A.X K2 Technical Report

SK Telecom

🤗 https://huggingface.co/skt/A.X-K2

Abstract

We introduce A.X K2, a 688B-parameter Mixture-of-Experts (MoE) language model trained from scratch as a high-performance foundation for *agentic* applications. A.X K2 is trained on approximately 8.5T tokens in total—fewer than its predecessor, A.X K1—by prioritizing a smaller but higher-quality data mixture; it nonetheless improves substantially over A.X K1 across the board—by over 30 percentage points on some benchmarks—reflecting substantial gains in token efficiency from an improved multi-stage data-processing pipeline. To support long contexts efficiently, we introduce Sparse Gated Attention (SGA), which combines sparse attention with gated attention, and adopt Gated Norm to stabilize large-scale training. To strengthen agentic capability, we substantially expand agentic and software-engineering data and extend the usable context length, enabling multi-step tool use and long-horizon task execution. A.X K2 further supports explicitly controllable reasoning through a simple yet effective Think-Fusion training recipe, enabling user-controlled switching between thinking and non-thinking modes within a single unified model. Extensive evaluations demonstrate that A.X K2 performs competitively against strong open-weight baselines, matching or exceeding them on math and Korean-language benchmarks.

1 Introduction

Large language models (LLMs) have advanced rapidly through the scaling of model capacity and advances in reinforcement learning, and the frontier is increasingly defined by *agentic* competence—multi-step reasoning, tool use, and long-horizon task execution (DeepSeek-AI, 2026; Kimi Team, 2025; Qwen Team, 2026b; GLM-5 Team, 2026). Turning such capabilities into deployable systems, however, poses challenges that raw scale alone does not resolve: a practical foundation model must pair broad knowledge and strong reasoning with efficient serving, controllable inference-time compute, and trainability under realistic data and hardware budgets.

We present **A.X K2**, a 688B-parameter Mixture-of-Experts (MoE) language model with 33B active parameters, trained from scratch as a high-performance, agentic foundation model and the successor to A.X K1 (SKT, 2026). Guided by MoE scaling laws (Tian et al., 2025), the architecture maximizes knowledge capacity within fixed hardware constraints while prioritizing inference throughput over strictly compute-optimal training. A.X K2 is trained on approximately 8.5T tokens in total—about 8.2T in pre-training and the remainder in post-training, fewer than A.X K1—using a smaller but higher-quality mixture ori-

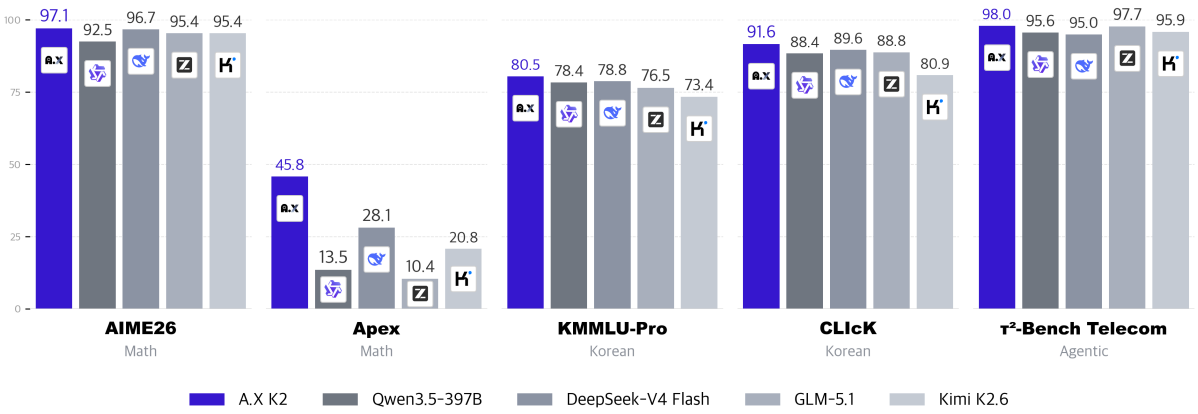


Figure 1: **Performance comparison of A.X K2 against leading open-weight models across representative benchmarks** (thinking mode; cf. Table 6). A.X K2 leads on mathematics (AIME26, Apex), Korean-language understanding (KMMLU-Pro, CLIcK), and agentic tool use (τ^2-Bench Telecom).

ented toward agentic workloads, yet it remains competitive with leading open-weight models, reflecting substantial gains in token efficiency.

Because deep reasoning is costly—long chains of thought inflate latency and serving cost (Snell et al., 2024) and are wasteful on simple queries—A.X K2 gives users explicit control over the extent of reasoning. Through a simple yet effective *Think-Fusion* recipe, a single unified model supports both a *thinking* mode for complex problems and a *non-thinking* mode for concise, low-latency responses, allowing deployments to trade quality for cost on a per-request basis.

Beyond its technical contributions, A.X K2 is part of a national *Sovereign AI* effort. By building a frontier-scale model that deeply understands the Korean language and culture, we aim to reduce reliance on foreign proprietary systems and accelerate the adoption of trustworthy, controllable AI across Korean industry, government, and academia.

We highlight four key aspects of A.X K2 as follows:

- **Token-Efficient Pre-training:** We demonstrate large-scale MoE training with 688B total parameters (and 33B active parameters) using approximately 8.5T tokens in total ($\sim$8.2T in pre-training), guided by scaling laws under a fixed compute budget. Despite training on fewer tokens than A.X K1 (SKT, 2026) ($\sim$10T), A.X K2 shows substantial improvements across the board—over 30 percentage points on some benchmarks—indicating improved token efficiency driven by an enhanced multi-stage data-processing pipeline, with further headroom available from additional data.
- **Gated Transformer Blocks for Stable Training and Efficient Long-Context Inference:** We introduce *Gated Transformer Blocks*, which integrate **Sparse Gated Attention (SGA)** and **Gated Norm (GN)**. SGA combines a lightweight indexer (DeepSeek-AI, 2025b) with gated attention (Qiu et al., 2025b): the sparse attention prunes the attention computation to curb long-context compute, while the head-specific output gate adds non-linearity and suppresses attention sinks, improving loss convergence and attention quality. GN (Qiu et al., 2026) further mitigates outliers in the hidden states—massive activations that inflate per-block scales—while improving the accuracy of low-bit formats such as FP8 and FP4. Together, these designs let A.X K2 extend its usable context length while remaining amenable to low-precision deployment, reducing memory footprint and inference latency on long-context workloads.
- **Think-Fusion Supervised Fine-Tuning (SFT) and Multi-Stage Reinforcement Learning:** Our *Think-Fusion* SFT recipe trains a single unified model on paired thinking and non-thinking responses, so that mode switching is learned from explicit control tokens rather than from superficial distributional cues. The resulting model supports a *non-thinking mode* for concise, low-latency responses and a *thinking mode* for extensive reasoning chains, empowering users to flexibly allocate compute at inference time based on task complexity. Subsequent multi-stage reinforcement learning jointly optimizes instruction following, human-preference alignment, agentic tool use, and safety under a shared reward framework.
- **Frontier-Scale Engineering Capability:** We demonstrate the feasibility of end-to-end engineering for a 688B-parameter model, including stable large-scale training, reproducible pipelines, and system-level optimizations, further establishing a foundation for scaling toward frontier-class model development.

Extensive evaluations show that A.X K2 performs competitively against strong open-weight baselines across English and Korean benchmarks—notably matching or exceeding them on math and Korean tasks—while providing explicit, user-controllable switching between thinking and non-thinking modes at inference time. Beyond benchmarks, our results validate the feasibility of unifying implicit and explicit reasoning processes within a single parameter space. We release A.X K2[1], positioning it as a practical, controllable foundation model.

Organization. The remainder of this report details the model architecture and tokenizer design (Sec. 2), pre-training (Sec. 3), post-training (Sec. 4), and evaluation results (Sec. 5).

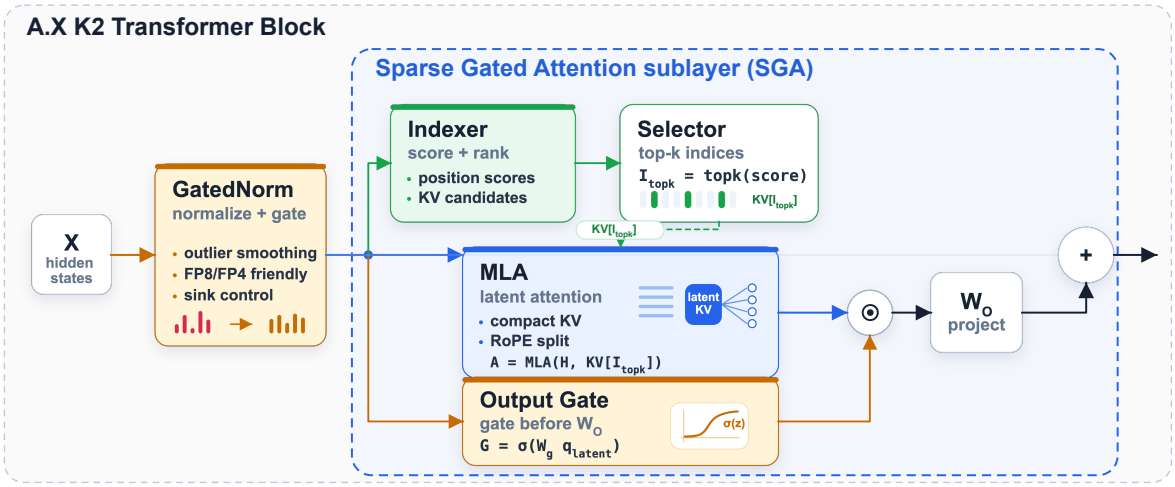


Figure 2: **The A.X K2 transformer block with the SGA sublayer.** A lightweight indexer scores and ranks key–value candidates, and the selector keeps the top-k indices I_{topk}; MLA then performs latent attention over the selected KV cache $\text{KV}[I_{\text{topk}}]$. A head-specific output gate $G = \sigma(W_g\, q_{\text{latent}})$ modulates the attention output before the output projection W_O. GN precedes the sublayer, smoothing outliers for stable, low-precision (FP8/FP4) computation.

Models	Total Params.	Activated Params.	Layers	Heads (Q / KV)	Hidden Size	Intermediate Size (Dense / Expert)	Routed / Activated Experts	Shared Experts	Vocab Size	Context Length (Native / YaRN)
A.X K2	688B	33B	61	64	7168	18432 / 2048	256 / 8	1	163,840	128K / 256K

Table 1: **Architecture overview of A.X K2, an MoE model built from A.X K2 transformer blocks.** The 128K context is trained natively via ABF, and 256K is reached at inference by increasing the YaRN scaling factor (Sec. 3.4).

2 Architecture

2.1 Model Configuration

The A.X K2 model is a large-scale MoE language model with 688B total parameters and 33B active parameters, developed as part of the Korean government's 'Sovereign AI foundation model' project.[2] Detailed architectural specifications are presented in Table 1. For the pre-training compute budget, given a training timeline of approximately 70 days on 512 NVIDIA B200 GPUs, we estimated the total available training FLOPs as a fixed compute budget, under which architectural trade-offs were guided by established scaling principles for MoE models.

Relative to A.X K1, the principal change in model scale is an increase in the number of routed experts from 192 to 256, which raises total capacity while keeping the activated parameter count at 33B. The expert count was set to meet the target total-parameter budget derived from the available FLOPs, and we chose 256—a power of two and a multiple of 128—to align with expert-parallel sharding and kernel tiling for implementation robustness.

