arXiv:2507.20534v1 [cs.LG] 28 Jul 2025

# KIMI K2: OPEN AGENTIC INTELLIGENCE

TECHNICAL REPORT OF KIMI K2

**Kimi Team**


## ABSTRACT

We introduce Kimi K2, a Mixture-of-Experts (MoE) large language model with 32 billion activated parameters and 1 trillion total parameters. We propose the MuonClip optimizer, which improves upon Muon with a novel QK-clip technique to address training instability while enjoying the advanced token efficiency of Muon. Based on MuonClip, K2 was pre-trained on 15.5 trillion tokens with zero loss spike. During post-training, K2 undergoes a multi-stage post-training process, highlighted by a large-scale agentic data synthesis pipeline and a joint reinforcement learning (RL) stage, where the model improves its capabilities through interactions with real and synthetic environments.

Kimi K2 achieves state-of-the-art performance among open-source non-thinking models, with strengths in agentic capabilities. Notably, K2 obtains 66.1 on Tau2-Bench, 76.5 on ACEBench (En), 65.8 on SWE-Bench Verified, and 47.3 on SWE-Bench Multilingual — surpassing most open and closed-sourced baselines in non-thinking settings. It also exhibits strong capabilities in coding, mathematics, and reasoning tasks, with a score of 53.7 on LiveCodeBench v6, 49.5 on AIME 2025, 75.1 on GPQA-Diamond, and 27.1 on OJBench, all without extended thinking. These results position Kimi K2 as one of the most capable open-source large language models to date, particularly in software engineering and agentic tasks. We release our base and post-trained model checkpoints[1] to facilitate future research and applications of agentic intelligence.


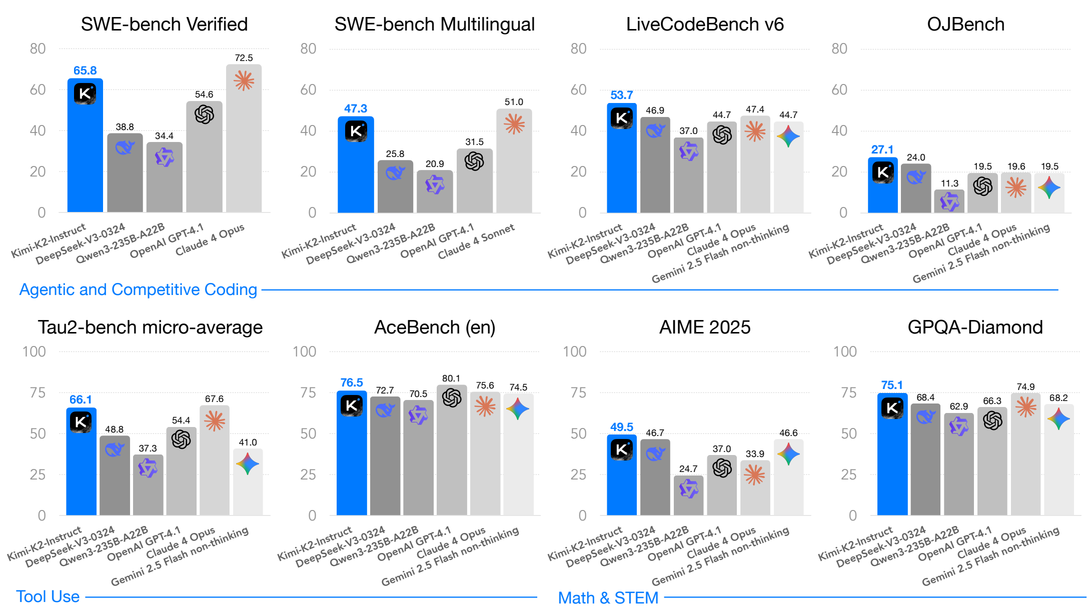


Figure 1: Kimi K2 main results.[2]

[1] https://huggingface.co/moonshotai/Kimi-K2-Instruct

[2] All models evaluated above are non-thinking models. For SWE-bench Multilingual, we evaluated only Claude 4 Sonnet because the cost of Claude 4 Opus was prohibitive.

