VAETKI Technical Report

NC-AI Consortium

January 2026

Abstract

This report introduces VAETKI, a series of large language models developed at 100B, 20B, and 7B scales to secure national data sovereignty and promote industrial AI Transformation (AX). The models utilize a decoder-only Mixture-of-Experts (MoE) architecture, integrating Multi-Latent Attention (MLA) with Local-Global Interleaving. This configuration achieves an 83% reduction in KV cache size and a 40% gain in training efficiency compared to conventional MLA. Our pre-training corpus involves a curated 5-trillion-token subset selected through a seven-stage refinement pipeline from a total of 20 trillion tokens. We further implemented a three-stage post-training pipeline—consisting of Reasoning-centric SFT, High-Quality SFT, and DPO—trained on 8 million instruction-tuning triplets. This effort was complemented by 17.6 billion tokens of specialized datasets for multilingual, cultural, and reasoning domains. Evaluation results on VAETKI-100B-A10B show significant performance on Korean benchmarks like CLIcK and KoBALT, alongside robust instruction-following capabilities as measured by IFEval. The results demonstrate that the VAETKI series provides a specialized foundation for high-performance AI applications optimized for the Korean context.

Authors

Principal Investigators (Project Leads)

NC-AI Yeonsoo Lee

ETRI Oh-woog Kwon

Korea University Heuiseok Lim

Core Developers (Lead Contributors)

NC-AI Young-rok Cha

ETRI Jong-hun Shin

Korea University Jungseob Lee

Contributors

NC-AI Ayoung Moon, Bohui Kim, Geonsoo Kim, Hyonsu Choe, Hyowon Wi, Hyungjong Noh, Jeongbeom Jeong, Jeongho Ju, Jonghyeon Lee, Junsoo Park, Kyungsoo Kim, Meiying Ren, Minkyoung Jung, Saebyeok Lee, Seungjin Lee, SunYoung Park, Younghyun Yu

ETRI Hongjin Kim, Hwa Jeon Song, Ilchae Jung, In Jun Park, Jeong Heo, Jeonguk Bang, Jihee Ryu, Jin Seong, Junyong Park, Kiyoung Lee, Kiyoung Park, Minkyu Lee, Ran Han, Sanghun Jeon, SangHun Kim, Sanghyun Cho, Soojin Jang, Soojong Lim, Yong-Ju Lee, Yoonhyung Kim, Young-Kil Kim, Youngwan Lee

Korea University Chanhee Park, Dahyun Jung, Dongjun Kim, Gyuho Shim, Hyeonseok Moon, Jaehyung Seo, Jungsun Jang, Junyoung Son, Minhyuk Kim, Seongtae Hong, Seungyoon Lee, Suhyune Son, Yongchan Chun, Youngjoon Jang

1 Introduction

Large Language Models (LLMs) have emerged as a foundational technology for artificial intelligence, enabling advanced capabilities in natural language understanding, generation, and reasoning across a wide range of applications. However, recent progress in LLMs has been predominantly driven by a small number of global big-tech companies, raising concerns regarding technological dependency, limited control over national data, and insufficient alignment with local languages and cultural contexts. In the case of the Korean language, existing foundation models often struggle to fully capture linguistic characteristics, cultural nuances, and domain-specific requirements, thereby limiting their effectiveness in domestic industrial and public-sector applications.

To address these challenges, this project aims to establish technological self-reliance in Korean foundation models by developing a large-scale, high-performance LLM that is deeply optimized for Korean language and cultural understanding. A central objective of this effort is to reduce dependence on foreign proprietary models and infrastructures, thereby safeguarding national data sovereignty while securing core LLM technologies that can be independently developed, operated, and extended within the domestic ecosystem.

In addition to technological independence, the proposed Korean foundation model is explicitly designed to accelerate AI transformation (AX) across a broad range of industries, including manufacturing, finance, and service sectors. By providing a versatile and general-purpose LLM capable of accommodating diverse domain-specific requirements, this work aims to significantly lower the barriers to AI adoption for domestic enterprises. In particular, the model is intended to enable organizations to leverage high-performance AI capabilities without the need for costly, large-scale infrastructure investments, thereby fostering a sustainable AI ecosystem and promoting widespread industrial innovation.