The architectural design of A.X K2 adopts several widely used components that have proven effective in large-scale MoE systems (Liu et al., 2024; Yang et al., 2025; GLM-4.5 Team, 2025). The backbone uses Multi-head Latent Attention (MLA) (Liu et al., 2024), which improves Key–Value cache efficiency and reduces memory overhead under long-context settings. On top of MLA, we apply a head-specific output gate (gated attention) (Qiu et al., 2025b) throughout pre-training; the gate introduces non-linearity into the attention output, mitigates attention sinks, and improves loss convergence. For additional training stability, the query and key projections are normalized before the attention scores are computed (QK-normalization) (Dehghani et al., 2023). Of the 61 transformer layers, the first is a dense feed-forward layer and the remaining 60 are MoE layers; the expert-routing and load-balancing strategy is detailed in Section 3.

[1] https://huggingface.co/skt/A.X-K2

[2] The Ministry of Science and ICT (MSIT), Republic of Korea, through the National IT Industry Promotion Agency (NIPA) (Grant No. PJT-26-010018).

To improve training stability and mitigate hidden-state outliers, we adopt Gated Norm (GN) (Qiu et al., 2026), which applies a gating operation immediately after RMSNorm (Zhang & Sennrich, 2019). By suppressing outlier amplification in the normalized output, GN yields smoother loss convergence. As an additional benefit, it produces an activation distribution with suppressed outliers that is favorable for low-precision (e.g., NVFP4) serving. The gating operation of GN incurs an approximately 5% reduction in training throughput, a cost we accepted in exchange for these stability and low-precision benefits; with GN in place, training remained sufficiently stable *without* the dual-normalization scheme used in A.X K1.

Whereas A.X K1 stabilized training with a dual-normalization scheme—inspired by Gemma (Gemma Team, 2025) and early GLM-4 designs that stack normalization layers around the attention and MLP blocks—A.X K2 removes this redundancy entirely. The single post-normalization gating of GN subsumes the stabilizing effect of the earlier dual-normalization design while reducing architectural complexity.

Separately, informed by empirical observations in (Kimi Team, 2025), we configured the model with 64 attention heads and integrated shared dense experts as an operating point that prioritizes inference efficiency by reducing attention overhead while enhancing knowledge sharing.

The tokenizer is inherited unchanged from A.X K1: a byte-level BPE (Sennrich et al., 2016) vocabulary of 163,840 tokens covering English, Korean, Chinese, Japanese, and Spanish.

2.2 Gated Transformer Blocks for Mitigating Attention Sinks and Outliers

Sparse Gated Attention. To make long-context inference efficient while keeping training stable, A.X K2 combines two mechanisms into **SGA** (Figure 2)[3]: gated attention (Qiu et al., 2025b), applied throughout pre-training as described above, and a sparse-attention mechanism added for long contexts. For sparsity, we adopt the sparse-attention mechanism of DeepSeek-AI (2025b), which uses a lightweight indexer to select a small set of top-k tokens ($k = 2048$) per query, substantially reducing attention compute at long sequence lengths. As described in Sec. 3, this sparse component is introduced through a dedicated adaptation stage, in which the indexer is trained with a *sparse warmup and adaptation* after the model has been natively trained to a 128K context, so that long-context quality is preserved while inference cost is reduced. The gating, by contrast, is not specific to long context: it is the head-specific output gate already present in every attention layer during pre-training. The two mechanisms are also mutually reinforcing: because the indexer is trained to reproduce the attention distribution, its top-k quality is bounded by the quality of that distribution. By suppressing the attention-sink mass that standard softmax attention places on a few uninformative tokens (Qiu et al., 2025b), the output gate yields a better-calibrated signal so that the indexer spends its limited budget on genuinely relevant positions. Crucially, introducing the sparse component leaves long-context quality essentially intact: on LongBench (Bai et al., 2024), A.X K2 scores 62.80 before and 62.99 after SGA adaptation, so the efficiency gains below are obtained at no measurable quality cost. In long-context serving, these properties translate into concrete efficiency gains: as the input length grows into the tens of thousands of tokens, A.X K2 with SGA sustains substantially higher total-token throughput, lower time per output token (TPOT), and lower time-to-first-token (TTFT) latency than A.X K1 (SKT, 2026), as shown in Figure 4.

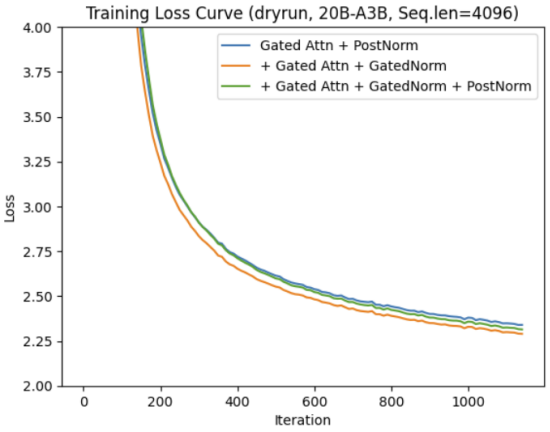


Figure 3: **Training loss for the GN ablation (dryrun, 20B-A3B, sequence length 4096).** Adding GN on top of gated attention improves loss convergence, while further stacking a post-MLP normalization layer yields no additional benefit, indicating that GN alone subsumes the stabilizing effect of the earlier dual-normalization design.

Gated Norm. A.X K2 additionally adopts **GN**, in which a learned, input-dependent gate modulates the normalized activation before it enters the residual stream. Beyond its stability benefits, GN bounds the magnitude of activations propagated across layers: it suppresses *massive activations*, a small number of hidden units that are orders of magnitude larger than the rest and persist across layers, which are a well-documented source of both training instability and low-precision quantization error (Qiu et al., 2025b; 2026). This matters for deployment under narrow block-scaled formats such as NVFP4, where

[3]We use "SGA" for our design; despite the shared name, it is unrelated to the method of Du et al. (2025).

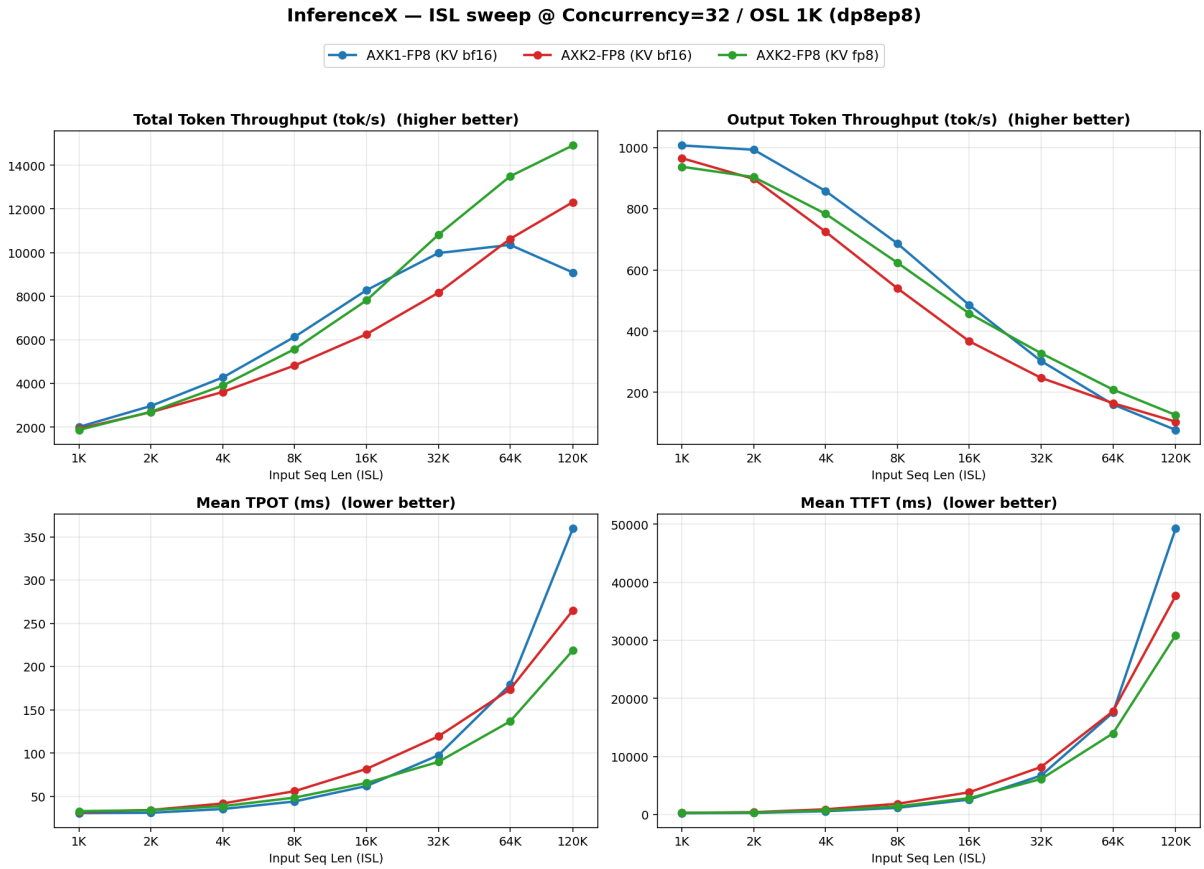


Figure 4: **Inference efficiency of A.X K2 with SGA across input sequence lengths (1K–120K input, 1K output, concurrency 32, dp8/ep8), comparing A.X K1 (FP8) with A.X K2 (FP8) under bf16 and fp8 KV cache. Top:** total and output token throughput (higher is better). **Bottom:** mean per-token latency (TPOT) and time-to-first-token (TTFT) (lower is better), all plotted against input sequence length (ISL). As the input length grows, the sparse-attention indexer and compressed KV cache of SGA deliver higher total-token throughput and lower latency than A.X K1.

4-bit (E2M1) elements share a single scale over a small block (e.g., 16 elements): a lone outlier inflates that block's scale and collapses the precision of all co-located values, an effect further amplified in the MLA latent KV cache through the up-projection. By keeping the per-channel activation distribution well behaved, GN, together with the SGA output gate, tightens the per-block dynamic range that low-precision formats must absorb, improving block-scale utilization and lowering FP4/W4A4 error.

3 Pre-Training

3.1 Pre-Training Dataset

A.X K2 is pre-trained on approximately 8.2 trillion tokens spanning diverse web text, code, STEM (Science, Technology, Engineering, and Mathematics), reasoning, books, and synthetic data. The corpus is built with the multi-stage data-processing pipeline developed for A.X K1 (SKT, 2026), which combines an in-house document-parsing pipeline (a Korean-specialized vision-language model with fine-tuned layout detection that recovers text from PDFs), a dual synthetic-data pipeline (seed-corpus-based reasoning and agentic data, and topic-based knowledge synthesis), and a three-stage curation pipeline: heuristic and model-based quality filtering with deduplication, domain-aware mixture and upsampling, and difficulty scoring for curriculum learning (Bengio et al., 2009). As these components are largely unchanged from A.X K1, we refer the reader to SKT (2026) for their details and focus here on the data updates specific to A.X K2.

For A.X K2, we refreshed and expanded this corpus along several axes. The English web and code corpora were updated to more recent snapshots—principally the Nemotron-CC v2.1 corpus (Su et al., 2025; NVIDIA, 2025a) and its code counterpart, Nemotron-Pretraining-Code-v2 (NVIDIA, 2025b)—and the Korean web corpus was likewise refreshed, while multilingual coverage was broadened by incorporating

	Stage 1	Stage 2 onward
Quality signal	educational value	educational value + difficulty
Domain taxonomy	web domains (26 topical)	expert domains (14 academic)
Per-domain threshold	baseline	raised
Curricular role	broad coverage	high-difficulty concentration

Table 2: **Stage-dependent filtering.** Both the quality signal and the domain taxonomy are refined from Stage 1 to Stage 2, once the corpus has been concentrated on higher-quality data.