# 1 Introduction

The development of Large Language Models (LLMs) is undergoing a profound paradigm shift towards *Agentic Intelligence* – the capabilities for models to autonomously perceive, plan, reason, and act within complex and dynamic environments. This transition marks a departure from static imitation learning towards models that actively learn through interactions, acquire new skills beyond their training distribution, and adapt behavior through experiences [63]. It is believed that this approach allows an AI agent to go beyond the limitation of static human-generated data, and acquire superhuman capabilities through its own exploration and exploitation. Agentic intelligence is thus rapidly emerging as a defining capability for the next generation of foundation models, with wide-ranging implications across tool use, software development, and real-world autonomy.

Achieving agentic intelligence introduces challenges in both pre-training and post-training. Pre-training must endow models with broad general-purpose priors under constraints of limited high-quality data, elevating token efficiency—learning signal per token—as a critical scaling coefficient. Post-training must transform those priors into actionable behaviors, yet agentic capabilities such as multi-step reasoning, long-term planning, and tool use are rare in natural data and costly to scale. Scalable synthesis of structured, high-quality agentic trajectories, combined with general reinforcement learning (RL) techniques that incorporate preferences and self-critique, are essential to bridge this gap.

In this work, we introduce Kimi K2, a 1.04 trillion-parameter Mixture-of-Experts (MoE) LLM with 32 billion activated parameters, purposefully designed to address the core challenges and push the boundaries of agentic capability. Our contributions span both the pre-training and post-training frontiers:

- We present **MuonClip**, a novel optimizer that integrates the token-efficient Muon algorithm with a stability-enhancing mechanism called QK-Clip. Using MuonClip, we successfully pre-trained Kimi K2 on 15.5 trillion tokens without a single loss spike.
- We introduce **a large-scale agentic data synthesis pipeline** that systematically generates tool-use demonstrations via simulated and real-world environments. This system constructs diverse tools, agents, tasks, and trajectories to create high-fidelity, verifiably correct agentic interactions at scale.
- We design **a general reinforcement learning framework** that combines verifiable rewards (RLVR) with a self-critique rubric reward mechanism. The model learns not only from externally defined tasks but also from evaluating its own outputs, extending alignment from static into open-ended domains.

Kimi K2 demonstrates strong performance across a broad spectrum of agentic and frontier benchmarks. It achieves scores of 66.1 on Tau2-bench, 76.5 on ACEBench (en), 65.8 on SWE-bench Verified, and 47.3 on SWE-bench Multilingual, outperforming most open- and closed-weight baselines under non-thinking evaluation settings, closing the gap with Claude 4 Opus and Sonnet. In coding, mathematics, and broader STEM domains, Kimi K2 achieves 53.7 on LiveCodeBench v6, 27.1 on OJBench, 49.5 on AIME 2025, and 75.1 on GPQA-Diamond, further highlighting its capabilities in general tasks. On the LMSYS Arena leaderboard (July 17, 2025)[3], Kimi K2 ranks as the top 1 open-source model and 5th overall based on over 3,000 user votes.

To spur further progress in Agentic Intelligence, we are open-sourcing our base and post-trained checkpoints, enabling the community to explore, refine, and deploy agentic intelligence at scale.

# 2 Pre-training

The base model of Kimi K2 is a trillion-parameter mixture-of-experts (MoE) transformer [72] model, pre-trained on 15.5 trillion high-quality tokens. Given the increasingly limited availability of high-quality human data, we posit that token efficiency is emerging as a critical coefficient in the scaling of large language models. To address this, we introduce a suite of pre-training techniques explicitly designed for maximizing token efficiency. Specifically, we employ the token-efficient Muon optimizer [33, 46] and mitigate its training instabilities through the introduction of QK-Clip. Additionally, we incorporate synthetic data generation to further squeeze the intelligence out of available high-quality tokens. The model architecture follows an ultra-sparse MoE with multi-head latent attention (MLA) similar to DeepSeek-V3 [10] , derived from empirical scaling law analysis. The underlying infrastructure is built to optimize both training efficiency and research efficiency.

[3] https://lmarena.ai/leaderboard/text

### 2.1 MuonClip: Stable Training with Weight Clipping

We train Kimi K2 using the token-efficient Muon optimizer [33], incorporating weight decay and consistent update RMS scaling [46]. Experiments in our previous work Moonlight [46] show that, under the same compute budget and model size — and therefore the same amount of training data — Muon substantially outperforms AdamW [36, 48], making it an effective choice for improving token efficiency in large language model training.

