Knocking-Heads Attention

Anonymous ACL submission

Abstract

Talking-heads attention mitigates the issue of head isolation in existing multi-head attention mechanisms by linearly mixing attention maps across heads. While this improves representational capacity, it is incompatible with existing attention kernels and incurs significant computational overhead. To address this limitation, we propose knocking-heads attention (KHA), which enables attention heads to "knock" on each other — facilitating cross-head feature-level interactions before the scaled dot-product attention. This is achieved by applying a shared, diagonally-initialized projection matrix across all heads. The diagonal initialization preserves head-specific specialization at the start of training while allowing the model to progressively learn integrated cross-head representations. KHA adds only minimal parameters and FLOPs and can be seamlessly integrated into MHA, GQA, GTA, and other attention variants. We validate KHA by training a 6.1B parameter MoE model (1.01B activated) on 1T high-quality tokens. Compared to standard attention, KHA brings superior and more stable training dynamics, achieving better performance across downstream tasks.

1 Introduction

One of the key factors behind the success of large language models is the design of multi-head attention (MHA) (Vaswani et al., 2017), which has been preserved across various subsequent attention variants (Shazeer et al., 2020; Ainslie et al., 2023; Liu et al., 2024; Zadouri et al., 2025). MHA enables models to simultaneously process information from multiple representation subspaces across different sequence positions, with each attention head serving distinct functions in sequence modeling (Xiao et al., 2023, 2024; Qin et al., 2025).

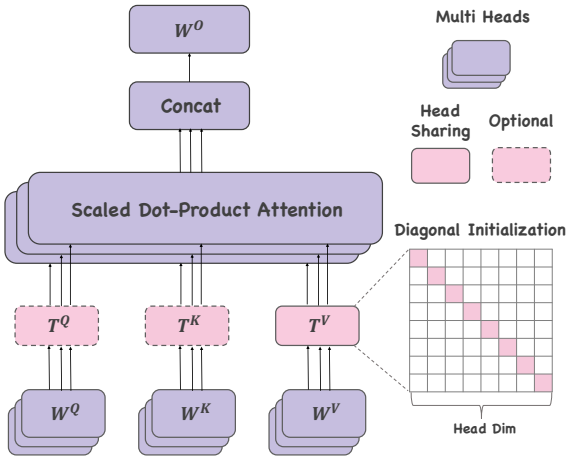


Figure 1: The knocking-heads attention architecture. Purple represents the original multi-head attention, while pink represents the added knocking-heads projections. T^Q and T^K within the dashed box are optional projections due to their lower importance compared to T^V.

However, different heads operate independently when modeling attention, with each head computing its output separately before concatenation and forward propagation, lacking mutual interaction. Talking-heads attention (Shazeer et al., 2020) addresses this limitation by introducing linear projections on either logits before the attention softmax operation or attention scores after the softmax operation to enable inter-head communication. Nevertheless, talking-heads projections on quadratically-scaling attention matrices lead to dramatically increased computational complexity, especially when the number of heads is large.

In this paper, we propose knocking-heads attention (KHA) (Fig. 1), a variant of multi-head attention that achieves inter-head interaction at the feature level through unified transformations applied to all heads. Compared to standard multi-head attention, KHA introduces additional shared projections for query, key, and value representations, supplementing the original head-specific projections. Among these,

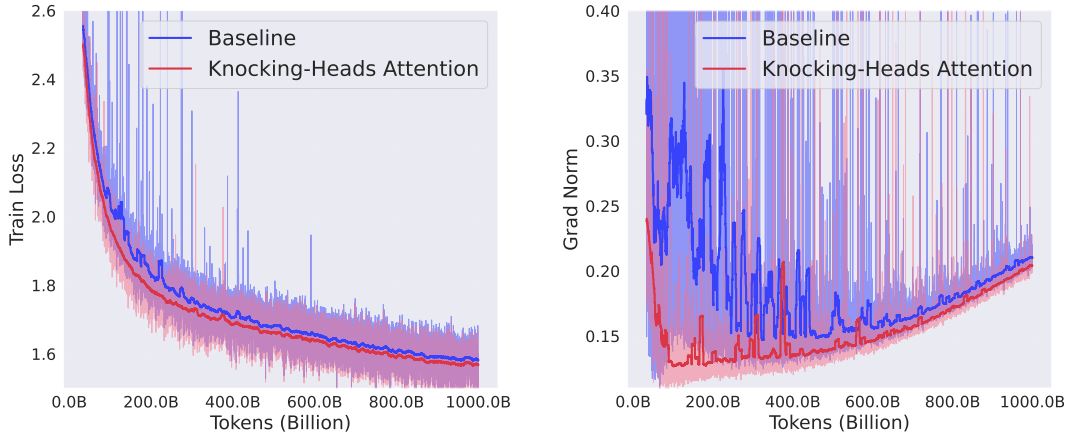


Figure 2: (Left) Training loss over 1T tokens for 6.1B MoE models (1.01B activated parameters): baseline *vs.* knocking-heads attention. (Right) The comparisons of gradient norms. The results show that KHA maintains consistently lower training loss and exhibits better stability for both loss and gradient norm.

the knocking-heads projections for queries and keys are optional since they provide smaller improvements compared to the value projections. We initialize these knocking-heads projections with diagonal matrices, allowing the model to start from isolated heads and gradually develop inter-head communication during training, eventually reaching an optimal balance between head specialization and cross-head collaboration. **In terms of effectiveness**, KHA improves expressiveness by coupling different heads via shared projections. Meanwhile, parameter sharing across heads can act as an implicit regularizer, potentially alleviating head-wise activation/gradient outliers, which in turn improve training stability. **In terms of efficiency**, knocking-heads projections introduce minimal overhead ($< 1\%$ additional FLOPs and parameters) during training and, when implemented as linear transformations rather than MLPs, can be fully integrated into the original projections at inference time.