FineWeb2 (Penedo et al., 2025). New synthetic data was generated using stronger open-source models alongside in-house models (SKT, 2025). Reflecting the agentic and long-horizon objectives of A.X K2, we substantially increased the proportion of agentic and software-engineering data, including tool-use trajectories and repository-level coding data, as well as high-quality long-context data drawn from web and PDF sources. In the later stages we additionally blended in instruction-style (SFT-format) data to strengthen instruction-following and task competence ahead of post-training. The resulting stage-wise category mixture is summarized in Appendix B.1 and illustrated in Figure 5.

Personal information. Corpora collected in-house pass an internal PII masking pipeline before training. Detected identifiers—primarily email addresses and numeric identifiers such as phone numbers—are replaced with type-specific placeholder tokens (for example `<|email|>`) rather than deleted, so that the surrounding text remains intact and the document stays usable as training signal; detection rules are maintained per identifier type and reviewed for false positives. Publicly released datasets are used as distributed and are not re-processed, since their publishers already apply their own PII handling.

3.2 Quality Filtering and Difficulty-Aware Curation

Although the overall curation framework is inherited from A.X K1, the filtering regime for A.X K2 was substantially revised to operate under a tighter compute budget. The full candidate pool—roughly 16.2 trillion tokens—far exceeded what could be consumed within the project's compute and schedule constraints, so rather than indiscriminately training on all available data, we prioritized the most *training-efficient* subset and arranged it as a difficulty-increasing curriculum, concentrating the highest-value, highest-difficulty data in the later stages of training (Hu et al., 2024; Dubey et al., 2024; Olmo, 2025; Bengio et al., 2009). We assess each document along two axes—quality and domain—and progressively tighten both as training advances.

A stage-dependent filtering scheme. Quality and domain are assessed by progressively stricter criteria across stages, as summarized in Table 2. In Stage 1, quality is measured by an *educational-value* score and documents are partitioned by a coarse *web-domain* classifier into 26 general topical categories.[4] From Stage 2 onward, once the corpus has been concentrated on higher-quality data, quality is measured by educational value *and* difficulty, and documents are re-partitioned by a finer *expert-domain* classifier over 14 academic and professional fields.[5] In both regimes, the quality threshold is set *per domain* rather than globally so that it tracks each domain's distribution, and it is raised in Stage 2.

Both scores are produced by a compact A.X-Encoder–based classifier distilled from a larger teacher: the teacher labels a seed set, the classifier is trained on these labels, and inference over the full corpus yields per-document educational-value and difficulty scores on a 0–4 scale. Restricting the difficulty signal to Stage 2 and beyond induces the difficulty-increasing curriculum, and the per-domain thresholds are deliberately relaxed for fields dominated by long-tail knowledge so as to preserve rare content.

Web corpora and stage-wise consumption. The two principal web sources are an aggregated English web corpus (≈9.1T tokens), of which Nemotron-CC v2.1 (Su et al., 2025; NVIDIA, 2025a) is the largest single contributor alongside several additional English web sources, and an in-house Korean crawl (≈1.37T tokens). Under the scheme above, the English web supplies ≈3.68T unique tokens in Stage 1 and ≈0.22T in Stage 2, and the Korean web supplies ≈1.12T and ≈0.06T, respectively; all per-stage figures are *unique*-token counts, obtained by capping each dataset's epoch multiplicity at one (Table 3).

[4] Arts and entertainment, autos and vehicles, beauty and fitness, books and literature, business and industrial, computers and electronics, finance, food and drink, games, health, hobbies and leisure, home and garden, internet and telecom, jobs and education, law and government, news, online communities, people and society, pets and animals, real estate, science, sensitive subjects, shopping, sports, travel and transportation, and adult content.

[5] Mathematics, physics, chemistry, biology, earth science, manufacturing engineering, other engineering, IT, humanities, social science, law, economics, medicine, and culture.

Source	Corpus	Stage 1	Stage 2
English Web (primarily Nemotron-CC v2.1)	9.1T	3.68T	0.22T
Korean Web (in-house crawl)	1.37T	1.12T	0.06T

Table 3: **Web-corpus sizes and the unique-token volume consumed per stage.** All counts are measured with the A.X K2 tokenizer (Sec. 2) and therefore need not match the token counts published with the source datasets, which use different tokenizers.

3.3 Pre-Training Process

To effectively optimize A.X K2 across varying data distributions, we adopt a Warmup-Stable-Decay (WSD) learning-rate schedule (Hu et al., 2024) across a multi-stage pre-training pipeline (Figure 5). Pre-training is organized into three stages, defined by their training objectives, data characteristics, and corresponding phase of the WSD schedule: *Stage 1* (general knowledge), *Stage 2* (advanced knowledge and reasoning), and *Stage 3* (long-context extension). Stages 1 and 2 are described below, while Stage 3 is detailed separately in Section 3.4.

1. *Stage 1 — General Knowledge (Warmup and Stable Phase):* A.X K2 is trained on a corpus of 6.4 trillion tokens at a sequence length of 4,096, establishing a robust foundation in both linguistic proficiency and general world knowledge. After an initial learning-rate warmup and batch-size ramp-up, the model is trained at the constant maximum learning rate for the remainder of the stage—the stable phase of the WSD schedule—to maximize knowledge acquisition.
2. *Stage 2 — High-Quality Reasoning (Decay Phase):* Using a curated corpus of 1.4 trillion tokens at a sequence length of 4,096, this stage is strictly filtered for high-complexity web data and further incorporates a diverse mixture of domain-specific datasets, including STEM, reasoning-intensive tasks, refined PDF content, and high-quality synthetic data. Here we begin the learning-rate decay, annealing from the maximum learning rate to stabilize the model on these high-quality distributions.
3. *Stage 3 — Long-Context Adaptation:* We progressively extend the usable context length to 32K and then 128K, followed by a sparse-attention adaptation. Owing to its distinct objectives and recipe, this stage is detailed separately in Section 3.4.

3.4 Long-Context Adaptation

Stage 3 enables long-context capabilities through three sub-stages: we natively extend the context length to 32K (Stage 3A, approximately 0.22T tokens) and subsequently to 128K (Stage 3B), followed by sparse-attention adaptation (Stage 3C; described below), rather than relying on post-hoc interpolation. Stages 3B and 3C together consume approximately 0.14T tokens. For the extension, we follow the Adjusted Base Frequency (ABF) approach (Xiong et al., 2023): the RoPE base is kept at its default value of 10^4 throughout Stages 1 and 2 (sequence length 4,096), raised to 10^6 at the start of the 32K extension (Stage 3A), and then held at 10^6 for the 128K extension (Stage 3B) and the subsequent SGA adaptation (Stage 3C), so that the full 128K context is learned natively without any further base adjustment. In our experiments, this native base adjustment yielded stronger long-context retrieval than relying on RoPE interpolation alone. This design is motivated by the demands of agentic tasks, which require long-horizon reasoning, planning, and action over extended contexts; accordingly, the long-context stages incorporate data featuring long chains of thought and repository-level code.

For RoPE scaling, we use YaRN (Peng et al., 2023), which can be applied in two modes. A *static* mode uses a single fixed scaling factor across the entire sequence, which favors fast convergence during training (as adopted by DeepSeek and Kimi-K2 (Liu et al., 2024; Kimi Team, 2025)); a *dynamic* mode instead leaves positions within the originally trained length unscaled and interpolates only beyond it, recomputing the factor at inference time from the actual sequence length to minimize extrapolation error (as adopted by Qwen (Yang et al., 2025)). We keep the YaRN attention-temperature scaling factor (*mscale*) at 1.0, applying no additional rescaling to the attention logits. Because A.X K2 reaches 128K natively through ABF, longer contexts are supported by increasing the YaRN scaling factor, with the static or dynamic mode chosen by the serving stack. With an enlarged scaling factor—and with the RoPE base itself left unchanged at 10^6—A.X K2 extends to a 256K context and attains strong needle-in-a-haystack (NIAH) retrieval at this length, as shown in Fig. 6.

Finally, after reaching 128K, we apply a dedicated adaptation stage (Stage 3C) that introduces the sparse-attention component of SGA (Sec. 2.2) to improve long-context inference efficiency. Stage 3C reuses the

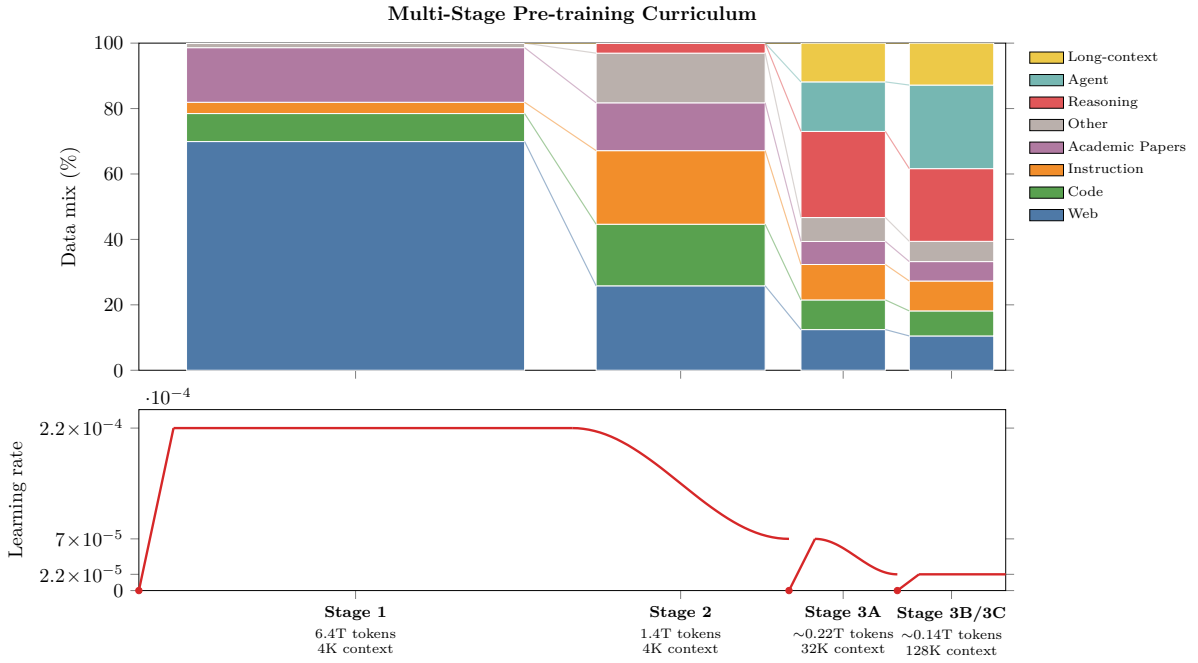


Figure 5: **Multi-stage pre-training curriculum of A.X K2. Top:** the stage-wise shift in data-mixture composition (%); earlier-introduced sources sit at the bottom of each bar and newly-introduced data (reasoning, agent, long-context) on top, with light connectors tracing each category across stages. **Bottom:** the Warmup–Stable–Decay (WSD) learning-rate schedule, where each of the three warm-ups (Stage 1, 3A, 3B) rises from zero and is followed by a cosine decay. Token counts and context lengths are annotated per stage; stage widths are illustrative and not to token scale.

same data mixture as Stage 3B—the two share a mixture, which differs from that of Stage 3A—so that introducing SGA does not require switching to a separate dataset, following the approach proposed in GLM-5 (GLM-5 Team, 2026). The SGA indexer, which selects the top 2,048 tokens per query, is trained by a KL-divergence loss that aligns its selection scores with the attention distribution. We begin Stage 3C with an indexer-only warmup, freezing all non-indexer parameters so that the indexer adapts without perturbing the already-converged backbone. Crucially, unlike DeepSeek-V3.2 (DeepSeek-AI, 2025b) and GLM-5 (GLM-5 Team, 2026), which first warm up the indexer against the *dense* (full) attention distribution before enabling sparse selection, we compute the indexer loss directly over the sparse top-k selection from the outset—i.e., a *sparse* warmup. After this warmup, the full model resumes training with the indexer loss down-weighted, producing the final sparse-attention model. Because the warmup runs under sparse attention rather than full dense attention, it is substantially faster than a dense warmup; in our experiments, this yielded a large speedup while incurring negligible difference in downstream performance, making sparse warmup the more cost-effective choice.