**Training instability when scaling Muon** Despite its efficiency, scaling up Muon training reveals a challenge: training instability due to exploding attention logits, an issue that occurs more frequently with Muon but less with AdamW in our experiments. Existing mitigation strategies are insufficient. For instance, logit soft-cap [69] directly clips the attention logits, but the dot products between queries and keys can still grow excessively before capping is applied. On the other hand, Query-Key Normalization (QK-Norm) [11, 81] is not applicable to multi-head latent attention (MLA), because its Key matrices are not fully materialized during inference.

**Taming Muon with QK-Clip** To address this issue, we propose a novel weight-clipping mechanism *QK-Clip* to explicitly constrain attention logits. QK-Clip works by rescaling the query and key projection weights post-update to bound the growth of attention logits.

Let the input representation of a transformer layer be $\mathbf{X}$. For each attention head $h$, its query, key, and value projections are computed as

$$\mathbf{Q}^h = \mathbf{X}\mathbf{W}_q^h, \quad \mathbf{K}^h = \mathbf{X}\mathbf{W}_k^h, \quad \mathbf{V}^h = \mathbf{X}\mathbf{W}_v^h.$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ are model parameters. The attention output is:

$$\mathbf{O}^h = \mathrm{softmax}\left(\frac{1}{\sqrt{d}}\mathbf{Q}^h\mathbf{K}^{h\top}\right)\mathbf{V}^h.$$

We define the max logit, a per-head scalar, as the maximum input to softmax in this batch $B$:

$$S_{\max}^h = \frac{1}{\sqrt{d}} \max_{\mathbf{X}\in B} \max_{i,j} \mathbf{Q}_i^h \mathbf{K}_j^{h\top}$$

where $i, j$ are indices of different tokens in a training sample $\mathbf{X}$.

The core idea of QK-Clip is to rescale $\mathbf{W}_k, \mathbf{W}_q$ whenever $S_{\max}^h$ exceeds a target threshold $\tau$. Importantly, this operation does not alter the forward/backward computation in the current step — we merely use the max logit as a guiding signal to determine the strength to control the weight growth.

A naïve implementation clips all heads at the same time:

$$\mathbf{W}_q^h \leftarrow \gamma^\alpha \mathbf{W}_q^h \qquad \mathbf{W}_k^h \leftarrow \gamma^{1-\alpha} \mathbf{W}_k^h$$

where $\gamma = \min(1, \tau/S_{\max})$ with $S_{\max} = \max_h S_{\max}^h$, and $\alpha$ is a balancing parameter typically set to $0.5$, applying equal scaling to queries and keys.

However, we observe that in practice, only a small subset of heads exhibit exploding logits. In order to minimize our intervention on model training, we determine a per-head scaling factor $\gamma_h = \min(1, \tau/S_{\max}^h)$, and opt to apply per-head QK-Clip. Such clipping is straightforward for regular multi-head attention (MHA). For MLA, we apply clipping only on unshared attention head components:

- $\mathbf{q}^C$ and $\mathbf{k}^C$ (head-specific components): each scaled by $\sqrt{\gamma_h}$
- $\mathbf{q}^R$ (head-specific rotary): scaled by $\gamma_h$,
- $\mathbf{k}^R$ (shared rotary): left untouched to avoid effect across heads.

**MuonClip: The New Optimizer** We integrate Muon with weight decay, consistent RMS matching, and QK-Clip into a single optimizer, which we refer to as **MuonClip** (see Algorithm 1).

We demonstrate the effectiveness of MuonClip from several scaling experiments. First, we train a mid-scale 9B activated and 53B total parameters Mixture-of-Experts (MoE) model using the vanilla Muon. As shown in Figure 2 (Left), we observe that the maximum attention logits quickly exceed a magnitude of 1000, showing that attention logits explosion is already evident in Muon training to this scale. Max logits at this level usually result in instability during training, including significant loss spikes and occasional divergence.