We conduct extensive experiments on language modeling. We first perform architectural exploration to identify the optimal configuration for knocking-heads projections, examining different projection types (linear, MLP-based, gated), target components (queries, keys, values), attention variants (MHA, MQA, GQA, GTA), and head configurations. Our analysis reveals that applying MLP-based knocking-heads projections to values yields the most significant improvements, with the benefits becoming evident at 4 value/key heads. Based on these findings, we adopt GQA with 32 query heads and 4 KV groups, enabling knocking-heads projections to achieve high performance while maintaining efficient KV caching. We validate our approach through comprehensive experiments on 1T tokens using 1.01B-active-parameter, 6.1B-total-parameter MoE models, with results shown in Fig. 2. Our method achieves significant improvements in Language Understanding (+4.32), Code (+3.9), and Math (+1.62) tasks, with an overall average score improvement of 1.26 points across all evaluated tasks. Additional experiments across various MoE and dense model scales further demonstrate the effectiveness of KHA. We summarize our main contributions as follows:

- We introduce knocking-heads attention, a novel head interaction mechanism that enables cross-head communication and improves training stability with minimal computational overhead. To preserve head specialization, we propose diagonal initialization that allows heads to maintain their distinct roles while gradually developing beneficial cross-head interactions.

- We conduct extensive experiments validating KHA's universality across different attention mechanisms, model types (MoE and dense), and model scales, demonstrating broad applicability and identifying optimal configurations.

- We demonstrate scalability with large-scale experiments on 1T tokens using a model with 1.01B active parameters (6.1B total), achieving consistent improvements in training stability and performance.

2 Related Work

Multi-head attention mechanisms allow different attention heads to learn diverse patterns (Xiao et al., 2023, 2024), but these heads lack interconnection and often exhibit significant redundancy (Cordonnier et al., 2020; Jin et al., 2024). Several approaches have explored inter-head communication to address this issue. Talking-heads attention (Shazeer et al., 2020) applies learnable transformations to attention weight matrices, but requires materializing these matrices, making it incompatible with FlashAttention (Dao et al., 2022), not to mention its substantial computational overhead when head counts are large relative to head dimensions. Collaborated multi-head attention (Cordonnier et al., 2020) shares projection matrices across heads with head-wise dimension-specific mixers, yet increases training FLOPs while limiting head specialization to feature reweighting. Mixture-of-head attention (Jin et al., 2024) dynamically selects head subsets per token but provides limited inter-head communication while incurring complex router training. Our proposed knocking-heads attention adopts head-sharing projections similar to collaborated attention but preserves head specialization through keeping original head-specific projections and diagonal initialization. The plug-and-play mechanism is compatible with any attention variants and FlashAttention framework, adding minimal FLOPs and parameters. More detailed comparisons are provided in Table 6 in the appendix.

3 Method

3.1 Classical Attention

Multi-head attention (MHA) enables the model to jointly attend to information from different representation subspaces. Given an input sequence $\mathbf{X} \in \mathbb{R}^{L\times d}$ where L is the sequence length and d is the hidden dimension, MHA employs n parallel attention heads to capture diverse attention patterns.

For each attention head $i \in \{1, 2, \ldots, n\}$, the queries, keys, and values are computed as:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \quad \mathbf{W}_i^Q \in \mathbb{R}^{d\times d_k}, \tag{1}$$

$$\mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \quad \mathbf{W}_i^K \in \mathbb{R}^{d\times d_k}, \tag{2}$$

$$\mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V, \quad \mathbf{W}_i^V \in \mathbb{R}^{d\times d_v}, \tag{3}$$

where $d_k = d_v = d/n$ represents the dimension of each head. The attention output for head i is computed as $\mathbf{O}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i$, following the scaled dot-product attention mechanism. The final MHA output concatenates all head outputs and applies a linear projection. To highlight the additive nature of multi-head contributions, we decompose the output projection matrix $\mathbf{W}^O \in \mathbb{R}^{d\times d}$ into block form $\begin{bmatrix}\mathbf{W}_1^O \\ \vdots \\ \mathbf{W}_n^O\end{bmatrix}$, where each $\mathbf{W}_i^O \in \mathbb{R}^{d_v\times d}$ corresponds to the projection for head i. This yields:

$$\begin{aligned}\text{MHA}(\mathbf{X}) &= \text{Concat}(\mathbf{O}_1, \ldots, \mathbf{O}_n)\mathbf{W}^O \\ &= [\mathbf{O}_1 \ \cdots \ \mathbf{O}_n]\begin{bmatrix}\mathbf{W}_1^O \\ \vdots \\ \mathbf{W}_n^O\end{bmatrix} \\ &= \sum_{i=1}^{n}\mathbf{O}_i\mathbf{W}_i^O,\end{aligned} \tag{4}$$

showing that the output is a sum of individual head contributions. Group Query Attention (GQA) extends this framework by partitioning n query heads into g groups, where each group shares the same key-value pair. This design reduces computational overhead while maintaining representational capacity, effectively interpolating between MHA ($g = n$) and multi-query attention ($g = 1$) (Shazeer, 2019).