3.5 Checkpoint Merging

As the final step of pre-training, we apply checkpoint merging—a Warmup-Stable-Merge (WSM) style of Stochastic Weight Averaging (SWA) (Izmailov et al., 2018)—to produce the released base model. Rather than committing to a single final checkpoint, we average the parameters of the last six checkpoints, spaced approximately 1.6B tokens apart (25 optimizer steps of ≈64M tokens each), directly in weight space. Averaging over nearby points along the optimization trajectory reduces the variance of the final solution and tends to settle in a flatter region of the loss landscape, improving robustness and generalization at no additional training cost. The merge is computed shard-wise over all model tensors, including the MoE router weights, which we found beneficial to average rather than exclude. We use the $1 - \sqrt{\cdot}$ weighting scheme, which assigns larger weight to the more recent checkpoints, so that the merged model stays close to the latest, best-converged checkpoint while still benefiting from averaging; uniform and exponential-moving-average weightings are also supported. Because the averaged checkpoints span only a narrow window (roughly 8B tokens in total) and therefore lie close together in parameter space, the merge incurs negligible cost while consistently yielding a stronger final model than any individual checkpoint in the window.

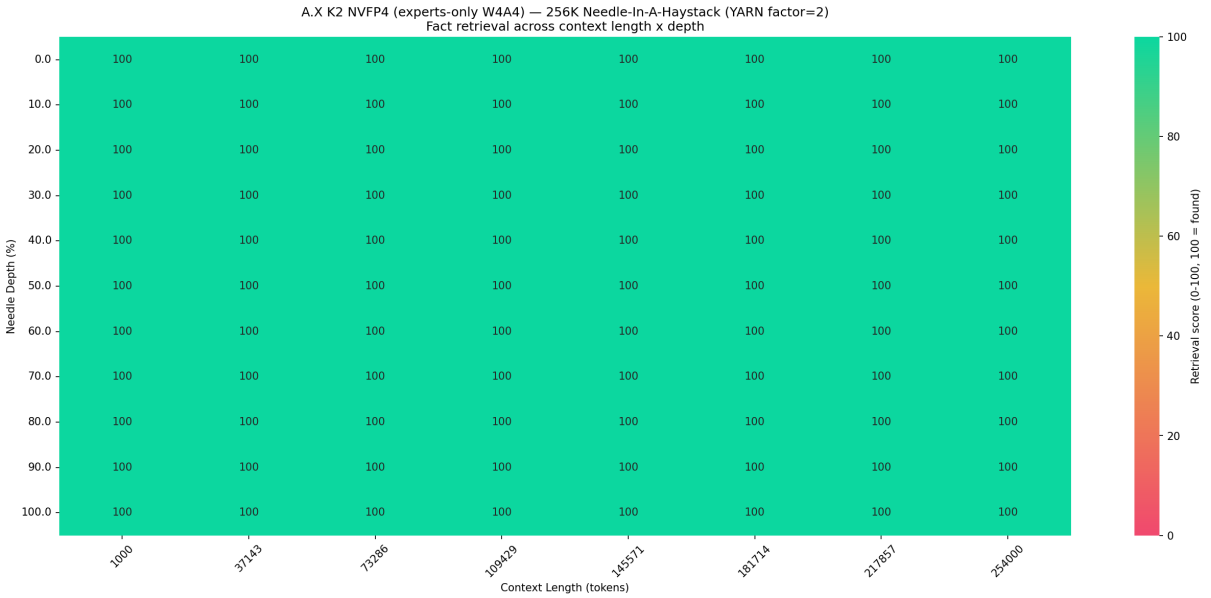


Figure 6: **Needle-in-a-haystack (NIAH) retrieval for A.X K2 at a 256K context length, obtained by increasing the YaRN scaling factor.** Each cell reports the retrieval score (0–100, 100 = found) across context length (x-axis) and needle depth (y-axis); A.X K2 attains a perfect score of 100 at every position, including under NVFP4 experts-only W4A4 quantization.

3.6 Hyperparameters

The magnitudes of the learning rate and batch size were selected based on the MoE scaling laws proposed by Tian et al. (2025), which were introduced earlier in the context of model size and architecture design. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$), following GPT-3 (Brown et al., 2020). The peak learning rate was set to 2.2×10^{-4} and the global batch size to 16,384 sequences (with a micro-batch size of 4), corresponding to approximately 64M tokens per step at a sequence length of 4,096. Following the WSD schedule, the learning rate is linearly warmed up over the first ∼6.1M sequences (≈25B tokens) and then held constant at 2.2×10^{-4} through the general stage; it is subsequently cosine-decayed to 7×10^{-5} across the reasoning stage. Each long-context extension stage then re-warms the learning rate at its start—over roughly 0.3M sequences for the 32K stage and 0.08M sequences for the 128K stage—to stabilize optimization as the sequence length and data distribution change: the 32K stage warms up to 7×10^{-5} and cosine-decays to 2.2×10^{-5}, and the 128K stage warms up to and then holds 2.2×10^{-5}, a level maintained through the SGA-adaptation stage.

To mitigate gradient spikes and ensure stability during the early training phase, a global batch-size ramp-up was applied, linearly increasing the batch size from 2,048 to 16,384 in increments of 2,048 over the first 72 million sequences (approximately 295B tokens at a sequence length of 4,096). As the sequence length grows during long-context training, the number of sequences per step is reduced accordingly (for example, to 2,048 sequences at a 32K length) so that the tokens processed per step remain close to 64M throughout.

In addition to the general optimization hyperparameters, several MoE-specific settings were configured to balance routing stability and expert utilization. Tokens are routed by a sigmoid-scored, pre-softmax router using grouped top-k selection: the 256 routed experts are partitioned into 8 groups, and each token activates the top 8 experts under a group top-k of 4, with the routing logits moderated by a top-k scaling factor of 2.5. A.X K1 used a router-bias mechanism together with a sequence-level auxiliary loss (SKT, 2026; Liu et al., 2024). In contrast, A.X K2 adopts the global auxiliary loss of Qiu et al. (2025a). A sequence-level loss enforces expert balance within each individual sequence—an overly local constraint that suppresses expert specialization and can degrade quality—whereas computing the balancing loss over a larger, global batch relaxes this constraint while still equalizing expert utilization. The global auxiliary loss alone, however, did not sufficiently reduce the maximal violation (MaxVio) of expert load (Wang et al., 2024)—the worst-case relative deviation of an expert's load from the uniform target—so we additionally retain a learnable expert-bias term (whose update rate adapts across stages), measuring MaxVio over the global batch rather than the per-microbatch quantity used in the original formulation.

The load-balancing strength is annealed as training progresses. During the general and reasoning stages,

the (global, sequence-level) auxiliary-loss coefficients are set to $(4 \times 10^{-3}, \text{off})$—the sequence-level term is only monitored rather than applied—and the expert-bias update rate is 1×10^{-3} to facilitate adaptation of expert selection. For the long-context extension stages, the coefficients are tightened to $(1 \times 10^{-3}, 1 \times 10^{-5})$ and the bias update rate is lowered to 1×10^{-4} to stabilize routing as the data distribution shifts toward higher-quality and long-context data. During the final SGA adaptation stage, the coefficients are further reduced to $(5 \times 10^{-4}, 2 \times 10^{-5})$; the expert bias is briefly frozen (update rate 0) while the sparse-attention indexer—which selects the top 2,048 tokens per query—is warmed up, and is then resumed at 1×10^{-4}.

3.7 Parallelism and Training Efficiency

Pre-training was conducted on 512 NVIDIA B200 GPUs (64 nodes of 8 GPUs) over approximately 70 days. We used pipeline parallelism (PP) of 8 with an interleaved virtual-pipeline degree of 2, and expert parallelism (EP) of 8, with the remaining ranks assigned to data parallelism; tensor parallelism was not used. The larger per-device memory of the B200—relative to the H200 used for A.X K1—allowed us to reduce pipeline parallelism from 16 to 8 while still holding the model without tensor parallelism, thereby lowering pipeline-bubble overhead. We employed a distributed optimizer, which partitions optimizer states across data-parallel ranks and achieves memory savings comparable to ZeRO Stage 2 (Rajbhandari et al., 2020). For expert parallelism, token dispatch and combine were handled by the HybridEP all-to-all backend rather than DeepEP, with expert-parallel communication confined to the eight-GPU intra-node NVLink domain and a small number of streaming multiprocessors (24–32) dedicated to the dispatch/combine kernels. These settings were chosen on the basis of large-scale profiling to preserve the intended total training compute within the project timeline.

Context parallelism was employed to support long-context training: a context-parallel degree of 2 was used for the 32K extension stage and a degree of 8 for the 128K stage, distributing attention computation across devices while keeping per-GPU memory usage within practical limits.

To manage activation memory pressure, A.X K2 uses *full* activation recomputation (Korthikanti et al., 2022), applied uniformly with each transformer layer forming its own recomputation unit. Although full recomputation incurs additional recompute cost relative to the selective scheme used in A.X K1 (SKT, 2026), the memory it frees allowed us to raise the micro-batch size to 4; in our setup this trade-off—full recomputation with a larger micro-batch—delivered higher per-GPU throughput than selective recomputation with a smaller micro-batch, by better amortizing kernel-launch and pipeline overheads across more samples. At the longer sequence lengths of the long-context stages (Stage 3A onward), however, even full recomputation left limited memory headroom, so the micro-batch size was reduced to 1.

We trained in MXFP8, a Microscaling (MX) FP8 format (Rouhani et al., 2023) in which each small block of consecutive elements (32 values) shares a single scaling factor, rather than applying one scale across an entire tensor as in conventional FP8 (Micikevicius et al., 2022). This finer-grained, block-wise scaling better accommodates outliers in activations and gradients while retaining the throughput and memory benefits of 8-bit storage. We used the E4M3 format for both the forward and backward passes. To preserve numerical stability, the main model parameters and gradients were maintained in FP32, while the exponential moving averages and squared gradients used by the optimizer were stored in BF16. All remaining eligible tensors were represented in MXFP8. We further gather the distributed-optimizer parameter shards directly in FP8 and overlap this all-gather with computation: transmitting parameters in FP8 reduces the inter-GPU communication volume and ensures that the same low-precision weights are used in the gather and in the subsequent matrix multiplications—which aids the numerical stability of FP8 training—while overlapping the gather with compute hides its latency and improves throughput. The outlier-suppressed activation distribution induced by GN (Sec. 2) further supports stable low-precision training and is favorable for FP8 serving.

While training uses the MX-style block size of 32 elements, the released checkpoint is distributed in a *blockwise* FP8 (E4M3) format with 128×128 weight blocks and dynamic per-token activation scaling, matching the recipe supported by mainstream FP8 serving kernels (Zhao et al., 2025). The model therefore serves in FP8 out of the box, with no separate post-hoc quantization step, and the same blockwise recipe is reused for RL rollouts (Sec. 4).