**Algorithm 1** MuonClip Optimizer

**for** each training step $t$ **do**
  // 1. Muon optimizer step
  **for** each weight $\mathbf{W} \in \mathbb{R}^{n \times m}$ **do**
    $\mathbf{M}_t = \mu \mathbf{M}_{t-1} + \mathbf{G}_t$ ▷ $\mathbf{M}_0 = \mathbf{0}$, $\mathbf{G}_t$ is the grad of $\mathbf{W}_t$, $\mu$ is momentum
    $\mathbf{O}_t = \text{Newton-Schulz}(\mathbf{M}_t) \cdot \sqrt{\max(n, m)} \cdot 0.2$ ▷ Match Adam RMS
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta\big(\mathbf{O}_t + \lambda \mathbf{W}_{t-1}\big)$ ▷ learning rate $\eta$, weight decay $\lambda$
  **end for**
  // 2. QK-Clip
  **for** each attention head $h$ in every attention layer of the model **do**
    Obtain $S^h_{\max}$ already computed during forward
    **if** $S^h_{\max} > \tau$ **then**
      $\gamma \leftarrow \tau / S^h_{\max}$
      $\mathbf{W}^h_{qc} \leftarrow \mathbf{W}^h_{qc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{kc} \leftarrow \mathbf{W}^h_{kc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{qr} \leftarrow \mathbf{W}^h_{qr} \cdot \gamma$
    **end if**
  **end for**
**end for**

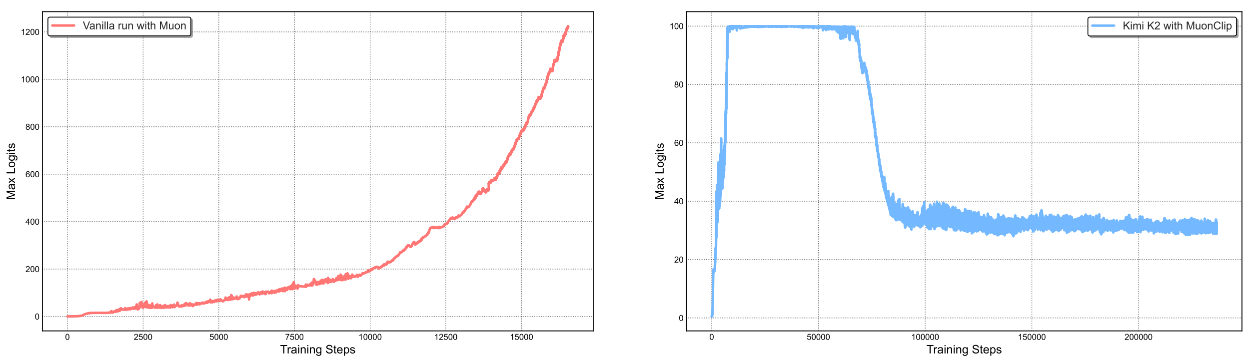


Figure 2: Left: During a mid-scale training run, attention logits rapidly exceed 1000, which could lead to potential numerical instabilities and even training divergence. Right: Maximum logits for Kimi K2 with MuonClip and $\tau$ = 100 over the entire training run. The max logits rapidly increase to the capped value of 100, and only decay to a stable range after approximately 30% of the training steps, demonstrating the effective regulation effect of QK-Clip.

Next, we demonstrate that QK-Clip does not degrade model performance and confirm that the MuonClip optimizer preserves the optimization characteristics of Muon without adversely affecting the loss trajectory. A detailed discussion of the experiment designs and findings is provided in the Appendix D.

Finally, we train Kimi K2, a large-scale MoE model, using MuonClip with $\tau = 100$ and monitor the maximum attention logits throughout the training run (Figure 2 (Right)). Initially, the logits are capped at 100 due to QK-Clip. Over the course of training, the maximum logits gradually decay to a typical operating range without requiring any adjustment to $\tau$. Importantly, the training loss remains smooth and stable, with no observable spikes, as shown in Figure 3, validating that MuonClip provides robust and scalable control over attention dynamics in large-scale language model training.

### 2.2 Pre-training Data: Improving Token Utility with Rephrasing

Token efficiency in pre-training refers to how much performance improvement is achieved for each token consumed during training. Increasing token utility—the effective learning signal each token contributes—enhances the per-token impact on model updates, thereby directly improving token efficiency. This is particularly important when the supply of high-quality tokens is limited and must be maximally leveraged. A naive approach to increasing token utility is through repeated exposure to the same tokens, which can lead to overfitting and reduced generalization.

