the baseline.

We also reduce the activated parameter count by removing the K and V projection weights from reused layers. For a GQA layer with hidden dimension d_{model}, n_{kv} KV heads, and head dimension d_h, the K/V projection parameters per layer are

$$P_{\text{KV}}^{\text{layer}} = 2d_{\text{model}}n_{\text{kv}}d_h, \tag{34}$$

where the factor of 2 accounts for W^K and W^V. If DepthGraft reuses KV in L_{reuse} layers, the removed activated parameters are

$$\Delta P_{\text{active}} = L_{\text{reuse}} \cdot 2d_{\text{model}}n_{\text{kv}}d_h. \tag{35}$$

For the 16B A3B GQA MoE model, d_{model}=2048, n_{kv}=8, d_h=128, and L_{reuse}=9. This gives

$$\Delta P_{\text{active}} = 9 \times 2 \times 2048 \times 8 \times 128 = 37.7\text{M}. \tag{36}$$

Relative to the 3B activated-parameter budget, this corresponds to approximately 1.3% fewer activated parameters. Thus, DepthGraft obtains its downstream gains while slightly reducing, rather than increasing, the active model capacity.

F.3 Parallel Training Compatibility

Pipeline parallelism (PP) is essential for training large Transformers whose parameters and activations cannot fit on a single device. PP partitions the model along depth and assigns different layer blocks to different physical pipeline ranks. Virtual pipeline parallelism (VPP) further splits each physical stage into smaller virtual chunks, typically at layer granularity, and interleaves their execution to reduce pipeline bubbles and improve hardware utilization. For cross-layer KV sharing, the main systems concern is whether a reuse layer needs to fetch source KV tensors from a different PP rank; such accesses would introduce additional cross-stage communication and complicate activation lifetime management.

DepthGraft is designed to avoid this issue. Because each deep layer reuses a stride-aligned shallow–middle source pair, the source and reuse layers can be co-located on the same physical PP rank under an appropriate VPP partition. Table 12 gives a balanced example for a 36-layer model with PP size 4. Using zero-based layer indices, the network is divided into three 12-layer stages: layers 0–11, 12–23, and 24–35. Each deep layer $i + 24$ reconstructs KV from the stride-aligned pair $(i, i + 12)$, and each triple $(i, i + 12, i + 24)$ is placed on the same physical PP rank. Thus, cross-layer KV production and consumption remain local to one device.

Table 12: A balanced PP/VPP-compatible layer placement for a 36-layer model with PP size 4. Each stride-aligned source–reuse triple is placed on the same physical PP rank.

Depth band	**PP rank 0**	**PP rank 1**	**PP rank 2**	**PP rank 3**
Shallow sources	0, 1, 2	3, 4, 5	6, 7, 8	9, 10, 11
Middle sources	12, 13, 14	15, 16, 17	18, 19, 20	21, 22, 23
Deep reuse	24, 25, 26	27, 28, 29	30, 31, 32	33, 34, 35

This layout preserves the natural forward order as three repeated PP passes over shallow, middle, and deep depth bands, while keeping the non-local KV dependency local within each physical rank. With layer-granular VPP chunks, no additional inter-rank KV transfer is required for DepthGraft under this placement. In this balanced setting, every PP rank owns exactly three shallow layers, three middle layers, and three deep reuse layers, so the per-rank layer count is identical. For practical model depths that are not perfectly divisible by the number of PP ranks, the placement may introduce mild imbalance, but the final load remains close. This locality property is important for scaling DepthGraft to large distributed pretraining runs without changing the communication pattern of the baseline PP/VPP system.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are clearly presented in the abstract and introduction, and are further elaborated and substantiated throughout the subsequent sections of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Refer to Sec. 5

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: In Appendix C, we provide a detailed theoretical analysis under explicit assumptions.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 4 and Appendix A provide detailed model configurations, optimization settings, and evaluation protocols for implementing the method and evaluating its relative benefits under matched data and training conditions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The project page is available at https://github.com/jianke0604/DepthGraft.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Sec. 4 and Appendix A specify model architectures, KV-sharing configurations, optimizer settings, learning rates and schedules, warmup steps, weight decay, gradient clipping, context length, and few-shot evaluation protocols. Comparisons at each model scale use matched data and training recipes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Consistent with standard practice in LLM pretraining research, we do not perform repeated runs for statistical significance. Instead, we demonstrate the reliability of our method through consistent improvements across diverse benchmarks (Tables 1 and 2) and sustained advantages throughout training (Fig. 1).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Due to privacy restrictions, we do not disclose compute-resource details; the reported experimental settings support reproducing and evaluating the method's relative benefits under matched conditions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The proposed method can reduce KV-cache memory and prefill cost, which may lower the computational and energy cost of serving long-context language models. At the same time, improving the efficiency and quality of large language models may also make

harmful uses of such models easier to scale. Our work is architectural and does not target a specific deployment scenario; standard safety, privacy, and misuse-mitigation practices should therefore be applied when using models trained with this method.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators and original owners of all assets used in the paper are properly credited, and the licenses and terms of use are explicitly mentioned and fully respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.