4 Post-Training

The post-training pipeline consists of two SFT stages followed by multi-stage Reinforcement Learning (RL). The SFT phase comprises an indexer SFT stage and a main SFT stage. Before the main post-training stages, we conduct a short indexer SFT, in which both the model parameters and the sparse-attention

indexer parameters are trained. In subsequent stages, the indexer parameters are kept fixed and only the model parameters are updated.

4.1 Supervised Fine-Tuning

Like its predecessor (SKT, 2026), A.X K2 supports user-controllable hybrid inference, in which the thinking and non-thinking modes are explicitly specified at inference time. When thinking mode is enabled, the response follows the format `<think>...</think>{response}`, following (DeepSeek-AI, 2025a; Yang et al., 2025). When non-thinking mode is used, the response follows the format `</think>{response}`.

We observe that jointly training on thinking and non-thinking data can cause confusion between the two modes. In particular, thinking data, which is characterized by longer sample lengths, can dominate training and lead the model to generate thinking-mode responses even when non-thinking mode is requested. Prior work has mitigated this issue by adopting a dual-track training strategy, as in our previous work (SKT, 2026), or by adjusting the token ratio of reasoning to non-reasoning data to 1.5:1 (Bae et al., 2025). In this work, we instead unify training into a single track—our *Think-Fusion* SFT recipe—and achieve reliable mode switching through data engineering. We construct a paired SFT dataset in which each prompt is associated with both a thinking and a non-thinking response. This prevents the model from relying on superficial distributional differences between thinking and non-thinking data and encourages it to follow the explicit mode instruction. We train A.X K2 on the paired SFT dataset, in which the thinking-to-non-thinking token ratio is approximately 13:1, and track mode confusion throughout post-training to verify that explicit control tokens remain effective after RL.

4.1.1 Data

The SFT dataset is organized into paired reasoning (thinking) and instruction (non-thinking) examples. The reasoning split emphasizes tasks that benefit from explicit intermediate reasoning, including mathematics, knowledge-intensive science questions, code, and agentic tool-use trajectories. The instruction split emphasizes concise instruction following, general chat, safety, and non-thinking counterparts for selected reasoning prompts. This paired construction lets the model learn mode control from the control tokens rather than from category-specific artifacts.

For safety data, A.X K1 already benefited from the safety SFT corpus introduced by ESSA (Park et al., 2026), which frames safety alignment as specification-guided deliberation over domain–task-conditioned safety requirements. In A.X K2, we further extend this source rather than simply reusing it: we run additional rounds of safety-specification evolution, label a larger English–Korean prompt pool with domain and task metadata, and synthesize new safety reasoning examples conditioned on the selected safety specification. This yields safety data that is better matched to the A.X K2 chat template and post-training mixture, while preserving the core ESSA objective of reducing both unsafe compliance and unnecessary over-refusal.

Concretely, each safety prompt is first mapped to one of 50 domain–task combinations, covering five application domains and ten response-task types. The resulting label is used only to select the corresponding evolved safety specification; the synthesis prompt itself exposes the selected specification and the user request, but not the internal domain–task label. The generated sample is then converted into the same chat-sample format used by the rest of the SFT pipeline, with the reasoning portion serialized for thinking-mode supervision and the final answer written as a natural user-facing response. We apply rubric-based filtering and targeted meta-review to remove samples with unsafe residue, language mismatch, formatting leakage, or excessive refusal, and to prefer safe completions that remain useful when a benign objective can be preserved.

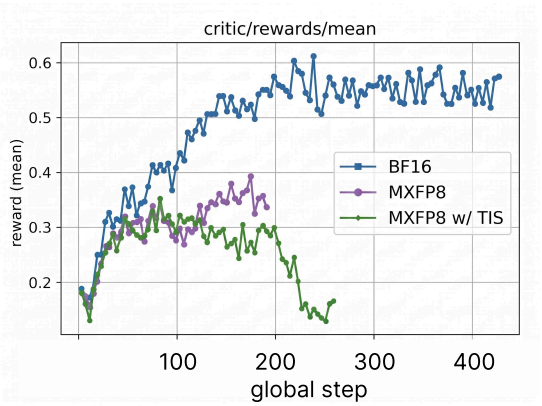
Table 4 shows the number of samples and ratio of the SFT dataset per category. The number of tokens and the distribution of token length are also provided in Appendix B.2.

4.1.2 Training

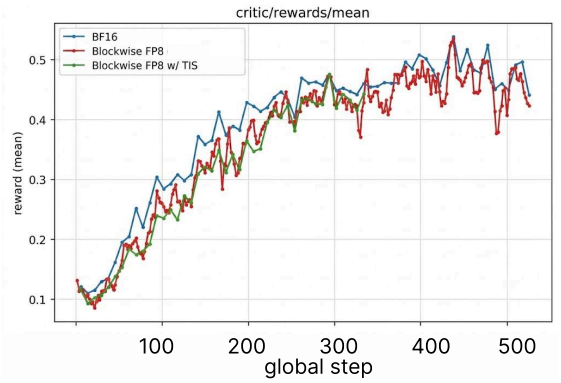
The SFT phase consists of two stages. First, we sample 0.4M examples without replacement from the SFT dataset and train the model together with the indexer so that the indexer learns the special control tokens. We then fix the indexer parameters and conduct the main SFT stage. For both stages, we train on packed 128K-token sequences, matching the model context length. For efficient data packing, we use a length-aware First-Fit Decreasing (FFD) bin-packing algorithm. We sort all samples in a buffer in descending order of length and place each sample into the first bin that can accommodate it, opening a new bin only when none is available. By placing the largest samples first and filling the remaining gaps with smaller ones, this approach reduces unused space within each bin. As a result, the packing becomes more compact, yielding a higher effective-token ratio and thereby reducing padding waste.

Stage	Category	Reasoning		Instruction		Total	
		Samples	Ratio (%)	Samples	Ratio (%)	Samples	Ratio (%)
Indexer SFT	Math	46,514	0.51	2,884	0.03	49,398	0.55
	Knowledge / Science	89,086	0.98	4,747	0.05	93,833	1.04
	Instruction Following	5,275	0.06	621	0.01	5,896	0.07
	Code	39,663	0.44	0	0.00	39,663	0.44
	Agent	105,415	1.16	2,690	0.03	108,105	1.19
	General	73,061	0.81	64,317	0.71	137,378	1.52
	Safety	0	0.00	3,613	0.04	3,613	0.04
	Sum	359,014	3.97	78,872	0.87	437,886	4.84
Main SFT	Math	585,678	6.47	90,247	1.00	675,925	7.47
	Knowledge / Science	1,355,439	14.98	148,375	1.64	1,503,814	16.62
	Instruction Following	500,510	5.53	8,122	0.09	508,632	5.62
	Code	496,978	5.49	65,144	0.72	562,122	6.21
	Agent	1,602,902	17.71	35,993	0.40	1,638,895	18.11
	General	1,750,642	19.34	1,892,501	20.91	3,643,143	40.26
	Safety	34,083	0.38	45,164	0.50	79,247	0.88
	Sum	6,326,232	69.91	2,285,546	25.26	8,611,778	95.16
Total		6,685,246	73.87	2,364,418	26.13	9,049,664	100.00

Table 4: **Data composition by sample count and ratio (%) across the SFT stages.**



(a) Training collapse under trainer–rollout precision mismatch.



(b) Stable training when trainer and rollout precision are matched.

Figure 7: **FP8 training stability depends on the trainer–rollout quantization consistency.** In all runs, the rollout engine, implemented with vLLM, uses the same blockwise FP8 format; only the trainer precision is varied. **Left:** When the trainer's precision is inconsistent with the rollout-side blockwise FP8 execution, reward learning becomes unstable and eventually collapses. Applying TIS does not prevent this collapse, indicating that token-level intervention alone cannot remove the underlying trainer–rollout precision mismatch. **Right:** When the trainer precision is aligned with the rollout-side blockwise FP8 format, training remains stable and avoids the collapse observed in the mismatched setting. These results suggest that stable low-precision RL requires matching the trainer's precision to the effective rollout precision.

For both SFT stages, we minimize the standard negative log-likelihood loss. We use an indexer KL-loss coefficient of 0.1 during indexer SFT and set it to 0 thereafter. For indexer training, we use a global batch size of 512 and a cosine learning-rate schedule with a peak of 1.1×10^{-5} and a minimum of 2.2×10^{-6}, warming up for approximately 5% of the total iterations. For main SFT, we reduce the global batch size to 128 and use a learning rate with a peak of 1.5×10^{-5} and a minimum of 3.0×10^{-6}, with 2% warmup iterations. As in pre-training, we use the global auxiliary loss with expert bias for expert load balancing and keep the hyperparameters unchanged. During training, we periodically evaluate performance and supplement the data accordingly. Table 4 presents the full sample counts used for the SFT stages.

Parallel Configuration	Relative Throughput (Output tokens per second)
TP8, DP1	1.0×
TP1, DP8	**1.6**×

Table 5: **Relative throughput comparison between tensor-parallel and data-parallel configurations.** Throughput is normalized to the TP8, DP1 setting, showing that TP1, DP8 achieves 1.6× higher throughput under the same experimental setup (input length = 131,072 tokens).

4.2 Reinforcement Learning Infrastructure

Scaling RL to a model of A.X K2's size exposes two systems-level requirements that the architecture itself does not address: the training and rollout engines must agree numerically under low precision, and the rollout stage, which dominates wall-clock cost for a large Mixture-of-Experts (MoE) model, must be parallelized to match. We address these with an end-to-end blockwise-FP8 recipe and a data-parallel, expert-parallel asynchronous rollout pipeline, described below.

Blockwise FP8 for Trainer–Rollout Consistency. RL requires the training backend (Transformer Engine) and the rollout engine (vLLM) to operate under a numerically consistent FP8 recipe: when the two quantization schemes diverge, the sampling and update distributions drift apart, inflating the policy KL and destabilizing learning. While Blackwell (B200/B300) natively supports the microscaling MXFP8 format, the vLLM rollout backend does not provide first-class MXFP8 support; its mature FP8 path for MoE modules instead targets the *blockwise* FP8 recipe developed for Hopper, with per-block scaling and FP32 scale factors, as used by DeepGEMM (Zhao et al., 2025). To keep the trainer and rollout on a single recipe, we standardize on blockwise FP8 end-to-end. Because the Blackwell Transformer Engine defaults to MX-style scaling, we enable blockwise FP8 on Blackwell through a patched TE branch that forces FP32 block scales. Figure 7 contrasts the two configurations: a trainer running MXFP8 against a blockwise-FP8 rollout exhibits reward collapse, whereas the consistent blockwise-FP8 end-to-end recipe trains stably to convergence.

Data-Parallel Asynchronous Rollout. Large-scale RL on an MoE model is dominated by rollout (generation) cost, so the rollout engine must be parallelized efficiently. The stock veRL pipeline (Sheng et al., 2025) is oriented toward small-model research and supports only single-replica rollout (DP1), which is adequate for dense models but leaves a large MoE such as A.X K2 severely throughput-bound: a single tensor-parallel replica serializes generation behind per-layer tensor-parallel all-reduces and underutilizes the cluster during the generation-heavy phase of each step. We instead develop a fully asynchronous rollout pipeline that shifts attention parallelism from tensor parallelism to data parallelism: rather than the trainer-style TP8/DP1 layout, the rollout engine runs TP1/DP8 while keeping expert parallelism fixed (EP8). This layout (i) maximizes MoE rollout throughput by replicating attention across eight data-parallel replicas and eliminating per-layer tensor-parallel all-reduces, (ii) keeps expert sharding identical to the trainer (EP8) so that policy weights can be resharded into the rollout engine cleanly, and (iii) decouples generation from optimization, allowing rollout and training to overlap. Together, these changes scale the rollout stage from the small-model regime to full-size MoE RL while preserving trainer–rollout consistency.