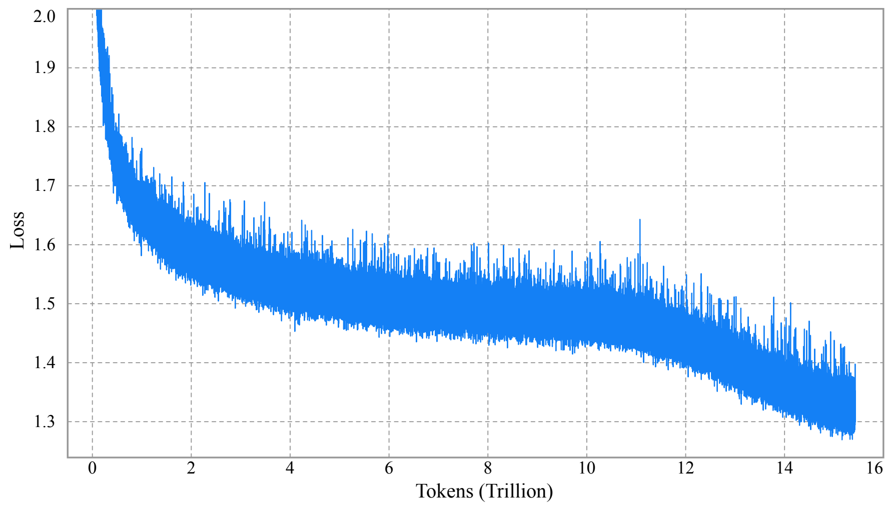


Figure 3: Per-step training loss curve of Kimi K2, **without smoothing or sub-sampling**. It shows no spikes throughout the entire training process. Note that we omit the very beginning of training for clarity.

A key advancement in the pre-training data of Kimi K2 over Kimi K1.5 is the introduction of a synthetic data generation strategy to increase token utility. Specifically, a carefully designed rephrasing pipeline is employed to amplify the volume of high-quality tokens without inducing significant overfitting. In this report, we describe two domain-specialized rephrasing techniques—targeted respectively at the Knowledge and Mathematics domains—that enable this controlled data augmentation.

**Knowledge Data Rephrasing** Pre-training on natural, knowledge-intensive text presents a trade-off: a single epoch is insufficient for comprehensive knowledge absorption, while multi-epoch repetition yields diminishing returns and increases the risk of overfitting. To improve the token utility of high-quality knowledge tokens, we propose a synthetic rephrasing framework composed of the following key components:

- **Style- and perspective-diverse prompting:** To enhance linguistic diversity while maintaining factual integrity, we apply a range of carefully engineered prompts. These prompts guide a large language model to generate faithful rephrasings of the original texts in varied styles and from different perspectives.
- **Chunk-wise autoregressive generation:** To preserve global coherence and avoid information loss in long documents, we adopt a chunk-based autoregressive rewriting strategy. Texts are divided into segments, rephrased individually, and then stitched back together to form complete passages. This method mitigates implicit output length limitations that typically exist with LLMs. An overview of this pipeline is presented in Figure 4.
- **Fidelity verification:** To ensure consistency between original and rewritten content, we perform fidelity checks that compare the semantic alignment of each rephrased passage with its source. This serves as an initial quality control step prior to training.

We compare data rephrasing with multi-epoch repetition by testing their corresponding accuracy on SimpleQA. We experiment with an early checkpoint of K2 and evaluate three training strategies: (1) repeating the original dataset for 10 epochs, (2) rephrasing the data once and repeating it for 10 epochs, and (3) rephrasing the data 10 times with a single training pass. As shown in Table 1, the accuracy consistently improves across these strategies, demonstrating the efficacy of our rephrasing-based augmentation. We extended this method to other large-scale knowledge corpora and observed similarly encouraging results, and each corpora is rephrased at most twice.