3.2 Knocking-Heads Attention

In MHA, each head operates with reduced dimensions, creating a low-rank bottleneck that limits individual head expressiveness (Bhojanapalli et al., 2020). Inspired by talking-heads attention (Shazeer et al., 2020) and head-sharing projections in collaborated attention (Cordonnier et al., 2020), we propose knocking-heads attention (KHA). After $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ projections but before individual scaled dot-product attention computations, KHA introduces head-sharing projection matrices (*i.e.* knocking-heads projections) alongside the original head-specific projections that enable head interaction while keeping head specification.

We explore two variants of knocking-heads projections that offer complementary advantages: linear transformations provide inference efficiency through matrix absorption, while

MLP transformations offer enhanced expressiveness through non-linearity.

KHA-Linear KHA-Linear applies shared linear transformations to enhance head coordination. For each head i, the transformed queries, keys, and values are computed as:

$$\tilde{\mathbf{Q}}_i = \mathbf{Q}_i \mathbf{T}^Q, \quad \mathbf{T}^Q \in \mathbb{R}^{d_k \times d_k}, \tag{5}$$

$$\tilde{\mathbf{K}}_i = \mathbf{K}_i \mathbf{T}^K, \quad \mathbf{T}^K \in \mathbb{R}^{d_k \times d_k}, \tag{6}$$

$$\tilde{\mathbf{V}}_i = \mathbf{V}_i \mathbf{T}^V, \quad \mathbf{T}^V \in \mathbb{R}^{d_v \times d_v}, \tag{7}$$

where $\mathbf{T}^Q$, $\mathbf{T}^K$, and $\mathbf{T}^V$ are shared transformation matrices across all heads. Crucially, during inference, these transformations can be absorbed into the original projection matrices:

$$\mathbf{W}_i^{Q'} = \mathbf{W}_i^Q \mathbf{T}^Q, \tag{8}$$

$$\mathbf{W}_i^{K'} = \mathbf{W}_i^K \mathbf{T}^K, \tag{9}$$

$$\mathbf{W}_i^{V'} = \mathbf{W}_i^V \mathbf{T}^V, \tag{10}$$

eliminating computation while preserving the benefits of enhanced head coordination.

Key Takeway As we will demonstrate in Section 4.2.3, $\mathbf{T}^V$ is the most critical component, as values learn head interactions most effectively. The transformations $\mathbf{T}^Q$ and $\mathbf{T}^K$ are optional, as their removal causes negligible performance degradation. Similar phenomenon can also be observed in the value residual (Zhou et al., 2025).

KHA-MLP To leverage non-linear expressiveness, we introduce an MLP-based transformation that our experiments show outperforms pure linear approaches. Given the parameter overhead of applying MLP to all queries, keys, and values, we focus solely on the most critical values, which maintains the same parameter count as the linear variant while providing superior representational capacity:

$$\begin{aligned} \tilde{\mathbf{V}}_i &= \mathrm{MLP}(\mathbf{V}_i) \\ &= 2 \cdot \left(\mathbf{V}_i \mathbf{W}^{\mathrm{up}} \odot \mathrm{Sigmoid}(\mathbf{V}_i \mathbf{W}^{\mathrm{gate}}) \right) \mathbf{W}^{\mathrm{down}}, \end{aligned} \tag{11}$$

where $\mathbf{W}^{\mathrm{up}}, \mathbf{W}^{\mathrm{gate}}, \mathbf{W}^{\mathrm{down}} \in \mathbb{R}^{d_v \times d_v}$ are shared across all heads. We use sigmoid-activated MLPs to facilitate zero initialization. Our ablation studies confirm that applying gating alone introduces detrimental effects, while the complete MLP structure enhances model expressiveness.

3.3 Initialization Strategy

The effectiveness of KHA relies critically on "zero" initialization of shared matrices to ensure they approximate identity mappings during early training. Our experiments show that random initialization causes loss to converge to much higher values, potentially making all heads overly similar.

For the KHA-Linear, we apply diagonal initialization to $\mathbf{T}^Q$, $\mathbf{T}^K$, and $\mathbf{T}^V$. For the KHA-MLP, $\mathbf{W}^{\mathrm{up}}$ and $\mathbf{W}^{\mathrm{down}}$ are diagonal-initialized, while $\mathbf{W}^{\mathrm{gate}}$ is zero-initialized. This ensures that $\mathrm{sigmoid}(\mathbf{V}_i \mathbf{W}^{\mathrm{gate}}) \cdot 2 = 1$ initially, which motivates our choice of sigmoid activation in Equation 11.

This initialization allows off-diagonal elements to progressively learn non-zero values, enabling the model to first establish head specialization before learning inter-head interactions.