4.3 Multi-Stage Reinforcement Learning

We conduct RL over multiple stages, keeping the training setup fixed while varying the data composition across stages. Rather than dedicating each stage to a distinct capability, every stage jointly optimizes instruction following, human-preference alignment, and agentic tool use under a shared reward framework; later stages additionally incorporate safety training. We target these four capabilities because, unlike math and coding, they are particularly well suited to on-policy RL, which can directly optimize behaviors that emerge during model generation. We also found that training a single capability with RL in isolation often led to over-optimization: the policy learned to exploit capability-specific reward signals while regressing on behaviors not represented in that stage.

Data and reward design. The training mixture is organized into four groups—instruction following, human preference, agentic tool use, and safety. Rather than fixing the mixture in advance, we treat it as a control surface across stages: using intermediate RL checkpoints, we repeatedly identified the model's weakest behaviors, synthesized targeted data to address them, and re-weighted or replaced datasets at each stage boundary. Thinking and non-thinking modes are sampled in roughly equal proportion throughout.

- **Instruction following** uses rule-based verifiable rewards that check explicit constraints: format, length, required expressions, and schema conformance for structured-output prompts. We apply difficulty filtering to this group with an in-house small model, discarding prompts that the model already solves reliably so that the on-policy signal concentrates on informative cases.
- **Human preference** is scored by a pointwise LLM-as-judge: for each prompt we obtain a reference answer from a strong external model and supply the judge with few-shot pointwise scoring anchors. The judge first classifies the task into one of six domains (factual, reasoning, coding, extraction, creative, open) and then applies a domain-specific rubric over four axes (accuracy, completeness, clarity, helpfulness), with explicit safeguards against verbosity bias and reward hacking.
- **Agentic tool use** is rewarded in two complementary ways. Single-step tool-call data uses verifiable rewards that compare emitted calls against reference calls for schema conformance and argument correctness. Long-horizon agentic tasks use a gated judge: a binary gate first checks that the response constitutes a structurally valid, executable tool-calling transcript, assigning the minimum reward on failure, and only gate-passing responses are forwarded to the reference-based pointwise judge described above—preventing the judge from rewarding fluent responses that never issue a valid tool call.
- **Safety** data comprises red-teaming prompts, safety-preference data, and model-identity data. Red-teaming responses are scored by an LLM judge along nine safety dimensions—covering harm categories such as insults, adult content, illegal activity, hate, and bias, as well as informativeness and comprehension—with a holistic harmfulness rating determining the reward, so that the policy is rewarded for safe completions and penalized for harmful ones rather than merely for refusing. Safety-preference data is judged against principle-based rubrics, using the dataset's preferred and rejected responses as calibration references, and identity data is judged against rubrics encoding the model's identity policy.

In addition, for datasets prone to excessive verbosity, we apply the group-relative length penalty (He et al., 2025). Within each rollout group, shorter correct responses are rewarded and longer ones penalized, while incorrect responses are never rewarded for brevity.

Algorithm and hyperparameters. All stages share a single optimization recipe. We optimize with CISPO (MiniMax, 2025), which clips the importance-sampling weight rather than the token-level update, so that low-probability but behavior-critical tokens continue to contribute gradient. Alongside the task reward, we use an auxiliary format reward and adopt GDPO (Liu et al., 2026), which normalizes each reward separately before combining them, preserving the resolution of each signal and improving multi-reward training stability. We do not apply a KL penalty and sample 16 rollouts per prompt at temperature 1.0, with a global batch size of 80 prompts and a learning rate of 2×10^{-6} held constant after a brief warmup. In total, RL consumes roughly 100K prompts across all stages. Throughout, we retain the AdamW configuration used in pre-training but reduce the optimizer epsilon from 10^{-8} to 10^{-15}, following the finding of MiniMax-M1 (MiniMax, 2025) that a much smaller epsilon stabilizes optimization under the small gradient magnitudes characteristic of RL.

5 Evaluation

5.1 Evaluation Settings

We evaluate A.X K2—alongside its predecessor A.X K1 and leading open-weight baselines—on representative public benchmarks widely used in recent large language model evaluations, with particular attention to Korean-language tasks that are underrepresented in existing evaluations. All models are evaluated in thinking mode; A.X K2 additionally supports a non-thinking mode through its Think-Fusion recipe (Sec. 4).

The evaluation is organized into six categories: Math, Korean, Code, Science & Knowledge, General, and Agentic, each targeting a distinct aspect of model capability. Benchmarks marked with an asterisk (*) are evaluated by Artificial Analysis. Detailed descriptions of each benchmark and the evaluation method are provided in Appendices B.4 and B.5.

5.2 Evaluation Results

To validate the effectiveness of A.X K2, we conducted a comprehensive evaluation across the six categories of Math, Korean, Code, Science & Knowledge, General, and Agentic capabilities. We compared our model against its predecessor A.X K1 and strong open-weight models—Qwen3.5-397B (Qwen Team, 2026a),

Domain	Benchmark	A.X K2 688B-A33B	A.X K1 519B-A33B	Qwen3.5 397B-A17B	Nemotron 3 Ultra	DeepSeek-V4 Flash	GLM-5.1	Kimi-K2.6	MiniMax M2.7
Math	AIME26	**97.1**	91.3	92.5	85.4	96.7	95.4	95.4	88.8
	Apex	**45.8**	1.0	13.5	3.1	28.1	10.4	20.8	1.0
	Apex-shortlist	**88.6**	50.8	61.2	45.5	88.0	72.1	72.6	30.3
	KMO26 (1st round)	92.5	85.0	78.8	85.0	**94.4**	78.1	88.1	80.0
Korean	KMMLU-Pro	**80.5**	68.9	78.4	69.2	78.8	76.5	73.4	66.6
	KoBALT	73.0	51.0	69.9	59.1	**75.3**	72.0	66.0	51.4
	CLIcK	**91.6**	85.3	88.4	84.1	89.6	88.8	80.9	76.1
Code	LiveCodeBench v6 (Feb–May)	84.0	74.9	82.6	76.4	**89.4**	86.4	86.5	69.8
	SciCode	41.0	24.1	42.0	39.9	44.9	43.8	**53.5**	47.0
Science & Knowledge	Humanity's Last Exam	27.8	8.3	27.3	26.6	32.1	28.0	**35.9**	28.1
	GPQA Diamond	85.6	76.4	89.3	86.7	89.4	86.8	**91.1**	87.4
General	IFBench	75.9	63.2	79.0	78.9	**81.2**	78.6	74.0	73.2
	AA-LCR	66.0	26.0	66.0	67.0	63.0	62.0	**70.0**	69.0
Agentic	τ^2-Bench* (Telecom)	**98.0**	86.0	95.6	83.3	95.0	97.7	95.9	84.8
	BrowseComp (≤10 searches)	9.3	–	26.9	13.4	16.8	**29.1**	21.5	14.2

Table 6: **Comparison with strong open-weight LLMs.** The best score in each row is shown in bold. For benchmarks marked with *, the scores of the other open-weight models are taken from Artificial Analysis. All benchmark scores are reported as percentages.

Model	4K	8K	16K	32K	64K	128K	256K	Overall
A.X K2	97.5	97.2	97.5	96.5	94.3	92.7	86.6	94.6

Table 7: **Long-context performance on RULER across sequence lengths.**

Nemotron 3 Ultra (NVIDIA, 2026), DeepSeek-V4 Flash (DeepSeek-AI, 2026), GLM-5.1 (Z.ai, 2026), Kimi-K2.6 (Kimi Team, 2026), and MiniMax-M2.7 (MiniMax, 2026)—as shown in Table 6.

A.X K2 is strongest in Math and Korean. In the Math domain, it achieves the best scores among all compared models on AIME26, Apex, and Apex-shortlist. Beyond final-answer benchmarks, we also tested A.X K2's proof-writing capability on IMO 2025 and the KMO26 2nd round, using the iterative proof refinement method of DeepSeekMath-V2 (DeepSeek-AI, 2025). A.X K2 scored 35/42 on IMO 2025 with a perfect 7/7 on each of the first five problems—reaching the gold-medal threshold of 35—and produced correct proofs for all 8 problems of the KMO26 2nd round. In the Korean domain, A.X K2 leads on KMMLU-Pro and CLIcK, and remains competitive on KoBALT, confirming its strength in the target language context. These results also reflect substantial gains over A.X K1 across every category, most notably on Apex (1.0 $\rightarrow$ 45.8), Humanity's Last Exam (8.3 $\rightarrow$ 27.8), and AA-LCR (26.0 $\rightarrow$ 66.0).

In the Science & Knowledge and General categories, A.X K2 performs on par with strong open-weight models: it matches the field on Humanity's Last Exam and AA-LCR, while GPQA Diamond and IFBench show a modest gap to the best baselines. In the Agentic category, despite only limited agentic reinforcement learning during post-training, A.X K2 attains a moderate level of agentic capability: it achieves the best score among all compared models on τ^2-Bench Telecom. The remaining gap on BrowseComp relative to models trained with heavier agentic optimization indicates clear headroom for future agentic post-training.

5.3 Long-Context Evaluation

Beyond the 256K needle-in-a-haystack (NIAH) retrieval reported in Section 3 (Figure 6), we evaluate long-context understanding on the RULER benchmark (Hsieh et al., 2024), which spans retrieval, multi-hop tracing, aggregation, and question-answering tasks across sequence lengths. As summarized in Table 7, A.X K2 sustains strong performance out to 256K, achieving an overall average of 94.6. Under zero-shot YaRN scaling (scaling factors of 1, 2, and 4 for 128K, 256K, and 512K, respectively), A.X K2 additionally attains a perfect NIAH retrieval score at all three lengths, and does so even after NVFP4 quantization.

5.4 Low-Precision Robustness

The outlier suppression provided by GN and the SGA output gate (Section 2) makes A.X K2 robust under low-precision deployment. Training in FP8 reduces the minimum serving-memory footprint to about 62% of A.X K1's BF16 requirement, and NVFP4 quantization further reduces it to roughly 57% of the FP8 model—about 36% of A.X K1 (Table 8).

Crucially, this compression incurs almost no quality loss. Table 9 compares FP8 and NVFP4 across representative Korean and English base-model tasks; NVFP4 tracks FP8 within roughly one point on nearly all benchmarks. On dedicated NPU hardware, A.X K2 further attains a 107% performance-per-watt

Model	Precision	Min. Memory
A.X K1	BF16	1038 GB
A.X K2	FP8	646 GB
A.X K2	NVFP4	370 GB

Table 8: **Minimum serving memory by model precision.**

Precision	CLIcK	GSM8K	MMLU	MMLU -Pro	KMMLU	KoBEST BoolQ	KoBEST COPA	MATH	MBPP	Human Eval	KLUE -MRC
FP8	84.21	80.21	82.27	69.71	78.41	96.72	88.30	60.50	72.00	79.88	76.20
NVFP4	84.06	77.71	82.00	68.04	77.70	96.01	88.60	58.08	70.40	81.10	76.40

Table 9: **Base-model task accuracy under FP8 vs. NVFP4 quantization.**

ratio relative to a comparable GPU (NVIDIA L40S), measured on Rebellions ATOM-Max.

6 Limitations

While A.X K2 demonstrates competitive performance and validates our efficient training and post-training recipe, several limitations remain, largely driven by real-world constraints and stability-focused engineering trade-offs. These limitations also delineate a clear roadmap for future iterations.