Table 1: SimpleQA Accuracy under three rephrasing-epoch configurations

| # Rephrasings | # Epochs | SimpleQA Accuracy |
|---|---|---|
| 0 (raw wiki-text) | 10 | 23.76 |
| 1 | 10 | 27.39 |
| 10 | 1 | 28.94 |

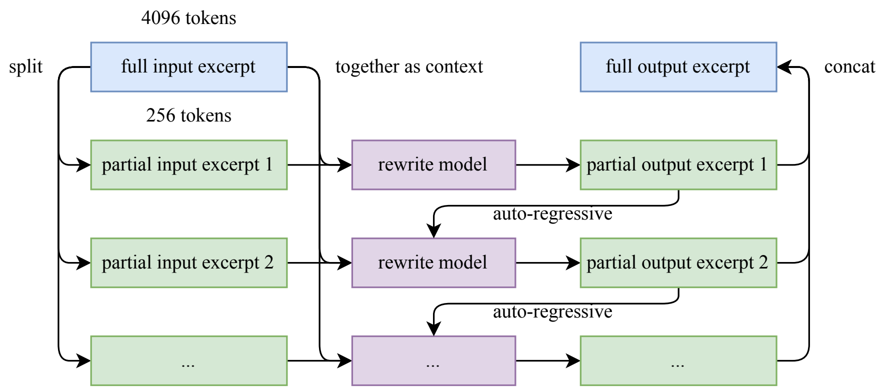


Figure 4: Auto-regressive chunk-wise rephrasing pipeline for long input excerpts. The input is split into smaller chunks with preserved context, rewritten sequentially, and then concatenated into a full rewritten passage.

**Mathematics Data Rephrasing** To enhance mathematical reasoning capabilities, we rewrite high-quality mathematical documents into a "learning-note" style, following the methodology introduced in SwallowMath [15]. In addition, we increased data diversity by translating high-quality mathematical materials from other languages into English.

Although initial experiments with rephrased subsets of our datasets show promising results, the use of synthetic data as a strategy for continued scaling remains an active area of investigation. Key challenges include generalizing the approach to diverse source domains without compromising factual accuracy, minimizing hallucinations and unintended toxicity, and ensuring scalability to large-scale datasets.

**Pre-training Data Overall** The Kimi K2 pre-training corpus comprises 15.5 trillion tokens of curated, high-quality data spanning four primary domains: Web Text, Code, Mathematics, and Knowledge. Most data processing pipelines follow the methodologies outlined in Kimi K1.5 [35]. For each domain, we performed rigorous correctness and quality validation and designed targeted data experiments to ensure the curated dataset achieved both high diversity and effectiveness.

### 2.3 Model Architecture

Kimi K2 is a 1.04 trillion-parameter Mixture-of-Experts (MoE) transformer model with 32 billion activated parameters. The architecture follows a similar design to DeepSeek-V3 [10] , employing Multi-head Latent Attention (MLA) [44] as the attention mechanism, with a model hidden dimension of 7168 and an MoE expert hidden dimension of 2048. Our scaling law analysis reveals that continued increases in sparsity yield substantial performance improvements, which motivated us to increase the number of experts to 384, compared to 256 in DeepSeek-V3. To reduce computational overhead during inference, we cut the number of attention heads to 64, as opposed to 128 in DeepSeek-V3. Table 2 presents a detailed comparison of architectural parameters between Kimi K2 and DeepSeek-V3.

Table 2: Architectural comparison between Kimi K2 and DeepSeek-V3

| | **DeepSeek-V3** | **Kimi K2** | $\Delta$ |
|---|---|---|---|
| #Layers | 61 | 61 | = |
| Total Parameters | 671B | 1.04T | $\uparrow$ 54% |
| Activated Parameters | 37B | 32.6B | $\downarrow$ 13% |
| Experts (total) | 256 | 384 | $\uparrow$ 50% |
| Experts Active per Token | 8 | 8 | = |
| Shared Experts | 1 | 1 | = |
| Attention Heads | 128 | 64 | $\downarrow$ 50% |
| Number of Dense Layers | 3 | 1 | $\downarrow$ 67% |
| Expert Grouping | Yes | No | - |

**Sparsity Scaling Law** We develop a sparsity scaling law tailored for the Mixture-of-Experts (MoE) model family using Muon. Sparsity is defined as the ratio of the total number of experts to the number of activated experts. Through carefully controlled small-scale experiments, we observe that — under a fixed number of activated parameters (i.e., constant FLOPs) — increasing the total number of experts (i.e., increasing sparsity) consistently lowers both the training and validation loss, thereby enhancing overall model performance (Figure 5). Concretely, under the compute-optimal sparsity scaling law, achieving the same validation loss of 1.5, sparsity 48 reduces FLOPs by 1.69×, 1.39×, and 1.15× compared to sparsity levels 8, 16, and 32, respectively. Though increasing sparsity leads to better performance, this gain comes with increased infrastructure complexity. To balance model performance with cost, we adopt a sparsity of 48 for Kimi K2, activating 8 out of 384 experts per forward pass.