3.4 Complexity Analysis

The training FLOPs for multi-head attention can be computed as:

$$\begin{aligned} \mathrm{FLOP}_{\mathrm{MHA}} &= 6Ld^2 + 4nL^2 d_k + 2Ld^2 \\ &= 8Ld^2 + 4L^2 d, \end{aligned} \tag{12}$$

where $6Ld^2$ accounts for query, key, and value projection forward and backward passes, $4nL^2 d_k$ represents attention score computation and attention-value multiplication, and $2Ld^2$ corresponds to the output projection matrix $\mathbf{W}^O$. Assuming $d_{\mathrm{ffn}} = 3d$, the training FLOPs for the feed-forward network (FFN) with up-gate-down structure can be computed as: $\mathrm{FLOP}_{\mathrm{FFN}} = 6Ld \cdot 3d = 18Ld^2$, where the factor of 6 accounts for forward and backward passes through three linear layers (up, gate, and down projection). Therefore, the total single-layer training FLOPs is:

$$\begin{aligned} \mathrm{FLOP}_{\mathrm{total}} &= \mathrm{FLOP}_{\mathrm{MHA}} + \mathrm{FLOP}_{\mathrm{FFN}} \\ &= 26Ld^2 + 4L^2 d. \end{aligned} \tag{13}$$

For both knocking-heads variants, the additional training FLOPs are identical:

$$\mathrm{FLOP}_{\mathrm{KHA}} = 6nL(\frac{d}{n})^2 = \frac{6Ld^2}{n} \tag{14}$$

where n is the number of attention heads. For a concrete example with $L = 2048$, $d = 1024$ and $n = 32$, the knocking-heads overhead represents only 0.55% of the total layer computation and 1.17% of the original MHA computation, demonstrating the efficiency.

4 Experiment

4.1 Training Setup

We conduct two sets of experiments: main experiments on 1T tokens and exploratory experiments on 100B tokens, both using high-quality multi-domain data.

Architecture We adopt MoE architectures for most of our experiments, as they consistently outperform dense models under equivalent training FLOPs (Dai et al., 2024). To maintain expert load balancing, we implement an updated loss-free balancing strategy (Wang et al., 2024a; Su, 2025). We follow Qwen's design (Yang et al., 2025) for attention implementation with QK RMS normalization, and maintain a head dimension of 128 throughout all experiments. For the main experiments, we use a configuration with 128 experts where 8 are activated, yielding 6.1B total parameters with 1.01B active parameters. We employ 32 attention heads with grouped query attention (group size $g = 4$) For exploratory experiments, we scale down to 64 experts with 4 activated and vary the hidden dimension to create model variants ranging from A0.44B-2.3B up to A1.6B-14.6B. These experiments evaluate different attention mechanisms including MHA, MQA, GTA, MLA, and GLA, as well as various GQA configurations.

Hyperparameters All models are trained using Adam optimizer with weight decay 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and gradient clipping at 1.0. We use FSDP for distributed training. The learning rate schedule includes 5% warmup steps followed by cosine annealing to 10% of peak rate at training completion. For main experiments, we use learning rate 4.78e-4, sequence length 8,192, and batch size 4.2M tokens. Exploratory experiments use learning rate 7.5e-4, sequence length 4,096, and batch size 2.1M tokens.

4.2 Architecture Exploration

We conduct exploratory experiments to evaluate KHA from the following perspectives. First, we examine the impact of different numbers of heads on KHA performance (Section 4.2.1). Second, we explore different positions and methods for applying knocking-heads projections (Section 4.2.2). Third, we test the compatibility of KHA with other attention variants (Section 4.2.3).

4.2.1 Ablation Study on Head Number

Table 1: Loss of knocking-heads variants with different KV heads number in GQA (32 query heads). All models reported are A0.8-6.6B MoE trained on 75B tokens. ΔL denotes the loss difference after adopting KHA, where lower values are better.

Model	# KV heads					
	1	2	4	8	16	32
Baseline	1.864	1.855	1.856	1.851	1.849	1.846
KHA-MLP	1.846	1.835	1.832	1.832	1.833	1.83
ΔL	**-0.017**	**-0.020**	**-0.024**	**-0.019**	**-0.016**	**-0.016**
KHA-Linear	1.857	1.845	1.838	1.841	1.832	1.834
ΔL	**-0.007**	**-0.010**	**-0.018**	**-0.010**	**-0.017**	**-0.012**

KHA's core mechanism relies on head-sharing, making it sensitive to the number of attention heads. We evaluate both KHA variants on GQA with varying KV head groups while fixing query heads. All experiments were performed using a 0.8B activated parameter model (6.6B total parameters) trained on 75B tokens. As shown in Table 1, the effectiveness of KHA scales with KV head count. Both variants show significant improvements as KV heads increase from 1 to 4, confirming that more heads provide greater opportunities for cross-head information sharing. KHA-MLP is more stable across different configurations than the linear-based ones, likely due to the diagonal-initialized MLP introducing beneficial non-linearity beyond head-sharing alone.

Table 2: Loss comparison of knocking-heads variants across different shared projection types (linear, gate, MLP) and positions (query, key, value). All models reported are A0.8-6.6B MoE with GQA ($g = 4$) trained on 75B tokens. Green indicates the **best** setting and red indicates the **worst** setting.