- **Infrastructure Constraints on Scaling:** Our training environment imposed system-level constraints on parallelism. In particular, we confined expert parallelism to the eight-GPU intra-node NVLink domain ($EP = 8$) to keep expert-parallel communication off slower inter-node links. This limited our exploration of higher-degree, multi-node expert parallelism, which could enable finer-grained expert partitioning; realizing it efficiently will require tighter integration between expert-parallel communication and the inter-node interconnect.
- **Optimization under Resource Constraints:** To maximize attainable performance under fixed time and GPU budget constraints, we selected model configurations guided by empirical scaling laws. Despite parameter-level optimization, these constraints led to modest performance shortfalls relative to similarly sized models trained with larger compute budgets. We expect that increasing available compute will allow us to relax these constraints and further improve model performance.
- **Scope of Modalities:** A.X K2 is currently text-only. While it exhibits strong reasoning performance, it lacks native multimodal understanding. We plan to extend the model with native multimodal capabilities in future work.

7 Conclusion

In this work, we presented A.X K2, a 688B-parameter MoE language model that bridges high-capacity reasoning with practical inference efficiency. Under fixed and time-bounded resources, we followed MoE scaling principles and used compute-budget-based planning to select a principled trade-off between model scale and training tokens, demonstrating that careful system-aware design can yield globally competitive capability. We also introduced *Think-Fusion*, a supervised fine-tuning recipe that trains on paired thinking and non-thinking responses, followed by multi-stage on-policy reinforcement learning, to enable explicit, user-controllable switching between *thinking* and *non-thinking* modes within a single unified model. Across mathematics, coding, and general-knowledge benchmarks in both English and Korean, our evaluations show that A.X K2 is competitive with strong open-weight baselines.

Beyond the compute-optimal architecture suggested by scaling laws, this project focuses on the system-level engineering required to translate that design into effective large-scale MoE training under real-world infrastructure constraints. Empirical profiling revealed systematic gaps between theoretical and realized throughput, driven by communication overhead, load imbalance, and execution-level inefficiencies. To preserve the intended training compute and schedule, we provisioned 512 NVIDIA B200 GPUs and iteratively optimized stability, utilization, and throughput over approximately 70 days of pretraining. This approach enabled stable convergence and predictable training behavior within fixed resource and timeline constraints.

Looking ahead, we will prioritize native multimodal capabilities and continued scaling toward the trillion-parameter regime. Building on the system and optimization insights established here, we aim to

further advance both model capability and the efficiency of large-scale training and inference.

A Contributors

All authors are listed alphabetically by first name. Names marked with an asterisk (*) indicate authors whose affiliations differ from their affiliation at the time of this work.

A.1 Model Engineering

Cheolseung Baek, Eunki Kim, Hyunho Yang, Hyunjun Eun, Jin Kim, Junyoung Park, Juyun Wee, Minsang Kim, Minsoo Kang, Seongho Choi, Sangyeol Lee, Seokhwan Jo, Seonghye Cho*, Seongmin Ok, Subin Yi, Sung Jun Cheon, Sungwan Kim, Tae Yoon Kim

A.2 Data Engineering

Dhammiko Arya, Gun Song, Gyoungeun Han, Minki Hong*, Minkyung Park*, SaeRom Kim, Sangjin Kim, Seojin Lee, Seokyoung Hong, Sereimony Sek, Singon Kim*, Sohee Park, Sungbin Yoon, Sungeun Lee, Sunwoo Lee, Wonbeom Jang, Yohan Ra, Yong-jin Han, Yujin Kang*, Yujin Lee

A.3 HPC Infrastructure

Seungsik Kim, Youngrang Kim

A.4 Business & Compliance

Seungmo Cho, Sooyeon Park, Youngjin Kim

Acknowledgment

This work was supported by the Ministry of Science and ICT (MSIT), Republic of Korea, through the National IT Industry Promotion Agency (NIPA) (Grant No. PJT-26-010018). This research was also conducted as part of the Sovereign AI Foundation Model Project (Data Track), organized by the Ministry of Science and ICT (MSIT) and supported by the National Information Society Agency (NIA), Republic of Korea (Grant No. 2026-AIData-WII01).

B Appendix

B.1 Pre-train Data Category Mixture

We report the designed composition of the pre-training mixture by language (Table 10) and by content category (Table 11). Percentages are over each stage's designed mixture; the tokens actually consumed may be smaller (e.g., Stage 1 trained on the first 6.4T tokens of its ∼8T schedule, while Stage 2, Stage 3A, and Stages 3B&3C used their full schedules of 1.4T, 0.22T, and 0.14T tokens, respectively). Stage 3B and Stage 3C share a single data mixture and are reported together. A dash (–) indicates a category not separately tracked for that stage, as the category taxonomy was refined across stages; the "Etc." row aggregates remaining minor categories (including encyclopedia, news, and articles). The "Academic Papers" category is predominantly academic papers, with a small amount of other specialized long-form domain text such as legal documents and patents. The "Instruction" category consists of instruction-style (SFT-format) data introduced during pre-training, and the "STEM" category covers science and engineering domain text distinct from the academic-paper sources. Categories are grouped by data source/domain and by the target capability they primarily serve. Percentages are rounded to two decimals and may not sum exactly to 100%.

Language	Stage 1 (%)	Stage 2 (%)	Stage 3A (%)	Stage 3B&3C (%)
English	72.72	76.80	88.66	90.24
Korean	15.40	14.33	7.07	6.15
Code	8.29	7.35	3.54	2.99
Japanese	1.41	0.50	0.24	0.20
Spanish	1.15	0.52	0.25	0.21
Chinese	1.02	0.51	0.25	0.21
Total	100.00	100.00	100.00	100.00

Table 10: Language composition of the pre-training mixture across stages (%).

Category	Stage 1 (%)	Stage 2 (%)	Stage 3A (%)	Stage 3B&3C (%)
By data source / domain				
Web	69.91	25.81	12.43	10.49
Code	8.56	18.83	9.07	7.65
Academic Papers	16.69	14.62	7.04	5.94
STEM	–	5.53	2.66	2.25
Books	0.66	3.74	1.80	1.52
Q&A	–	4.98	2.40	2.03
By target capability				
Instruction	3.45	22.49	10.83	9.14
Reasoning	–	3.06	26.33	22.22
Agent	–	–	15.14	25.55
Long-context	–	–	11.85	12.82
Etc.	0.73	0.92	0.46	0.38
Total	100.00	100.00	100.00	100.00

Table 11: Category composition of the pre-training mixture across stages (%).

B.2 SFT Data Statistics

As mentioned in the main text, we periodically assessed the model's performance during training and supplemented the data accordingly. Table 12 and Table 13 present the statistics over the entire set of samples used for SFT.

B.3 Safety Reasoning Data Construction

The safety portion of the SFT mixture builds on ESSA (Park et al., 2026), which evolves compact safety specifications for domain–task pairs and uses them to synthesize deliberation-structured safety supervision. A.X K1 used the ESSA safety data as part of its post-training mixture. For A.X K2, we extend this

Stage	Category	Reasoning		Instruction		Total	
		Tokens	Ratio (%)	Tokens	Ratio (%)	Tokens	Ratio (%)
Indexer SFT	Math	1.19B	1.96	19.45M	0.03	1.21B	1.99
	Knowledge / Science	0.86B	1.41	3.44M	0.01	0.86B	1.42
	Instruction Following	0.01B	0.02	0.54M	0.00	0.01B	0.02
	Code	0.54B	0.89	0	0.00	0.54B	0.89
	Agent	1.32B	2.18	91.63M	0.15	1.41B	2.33
	General	0.14B	0.23	34.21M	0.06	0.17B	0.29
	Safety	0	0.00	1.97M	0.00	1.97M	0.00
	Sum	4.06B	6.70	151.2M	0.25	4.21B	6.95
Main SFT	Math	15.00B	24.76	0.61B	1.01	15.61B	25.77
	Knowledge / Science	11.21B	18.50	0.11B	0.18	11.32B	18.68
	Instruction Following	1.27B	2.09	0.01B	0.01	1.27B	2.10
	Code	6.79B	11.20	0.09B	0.15	6.88B	11.35
	Agent	14.68B	24.24	1.13B	1.86	15.81B	26.10
	General	3.29B	5.44	2.11B	3.48	5.40B	8.91
	Safety	0.06B	0.10	0.02B	0.04	0.08B	0.14
	Sum	52.30B	86.32	4.08B	6.73	56.38B	93.05
Total		56.36B	93.02	4.23B	6.98	60.59B	100.00

Table 12: Token composition and ratio (%) across the SFT stages. Token counts are abbreviated as K (thousand), M (million), and B (billion).

Stage	Category	Reasoning		Instruction		Total	
		<32K	32–128K	<32K	32–128K	<32K	32–128K
Indexer SFT	Math	0.37	0.15	0.03	0.00	0.40	0.15
	Knowledge / Science	0.89	0.09	0.05	0.00	0.95	0.09
	Instruction Following	0.06	0.00	0.01	0.00	0.07	0.00
	Code	0.38	0.06	0.00	0.00	0.38	0.06
	Agent	1.02	0.15	0.02	0.01	1.03	0.16
	General	0.80	0.01	0.71	0.00	1.51	0.01
	Safety	0.00	0.00	0.04	0.00	0.04	0.00
	Sum	3.51	0.45	0.85	0.02	4.37	0.47
Main SFT	Math	4.61	1.86	0.92	0.07	5.53	1.94
	Knowledge / Science	13.84	1.13	1.64	0.00	15.48	1.13
	Instruction Following	5.53	0.00	0.09	0.00	5.62	0.00
	Code	4.80	0.69	0.72	0.00	5.52	0.69
	Agent	16.24	1.47	0.24	0.16	16.48	1.63
	General	19.20	0.15	20.90	0.02	40.10	0.16
	Safety	0.38	0.00	0.50	0.00	0.88	0.00
	Sum	64.60	5.31	25.00	0.25	89.60	5.56
Total		68.11	5.76	25.86	0.27	93.97	6.03

Table 13: Sample distribution by token length range, reported as ratio (%), across the SFT stages. "<32k" and "32–128k" denote the proportion of samples in each token-length range.

recipe with an additional specification-evolution and data-synthesis pass tailored to the A.X K2 training format.

The construction pipeline consists of four stages. First, we collect English and Korean safety prompts covering harmful requests, safety-adjacent benign requests, and over-refusal-sensitive cases. Second, each prompt is assigned a domain and task label using the ESSA taxonomy. These labels are not treated as supervision targets; they are used only to choose the relevant safety specification $S_{d,t}$ among the 50 evolved domain–task combinations. Third, a reasoning-capable teacher is prompted with the selected safety specification and the user request to produce a reasoning-mode answer. The prompt is designed so that the final answer does not cite internal specifications, rule identifiers, or taxonomy labels; instead, boundaries are explained in ordinary language and safe alternatives are provided when appropriate. Compared with the original ESSA synthesis format, which used explicit `##thought`/`##response` markers, the A.X K2 pipeline converts outputs into the standard chat-sample representation used by the target chat template.

Finally, we preserve raw request–response logs and source metadata before conversion, then run validation and filtering over the converted samples. The quality-control pass checks both safety and usefulness: unsafe operational residue, proxy harmful artifacts, privacy leakage, language mismatch, malformed reasoning/final-answer boundaries, and unnecessary refusal are routed to exclusion, regeneration, or meta-review. In particular, the filtering rubric rewards safe completion behavior: if the user's benign objective can be served without enabling harm, the preferred answer redirects to a useful safe artifact rather than issuing a blanket refusal.