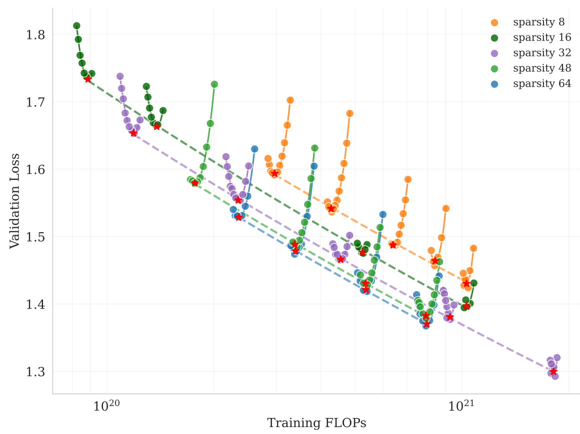


Figure 5: Sparsity Scaling Law. Increasing sparsity leads to improved model performance. We fixed the number of activated experts to 8 and the number of shared experts to 1, and varied the total number of experts, resulting in models with different sparsity levels.

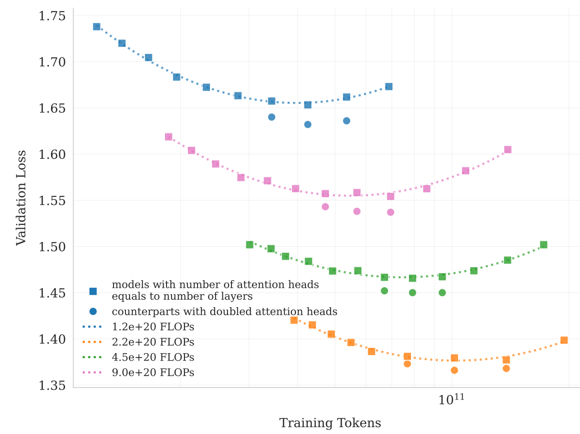


Figure 6: Scaling curves for models with number of attention heads equals to number of layers and their counterparts with doubled attention heads. Doubling the number of attention heads leads to a reduction in validation loss of approximately 0.5% to 1.2%.

**Number of Attention Heads** DeepSeek-V3 [10] sets the number of attention heads to roughly twice the number of model layers to better utilize memory bandwidth and enhance computational efficiency. However, as the context length increases, doubling the number of attention heads leads to significant inference overhead, reducing efficiency at longer sequence lengths. This becomes a major limitation in agentic applications, where efficient long context processing is essential. For example, with a sequence length of 128k, increasing the number of attention heads from 64 to 128, while keeping the total expert count fixed at 384, leads to an 83% increase in inference FLOPs. To evaluate the impact of this design, we conduct controlled experiments comparing configurations where the number of attention heads equals the number of layers against those with double number of heads, under varying training FLOPs. Under iso-token training conditions, we observe that doubling the attention heads yields only modest improvements in validation loss (ranging from 0.5% to 1.2%) across different compute budgets (Figure 6). Given that sparsity 48 already offers strong performance, the marginal gains from doubling attention heads do not justify the inference cost. Therefore we choose to 64 attention heads.

## 2.4 Training Infrastructure

### 2.4.1 Compute Cluster

Kimi K2 was trained on a cluster equipped with NVIDIA H800 GPUs. Each node in the H800 cluster contains 2 TB RAM and 8 GPUs connected by NVLink and NVSwitch within nodes. Across different nodes, $8 \times 400$ Gbps RoCE interconnects are utilized to facilitate communications.