Type	Place			Loss	ΔL	Type	Place			Loss	ΔL
	Q	K	V				Q	K	V		
-	-	-	-	1.856	–	Gate	-	-	✓	1.919	**+0.063**
Linear	✓	-	-	1.849	**-0.007**	MLP	✓	-	-	1.848	**-0.008**
	-	✓	-	1.848	**-0.008**		-	✓	-	1.848	**-0.008**
	-	-	✓	1.839	**-0.017**		-	-	✓	1.832	**-0.024**
	✓	✓	✓	1.838	**-0.018**		✓	✓	✓	1.832	**-0.024**

4.2.2 Ablation Studies on Different Variants of Knocking-heads

We experiment with knocking-heads variants beyond the KHA-Linear and KHA-MLP presented in Section 3.2, exploring different architectural choices for inter-head knowledge sharing. The training setup is similar as Section 4.2.1 and we fix the group size $g = 4$ in GQA based on the results in Table 1. Our investigation focused on two key design dimensions: (1) the type of shared transformation matrices (linear layers, gating mechanisms, or MLPs) and (2) their placement within the attention mechanism (query, key, or value projections).

Table 2 reveals several interesting findings. First, incorporating shared transformations across attention heads consistently improves performance, with benefits observed across all three projection positions (Query, Key, Value) and for both linear and MLP-based transformations. Interestingly, value projections benefit most from the knocking-heads mechanism, suggesting that sharing learned representations in the value space is particularly effective for capturing cross-head dependencies. When comparing transformation types, MLPs demonstrate better performance over linear layers for value projections. This advantage likely stems from non-linear activations in MLPs, but using standalone gating mechanisms as shared transformations proves detrimental to model.

4.2.3 Compatibility with Other Attention Variants

Table 3: Loss of knocking-heads on various attention variants with different head configurations. All models reported are roughly A0.8-6.6B MoE trained on 100B tokens. ΔL denotes the loss difference after adopting KHA, where lower values indicate better performance. KHA-MLP is used for all variants except MLA, which uses KHA-Linear.

Model	Head Dim.	# KV Head	# Query Head	Cache	Activated Params	Baseline	KHA	ΔL
MLA	128	4	8	512+64	970M	1.812	1.801	**-0.011**
GTA	128	4	32	512+64	868M	1.819	1.802	**-0.017**
MQA	128	1	32	256	859M	1.814	1.801	**-0.013**
GQA	128	8	16	2048	795M	1.801	1.791	**-0.010**
GQA	128	4	32	1024	880M	1.807	1.787	**-0.020**
MHA	128	16	16	4096	852M	1.795	1.785	**-0.010**

KHA can actually adapt to any form of multi-head attention mechanism, but we focus specifically on softmax-based attention here. Except for MLA, which only works with KHA-Linear because they don't explicitly materialize queries/keys/values during inference, other variants are compatible with both KHA-Linear and KHA-MLP. Therefore, for MLA, we use KHA-Linear, while for others, based on the results in Table 2, we use KHA-MLP. All experiments were performed using a model with approximately 0.8B activated parameters (around 6.6B total parameters) trained on 100B tokens.

Table 4: Performance comparison with and without KHA (32 query heads, 4 key/value heads), trained on 1T tokens with 1.01B active and 6.1B total parameters. "Overall Average" is the average score of all sub-tasks. Green values indicate **improvements**, while red indicate **decreases**.

	Metric	Baseline	KHA	Δ Score
General Knowledge Reasoning	ARC-challenge	53.56	55.25	+1.69
	AGIEval	32.97	31.33	-1.64
	HellaSwag	62.44	62.13	-0.31
	WinoGrande	60.85	63.77	+2.92
	PIQA	76.39	74.86	-1.53
	Average	57.24	57.47	+0.23
Professional Knowledge	MMLU	51.22	51.24	+0.02
	MMLU-Pro	23.42	21.62	-1.80
	CMMLU	47.52	47.32	-0.20
	C-Eval	49.07	47.02	-2.05
	CommonsenseQA	59.05	59.54	+0.49
	GPQA	26.26	27.27	+1.01
	Average	42.76	42.34	-0.42
Language Understanding	RACE-middle	69.08	73.40	+4.32
	RACE-high	61.89	66.21	+4.32
	Average	65.49	69.81	+4.32
Code	HumanEval-Plus	35.98	43.29	+7.31
	MBPP	35.60	37.60	+2.00
	MBPP-Plus	43.12	45.50	+2.38
	Average	38.23	42.13	+3.90
Math	GSM8K	46.17	47.16	+0.99
	MATH	32.66	33.44	+0.78
	CMATH	66.30	69.40	+3.10
	Average	48.38	50.00	+1.62
Overall Average		49.13	50.39	+1.26

As shown in Table 3, KHA consistently improves all tested variants including GQA, MHA, GTA, and MQA. The results demonstrate the ability of knocking-heads projections to recover performance losses incurred by KV-cache optimizations, allowing models to maintain memory efficiency without sacrificing quality. For example, baseline GQA4(32) underperforms MHA16(16) by 0.012 loss despite using 4× less KV-cache. When both apply knocking-heads projections, GQA4(32) achieves a 0.02 loss reduction, shrinking the gap between GQA4(32) and MHA16(16) to just 0.002. Notably, knocking-heads projections even benefits GTA, which shares non-RoPE features between keys and values, highlighting knocking-heads projection's broad generalizability.