B.4 Details of Evaluation Method

Serving setup All open-weight baseline models are evaluated via OpenRouter, with the provider fixed to the model publisher only. We set the reasoning effort to `xhigh` and leave all generation parameters unset—using the provider's defaults—except for the maximum number of output tokens. As OpenRouter appears to cap the output at 64K tokens by default, we override it as follows: (1) where available, we use the maximum output token length stated on the OpenRouter model description page; (2) if the context limit is hit under this setting, we instead set the maximum output length to the stated maximum context length minus 100K tokens. For Qwen3.5-397B-A17B served by Alibaba, the maximum output length is fixed at 65,536 tokens. For all other models, the resulting maximum output length is at least 128K tokens, including when it is computed as the context length minus 100K tokens.

Metrics All benchmark results are reported as pass@1 scores averaged over multiple generations per sample, with the number of generations varying by benchmark. All mathematics benchmarks use eight generations per sample.

BrowseComp For BrowseComp, we use the LLM Context API from Brave Search as the only tool available to the model; no further web-page fetching or summarization is performed. Due to search cost, the number of searches is limited to at most 10 per problem.

B.5 Details of Evaluation Benchmarks

Our evaluation prompt templates largely follow those of A.X K1 (SKT, 2026); we summarize the benchmark suite here for completeness. For A.X K2, responses are generated for all benchmarks until the model reaches its context-length limit, and responses that hit the limit are always graded as incorrect.

Math We use AIME26 (Mathematical Association of America, 2026), a prestigious high-school–level mathematics competition problem set drawn from the 2026 American Invitational Mathematics Examination (AIME); Apex (Dekoninck et al., 2026), a MathArena benchmark of 12 short-form-answer problems from 2025 mathematics competitions that then-state-of-the-art models could not solve, together with Apex-shortlist, a companion set of 48 less difficult problems on which those models reached roughly 50% accuracy; and KMO26 (1st round), consisting of problems from the first round of the 2026 Korean Mathematical Olympiad. All math benchmarks are evaluated based on the correctness of the final answer, using a chat-based chain-of-thought (Chat CoT) prompting setup that encourages sufficient intermediate reasoning, and are reported as pass@1 averaged over 8 generations (Appendix B.4). The evaluation prompt and a usage example for AIME26 are shown in Fig. 8.

Evaluation Prompt for AIME26

Solve the following math problem efficiently and clearly. The last line of your response should be of the following format: 'Therefore, the final answer is: $\boxed{ANSWER}$. I hope it is correct' (without quotes) where ANSWER is just the final number or expression that solves the problem. Think step by step before answering.

Patrick started walking at a constant rate along a straight road from school to the park. One hour after Patrick left, Tanya started running along the same road from school to the park. One hour after Tanya left, Jose started bicycling along the same road from school to the park. Tanya ran at a constant rate of 2 miles per hour faster than Patrick walked, Jose bicycled at a constant rate of 7 miles per hour faster than Tanya ran, and all three arrived at the park at the same time. The distance from the school to the park is $\frac{m}{n}$ miles, where m and n are relatively prime positive integers. Find $m + n$.

Figure 8: One example of evaluation prompt for AIME26. The model's response is evaluated by comparing the `ANSWER` field against the ground-truth answer.

Korean For Korean evaluation, we use KMMLU-Pro (Hong et al., 2025), KoBALT (Shin et al., 2025), and CLIcK (Kim et al., 2024). KMMLU-Pro includes questions based on the Korean national professional qualification exams, KoBALT measures Korean linguistic knowledge, and CLIcK evaluates Korean cultural and linguistic knowledge. CLIcK is evaluated in a standard zero-shot setting, consistent with the evaluation protocol used for pre-trained models.

Code We use LiveCodeBench v6 (Jain et al., 2024) and SciCode (Tian et al., 2024). For LiveCodeBench, generated code is executed using a Python interpreter, and correctness is determined by matching execution results with expected outputs; since the LiveCodeBench series includes recently updated problems, reducing the risk of data contamination, we evaluate only the February–May 2025 subset. SciCode consists of research-level scientific programming problems curated by scientists.

Science & Knowledge We use GPQA Diamond (Rein et al., 2024) and Humanity's Last Exam (Center for AI Safety et al., 2026). GPQA Diamond consists of high-difficulty, graduate-level questions in physics, chemistry, and biology, and HLE evaluates advanced knowledge and reasoning capabilities.

General We use IFBench (Pyatkin et al., 2025), which assesses compliance with explicit constraints such as output format, length limits, required expressions, or language usage under rule-based criteria, and AA-LCR (Artificial Analysis Team, 2025), which evaluates the model's ability to maintain coherence and retain key information when generating responses based on long input contexts.

Agentic τ^2-Bench (Barres et al., 2025), building on τ-bench (Yao et al., 2024), evaluates conversational agents in a dual-control environment; we report the Telecom domain score from Artificial Analysis. BrowseComp (Wei et al., 2025) evaluates browsing agents on locating hard-to-find information on the web; our search-tool configuration is described in Appendix B.4.

References

Artificial Analysis Team. Artificial analysis long context reasoning benchmark(lcr), 2025.

Kyunghoon Bae, Eunbi Choi, Kibong Choi, Stanley Jungkyu Choi, Yemuk Choi, Kyubeen Han, Seokhee Hong, Junwon Hwang, Taewan Hwang, Joonwon Jang, Hyojin Jeon, Kijeong Jeon, Gerrard Jeongwon Jo, Hyunjik Jo, Jiyeon Jung, Euisoon Kim, Hyosang Kim, Jihoon Kim, Joonkee Kim, Seonghwan Kim, Soyeon Kim, Sunkyoung Kim, Yireun Kim, Yongil Kim, Youchul Kim, Edward Hwayoung Lee, Gwangho Lee, Haeju Lee, Honglak Lee, Jinsik Lee, Kyungmin Lee, Sangha Park, Young Min Paik, Yongmin Park, Youngyong Park, Sanghyun Seo, Sihoon Yang, Heuiyeen Yeen, Sihyuk Yi, and Hyeongu Yun. EXAONE 4.0: Unified large language models integrating non-reasoning and reasoning modes, 2025. URL https://arxiv.org/abs/2507.11407.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649:1139–1146, 2026. doi: 10.1038/s41586-025-09962-4.

DeepSeek-AI. DeepSeekMath-V2: Towards self-verifiable mathematical reasoning, 2025. URL https://arxiv.org/abs/2511.22570.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025a. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025b. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Mostafa Dehghani et al. Scaling vision transformers to 22 billion parameters, 2023. URL https://arxiv.org/abs/2302.05442.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms, 2026. URL https://arxiv.org/abs/2605.00674.

P. Du, Y. Gao, L. Li, and X. Li. SGAMF: Sparse gated attention-based multimodal fusion method for fake news detection. *IEEE Transactions on Big Data*, 11(2):540–552, 4 2025. doi: 10.1109/TBDATA.2024.3414341.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

GLM-4.5 Team. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models, 2025. URL https://arxiv.org/abs/2508.06471.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Qianyu He, Siyu Yuan, Xuefeng Li, Mingxuan Wang, and Jiangjie Chen. Thinkdial: An open recipe for controlling reasoning effort in large language models. *arXiv preprint arXiv:2508.18773*, 2025.

Seokhee Hong, Sunkyoung Kim, Guijin Son, Soyeon Kim, Yeonjung Hong, and Jinsik Lee. From kmmlu-redux to kmmlu-pro: A professional korean benchmark suite for llm evaluation, 2025. URL https://arxiv.org/abs/2507.08924.

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

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Eunsu Kim, Juyoung Suk, Philhoon Oh, Haneul Yoo, James Thorne, and Alice Oh. CLIcK: A benchmark dataset of cultural and linguistic intelligence in korean. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 3335–3346, Torino, Italia, 2024. ELRA and ICCL.

Kimi Team. Kimi K2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Kimi Team. Kimi-K2.6, 2026. URL https://huggingface.co/moonshotai/Kimi-K2.6.

Vijay Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models, 2022. URL https://arxiv.org/abs/2205.05198.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. URL https://arxiv.org/abs/2601.05242.

Mathematical Association of America. American invitational mathematics examination (aime) 2026, 2026.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. FP8 formats for deep learning, 2022. URL https://arxiv.org/abs/2209.05433.

MiniMax. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

MiniMax. MiniMax-M2.7, 2026. URL https://huggingface.co/MiniMaxAI/MiniMax-M2.7.

NVIDIA. Nemotron-CC-v2.1, 2025a. URL https://huggingface.co/datasets/nvidia/Nemotron-CC-v2.1.

NVIDIA. Nemotron-Pretraining-Code-v2, 2025b. URL https://huggingface.co/datasets/nvidia/Nemotron-Pretraining-Code-v2.

NVIDIA. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026. URL https://arxiv.org/abs/2606.15007.

Team Olmo. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Dongcho Park, Yeowon Jung, and Sung Jun Cheon. ESSA: Evolved safety specification alignment. ICML 2026 Workshop on Trustworthy AI for Good, 2026. URL https://openreview.net/forum?id=Imcv0X7c36&noteId=ZgVGRTdLER. Presented on July 10, 2026.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. FineWeb2: One pipeline to scale them all – adapting pre-training data processing to every language, 2025. URL https://arxiv.org/abs/2506.20920.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models, 2023. URL https://arxiv.org/abs/2309.00071.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2025.

Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models, 2025a. URL https://arxiv.org/abs/2501.11873.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025b. URL https://arxiv.org/abs/2505.06708.

Zihan Qiu, Zeyu Huang, Kaiyue Wen, Peng Jin, Bo Zheng, Yuxin Zhou, Haofeng Huang, Zekun Wang, Xiao Li, Huaqing Zhang, et al. A unified view of attention and residual sinks: Outlier-driven rescaling is essential for transformer training. *arXiv preprint arXiv:2601.22966*, 2026.

Qwen Team. Qwen3.5-397B-A17B, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-397B-A17B.

Qwen Team. Qwen3.5-Omni technical report, 2026b. URL https://arxiv.org/abs/2604.15804.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Bita Darvish Rouhani et al. Microscaling data formats for deep learning, 2023. URL https://arxiv.org/abs/2310.10537.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 1715–1725, 2016.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Hyopil Shin, Sangah Lee, Dongjun Jang, Wooseok Song, Jaeyoon Kim, Chaeyoung Oh, Hyemi Jo, Youngchae Ahn, Sihyun Oh, Hyohyeong Chang, Sunkyoung Kim, and Jinsik Lee. KoBALT: Korean benchmark for advanced linguistic tasks, 2025. URL https://arxiv.org/abs/2505.16125.

SKT. A.X 4.0, 2025. URL https://huggingface.co/skt/A.X-4.0.

SKT. A.X K1 technical report, 2026. URL https://arxiv.org/abs/2601.09200.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025.

Changxin Tian, Kunlong Chen, Jia Liu, Ziqi Liu, Zhiqiang Zhang, and Jun Zhou. Towards greater leverage: Scaling laws for efficient mixture-of-experts language models, 2025. URL https://arxiv.org/abs/2507.17702.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. Scicode: A research coding benchmark curated by scientists. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30624–30650. Curran Associates, Inc., 2024. doi: 10.52202/079017-0963. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/36850592258c8c41cecdaa3dea5ff7de-Paper-Datasets_and_Benchmarks_Track.pdf.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024. URL https://arxiv.org/abs/2408.15664.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models, 2023. URL https://arxiv.org/abs/2309.16039.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Z.ai. GLM-5.1, 2026. URL https://huggingface.co/zai-org/GLM-5.1.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Chenggang Zhao, Zhean Xu, Liang Zhao, Jiashi Li, Chenhao Xu, Anyi Xu, Shengyu Liu, Kexing Zhou, and Kuai Yu. Deepgemm: clean and efficient blas kernel library on gpu. https://github.com/deepseek-ai/DeepGEMM, 2025.