### 2.4.2 Parallelism for Model Scaling

Training of large language models often progresses under dynamic resource availability. Instead of optimizing one parallelism strategy that's only applicable under specific amount of resources, we pursue a flexible strategy that allows Kimi K2 to be trained on any number of nodes that is a multiple of 32. Our strategy leverages a combination of 16-way

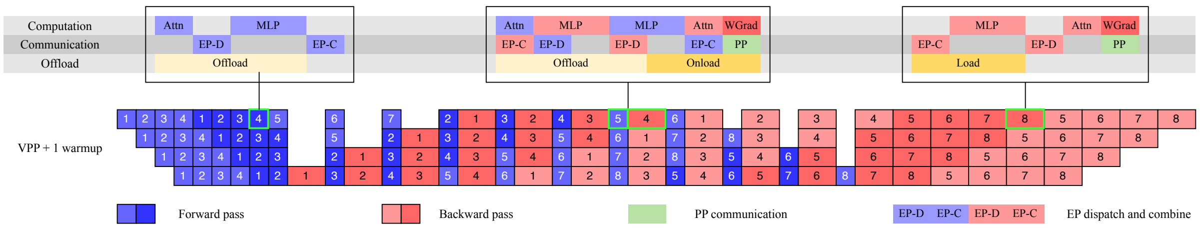


Figure 7: Computation, communication and offloading overlapped in different PP phases.

Pipeline Parallelism (PP) with virtual stages [28, 53, 38, 57, 47, 21], 16-way Expert Parallelism (EP) [39], and ZeRO-1 Data Parallelism [60].

Under this setting, storing the model parameters in BF16 and their gradient accumulation buffer in FP32 requires approximately 6 TB of GPU memory, distributed over a model-parallel group of 256 GPUs. Placement of optimizer states depends on the training configurations. When the total number of training nodes is large, the optimizer states are distributed, reducing its per-device memory footprint to a negligible level. When the total number of training nodes is small (e.g., 32), we can offload some optimizer states to CPU.

This approach allows us to reuse an identical parallelism configuration for both small- and large-scale experiments, while letting each GPU hold approximately 30 GB of GPU memory for all states. The rest of the GPU memory are used for activations, as described in Sec. 2.4.3. Such a consistent design is important for research efficiency, as it simplifies the system and substantially accelerates experimental iteration.

**EP communication overlap with interleaved 1F1B** By increasing the number of warm-up micro-batches, we can overlap EP all-to-all communication with computation under the standard interleaved 1F1B schedule [21, 53]. In comparison, DualPipe [10] doubles the memory required for parameters and gradients, necessitating an increase in parallelism to compensate. Increasing PP introduces more bubbles, while increasing EP, as discussed below, incurs higher overhead. The additional costs are prohibitively high for training a large model with over 1 trillion parameters and thus we opted not to use DualPipe.

However, interleaved 1F1B splits the model into more stages, introducing non-trivial PP communication overhead. To mitigate this cost, we decouple the weight-gradient computation from each micro-batch's backward pass and execute it in parallel with the corresponding PP communication. Consequently, all PP communications can be effectively overlapped except for the warm-up phase.

**Smaller EP size** To ensure full computation-communication overlap during the 1F1B stage, the reduced attention computation time in K2 (which has 64 attention heads compared to 128 heads in DeepSeek-V3) necessitates minimizing the time of EP operations. This is achieved by adopting the smallest feasible EP parallelization strategy, specifically EP = 16. Utilizing a smaller EP group also relaxes expert-balance constraints, allowing for near-optimal speed to be achieved without further tuning.

### 2.4.3 Activation Reduction

After reserving space for parameters, gradient buffers, and optimizer states, the remaining GPU memory on each device is insufficient to hold the full MoE activations. To ensure the activation memory fits within the constraints, especially for the initial pipeline stages that accumulate the largest activations during the 1F1B warm-up phase, the following techniques are employed.

**Selective recomputation** Recomputation is applied to inexpensive, high-footprint stages, including `LayerNorm`, `SwiGLU`, and `MLA` up-projections [10]. Additionally, `MoE` down-projections are recomputed during training to further reduce activation memory. While optional, this recomputation maintains adequate GPU memory, preventing crashes caused by expert imbalance in early training stages.

**FP8 storage for insensitive activations** Inputs of `MoE` up-projections and `SwiGLU` are compressed to FP8-E4M3 in $1\times$ 128 tiles with FP32 scales. Small-scale experiments show no measurable loss increase. Due to potential risks of performance degradation that we observed during preliminary study, we do not apply FP8 in computation.