Table 5: Performance of KHA on different architectures with varying model sizes. All models are trained on 100B tokens and adopt 4 key/value heads. The best results in each row are shown in **bold**.

Type	Params	Layers	Hidden Size	GQA	KHA-MLP	KHA-Linear
MoE	A0.44B-2.3B	12	1152	1.896	**1.881**$_{-0.015}$	1.882$_{-0.014}$
	A0.8B-6.6B	18	1536	1.807	**1.787**$_{-0.020}$	1.791$_{-0.016}$
	A1.6B-14.6B	23	2048	1.762	**1.737**$_{-0.025}$	1.742$_{-0.020}$
Dense	0.61B	24	1024	2.017	2.013$_{-0.004}$	**2.007**$_{-0.014}$
	1.68B	24	2048	1.892	1.884$_{-0.008}$	**1.872**$_{-0.020}$
	3.94B	36	2560	1.815	1.811$_{-0.004}$	**1.807**$_{-0.008}$

4.3 Large-scale Pretraining

To validate KHA's effectiveness in large-scale pre-training scenarios, we conducted controlled experiments on a 6.1B-parameter MoE model with 1.01B activated parameters, training on 1T tokens. Based on results in Section 4.2, we chose a baseline configuration with GQA ($g = 4$) and 32 query heads to evaluate adding KHA-MLP's impact. As shown in Fig. 2(Left), the baseline model exhibited numerous training spikes during the first half of training, whereas applying KHA significantly reduced spike frequency. Furthermore, once loss stabilized in the latter half of training, the model with KHA consistently achieved **0.015** lower loss at equivalent training steps.

We comprehensively evaluated the final trained models on the downstream tasks mentioned in Appendix A.2. The results in Table 4 demonstrate that while performance on general knowledge and professional knowledge tasks remained comparable between models with and without KHA, significant improvements were observed in language understanding, code, and math tasks, with average score increases of **4.32**, **3.9**, and **1.62** points, respectively. The overall average improvement across all tasks was **1.26** points. In summary, KHA has proven effective in large-scale training, not only substantially reducing training loss spikes but also delivering superior model performance.

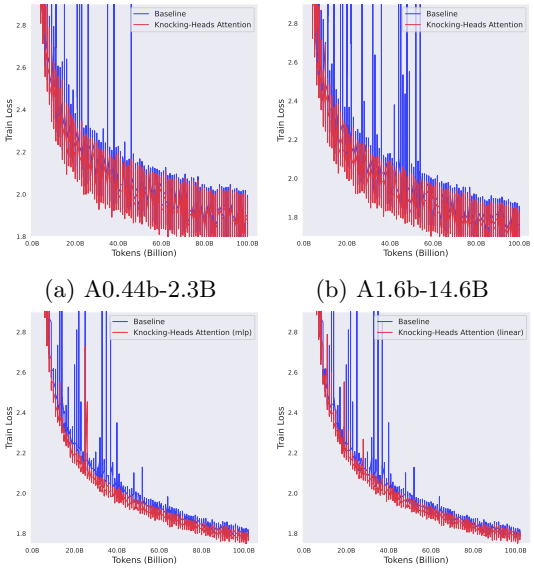
4.4 Scalability Compared with Transformer

As shown in Table 5, we evaluate KHA across different model architectures and scales, training MoE models (2.3B-14.6B parameters) and dense models (0.61B-3.94B parameters) on 100B tokens. All models employ GQA with 4 KV heads, 32 query heads, and 128 head dimensions. Knocking-heads delivers substantial improvements for MoE architectures: while scaling baseline MoE from 6.6B to 14.6B parameters reduces loss by 0.045, knocking-heads attention achieves a loss reduction of 0.02 with negligible training overhead. The benefits become more pronounced at larger MoE scales. Additionally, we find that KHA-Linear provides greater improvements on dense models.

4.5 Visualization

4.5.1 Training Stablity

We present the loss curves for selected experiments from Section 4.3 and Section 4.4 in Fig.2 and Fig.3, respectively. Loss spikes occur more frequently during the first half of training, with the 1.6B activated MoE model exhibiting significantly higher spike frequency than the 0.4B activated mode. Across all model scales, KHA effectively mitigates loss spikes and improves training stability. For the 0.8B activated MoE model, we compare loss curves of both KHA-Linear and KHA-MLP against the baseline. Both variants successfully reduce loss spikes, confirming that the head-sharing mechanism itself suppresses training instability. We hypothesize this stems from the head-sharing mechanism acting as implicit regularization.

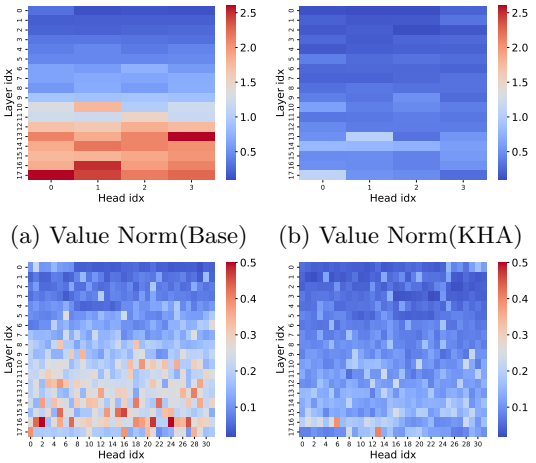


(a) A0.44b-2.3B (b) A1.6b-14.6B

(c) A0.8b-6.6B(MLP) (d) A0.8b-6.6B(Linear)

Figure 3: Training loss curves before and after applying knocking-heads across different model sizes, and the loss curves in (c) and (d) are smoothed for better visualization.

To further investigate why KHA improves



(a) Value Norm(Base) (b) Value Norm(KHA)

(c) Ouput Norm(Base) (d) Ouput Norm(KHA)

Figure 4: Norm analysis for models in Section 4.3.

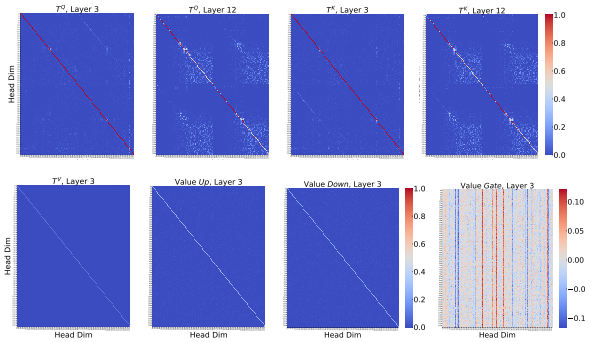


Figure 5: Visualization of learned knocking-heads projection weights across different layers and types. We apply 0-1 clipping to all knocking-heads projection weights except W^{gate}, including T^K, T^Q, T^V, W^{up}, and W^{down}, for comparative analysis.

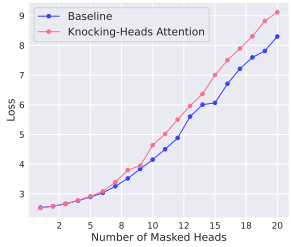


Figure 6: Loss with different number of randomly masked attention heads.

training stability, we computed the norms of intermediate activations, focusing on the outputs and value states of different attention heads. As shown in Fig. 4, model trained with KHA exhibit consistently smaller intermediate-activation norms and a more balanced distribution across heads. In contrast, the baseline model shows larger and more uneven norm distributions, particularly in later layers. In our view, this pattern arises because the shared modules in KHA constrain the degrees of freedom of individual heads, leading to more uniform head-wise activations.

4.5.2 Learnt Weight of Shared Matrix

We visualize the learned knocking-heads projection parameters in Fig. 5. T^Q and T^K, both applied before QK normalization, show similar block-structured patterns: some feature dimensions exhibit low diagonal values indicating inter-head interaction learning, while others retain high diagonal values for head-specific information. Some layers even exhibit minimal inter-head interaction, highlighting the adaptive nature of our method. Notably, T^V displays a distinct pattern with consistently low diagonal values across layers, indicating aggressive head-sharing. This may explain why value projections yield greater improvements in Table. 2. When using MLPs, the gate matrices also exhibit clear structural patterns.

4.6 Analysis of Head Specialization

Motivated by the expert-specialization analyses in DeepSeek-MoE (Dai et al., 2024), we study how KHA affects attention-head specialization. We use the large-scale model trained in Section 4.3. For each layer, we randomly mask a subset of heads by setting the corresponding head outputs in Eq. 4 to zero. As shown in Fig. 6, the loss increases more rapidly with the number of masked heads when using KHA. This sensitivity suggests tighter collaboration among attention heads under KHA, making each head less substitutable. In contrast, the baseline without KHA exhibits more independent and redundant heads.

5 Conclusion

We introduced KHA, a simple enhancement to MHA that enables cross-head communication through shared, diagonally-initialized transformation matrices. Our method addresses the limitation that attention heads operate in isolation while adding less than 1% computational overhead. Through experiments on 1T tokens using 6.1B parameter MoE models, we demonstrate that Knocking-Heads Attention achieves superior performance and significantly improves training stability compared to standard MHA. As a drop-in replacement for standard MHA, our approach offers a practical enhancement for transformer architectures.

Limitations

Although KHA exhibits many appealing properties in our experiments, several challenges remain. While the training scale considered in this paper already exceeds that of most existing studies, KHA may still encounter unforeseen issues when scaled to substantially larger training regimes. The improved training stability brought by KHA currently lacks deeper theoretical understanding and warrants further analysis.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. 2021. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Low-rank bottleneck in multi-head attention models. In *International conference on machine learning*, pages 864–873. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. 2024. Moh: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations.

In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 2002. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. CMMLU: measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11260–11285. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zhen Qin, Xuyang Shen, and Yiran Zhong. 2025. Elucidating the design space of decay in linear attention. *arXiv preprint arXiv:2509.05282*.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, and 1 others. 2025. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou. 2020. Talking-heads attention. *arXiv preprint arXiv:2003.02436*.

Jianlin Su. 2025. Moe travels 3.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Ning Tao, Anthony Ventresque, Vivek Nallur, and Takfarinas Saber. 2024. Enhancing program synthesis with large language models using many-objective grammar-guided genetic programming. *Algorithms*, 17(7):287.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025a. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Ling Team, Binwei Zeng, Chao Huang, Chao Zhang, Changxin Tian, Cong Chen, Dingnan Jin, Feng Yu, Feng Zhu, Feng Yuan, and 1 others. 2025b. Every flop counts: Scaling a 300b mixture-of-experts ling llm without premium gpus. *arXiv preprint arXiv:2503.05139*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. 2024a. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. CMATH: can your language model pass chinese elementary school math test? *CoRR*, abs/2306.16636.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2024. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang

Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ted Zadouri, Hubert Strauss, and Tri Dao. 2025. Hardware-efficient attention for fast decoding. *arXiv preprint arXiv:2505.21487*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2299–2314. Association for Computational Linguistics.

Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. 2025. Value residual learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28341–28356.

A Appendix

A.1 Related Work

A.1.1 Parameter Sharing in Architecture Design

Parameter sharing is a fundamental design principle in deep learning architectures. In CNNs (LeCun et al., 2002), shared convolutional kernels enable translation equivariance and improve generalization capability. ALBERT (Lan et al., 2020) achieves significant parameter reduction through cross-layer parameter sharing. DeepSeek-MoE (Dai et al., 2024) introduces shared experts to reduce parameter redundancy across routed experts. In this work, we introduce additional shared projection matrices (*i.e.*, knocking-heads projections) across different attention heads on top of the original head-specific projections to facilitate inter-head interaction and provide regularization effects.

A.1.2 Comparison of Interactive-heads Attention Mechanisms

We provide a comprehensive comparison of our knocking-heads attention with existing interactive-head mechanisms across multiple dimensions, as summarized in Table 6. Our analysis encompasses three representative approaches: talking-heads attention (Shazeer et al., 2020), collaborated multi-head attention (CollabHead) (Cordonnier et al., 2020), and mixture-of-head (MoH) (Jin et al., 2024).

The compared methods employ fundamentally different interaction strategies. Talking-heads attention uses learnable transition matrices to directly combine attention weights across heads, achieving strong interaction but with high complexity and FlashAttention incompatibility. MoH learns routers to select heads for different tokens, promoting head specialization but lacking direct head interaction. CollabHead enables head interaction by replacing individual head transformation matrices with one large shared matrix across all heads, which compromises head specification and increases training FLOPs due to the enlarged shared matrix. Our knocking-heads mechanism achieves head interaction through lightweight feature-sharing modules inserted into existing attention variants, maintaining both strong interaction and head specification with minimal overhead.

A.1.3 Loss Spikes in Pre-training

Loss spikes are a common challenge in large-scale pretraining and are often induced by particular interactions between the training data and the optimizer's internal state (Team et al., 2025b). Recent work has identified various underlying causes and solutions beyond simply skipping problematic data batches (Chowdhery et al., 2023). (Takase et al., 2023) found correlations between loss spikes and gradient spikes in specific parameters on 1.7B parameter models, proposing scaled initialization for embedding layers. (Qiu et al., 2025) reduced loss spikes by introducing gating mechanisms to avoid massive activations. Kimi-K2 (Team et al., 2025a) identified attention logit spikes in certain heads as the primary cause and proposed QK-clip for mitigation. Our knocking-heads structure addresses this issue differently—by sharing transformation matrices across attention heads, it introduces regularization effects by constraining the behavior of outlier heads, leading to fewer gradient-norm spikes and loss spikes.

A.2 Evaluation Benchmark

We assess model performance using a comprehensive benchmark spanning multiple downstream tasks, which collectively measure different aspects of model competence. The evaluation framework is organized into distinct task categories, including: (a) General Knowledge (*e.g.,* ARC (Bhakthavatsalam et al., 2021), AGIEval (Zhong et al., 2024), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019)) (b) Language Understanding (*e.g.,* RACE (Lai et al., 2017)) (c) Professional Knowledge (*e.g.,* MMLU (Hendrycks et al., 2021a), CMMLU (Li et al., 2024), MMLU-Pro (Wang et al., 2024b), GPQA (Rein et al., 2023), C-Eval (Huang et al., 2023), CommonsenseQA (Talmor et al., 2018))) (d) Math (*e.g.,* GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), CMATH (Wei et al., 2023) (e) Code (*e.g.,* HumanEval-plus (Liu et al., 2023), MBPP (Tao et al., 2024), MBPP-Plus (Liu et al., 2023).

Table 6: Comparison of interactive-heads attention mechanisms: talking-heads (Shazeer et al., 2020), collaborated multi-head (Cordonnier et al., 2020), mixture-of-head (Jin et al., 2024), and our knocking-heads attention. “compute control" is about FLOPs, “compatibility" includes attention variants and FlashAttention support, and “training stability" indicates loss spike frequency.

	Interaction Method	Head-Interaction	Head-Specifiction	Compa-tibility	Compute Control	Param Control	Training Stability
Talking-heads	Mixing	✓Strong	✓Strong	✗	✗	✓	unknown
Collaborated-heads	Sharing	✓Strong	✗Weak	✗	✗	✓	unknown
Mixture-of-head	Re-weight	✗Weak	✓Strong	✓	✓	✓	unknown
Knocking-heads(Ours)	Sharing	✓Strong	✓strong	✓	✓	✓	✓