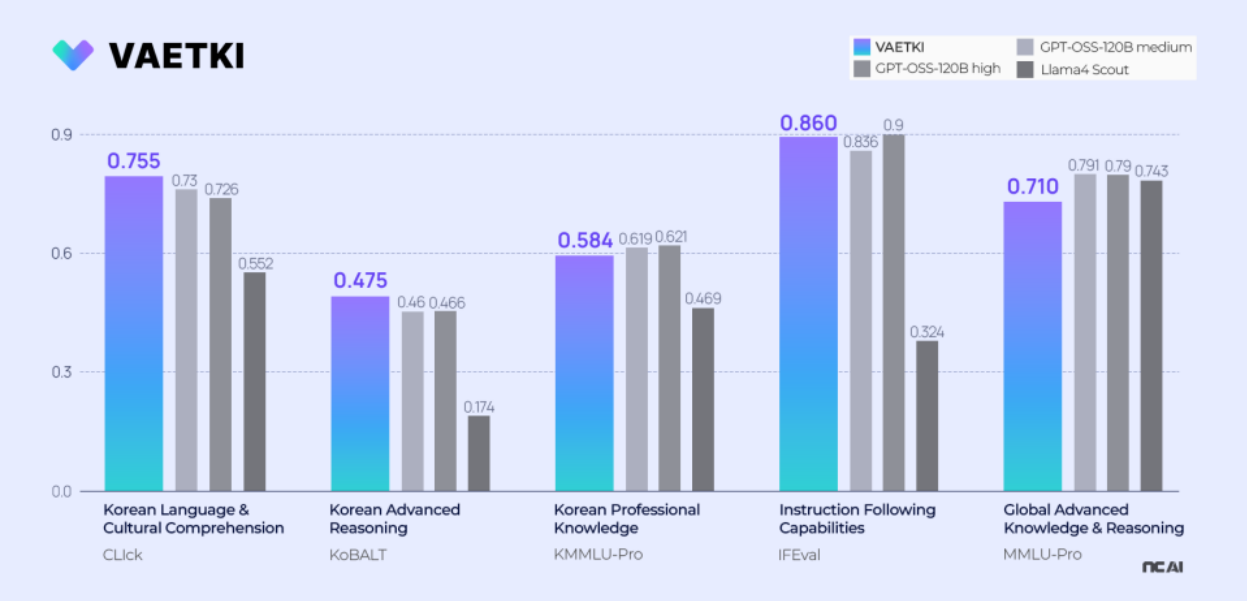

Figure 1: Performance comparison of VAETKI-100B-A10B and baselines across Korean and English benchmarks.

To this end, we designed and developed a family of language models at three different scales of 100B,

Table 1: Architectural configurations of the VAETKI model family

Model	VAETKI-100B-A10B	VAETKI-20B-A2B	VAETKI-7B-A1B
# Dense Layers	3	1	1
# MoE Layers	45	23	23
# Experts (Total)	128	128	64
# Experts (Active)	8	8	5
Attention Heads	24	16	12
Context Length	32k	16k	16k

20B, and 7B parameters, with each targeting distinct deployment scenarios and resource constraints. This multi-scale strategy enables both high-end performance for large-scale applications and efficient deployment in environments with limited computational resources, thereby supporting a wide spectrum of industrial and research use cases. By sharing a unified architectural design and training methodology across scales, the model family ensures consistency in performance characteristics while allowing flexible adaptation to diverse operational requirements.

2 Architecture

The VAETKI series adopts four Mixture-of-Experts (MoE) models, namely VAETKI-100B-A10B, VAETKI-20B-A2B, VAETKI-7B-A1B language models designed with a focus on efficient reasoning and scalable training.

2.1 Model Architecture

The VAETKI models are fundamentally based on a decoder-only Transformer architecture, following an autoregressive language modeling formulation. Each model consists of a stack of Transformer layers, where every layer is composed of a causal self-attention module and a Mixture-of-Experts (MoE) based feed-forward network. This design enables efficient scaling while preserving strong reasoning capabilities under sparse activation.

Attention Mechanism. Instead of conventional multi-head attention, we employ a refined attention structure that integrates Multi-Latent Attention (MLA) with a Local–Global Interleaving strategy to optimize both memory and computational efficiency. While MLA [1] significantly improves memory efficiency by compressing the KV context and maintains a stronger representational capacity compared to Grouped Query Attention (GQA) [2], it does not inherently address the quadratic computational complexity of the attention mechanism.

To manage this, we incorporate an interleaving approach inspired by the structural efficiency of Gemma 2 [3]. This design is based on the observation that expanding the receptive field inevitably leads to the dispersion of softmax attention weights [4, 5], a structural characteristic that often yields diminishing returns in long-context model performance. Recognizing this dispersion as a fundamental property rather than a problem to be solved, we adopt a pragmatic stance by reducing redundant global references. By strategically interleaving local sliding window attention with

Table 2: Empirical comparison of iteration time and efficiency between Vanilla MLA and Local–Global Interleaved MLA.

Layers	Time per Iteration (ms) (Vanilla MLA)	Time per Iteration (ms) (Interleaved MLA)	Efficiency
8	51,959	56,726	-0.084
24	74,049	56,334	0.314
36	108,740	75,895.4	0.433
48	144,483	103,178	0.400

periodic global attention, we focus computational resources on the most essential dependencies, effectively lowering the computational burden during both training and inference.

In practice, we follow the configuration established in Gemma 3 [6], which observed that the interleaving frequency does not lead to meaningful degradation in language modeling perplexity; accordingly, we retain a local-to-global ratio of 5:1. To support this dual-pattern approach, we apply Rotary Positional Embeddings (RoPE) with decoupled base frequencies, specifically setting $\theta = 10$K for local attention to maintain high-resolution local focus and $\theta = 1$M for global attention to preserve long-range dependency capabilities. By integrating these cross-architectural optimizations, our final configuration achieves an approximately 83% reduction in KV cache size compared to the standard MLA implementation when handling a sequence length of 128K.

Empirical Evaluation of Local-Global Interleaved MLA. To evaluate the efficiency gains of Local–Global Interleaved MLA, we conducted empirical experiments using eight NVIDIA H100 NVL PCIe GPUs. The evaluation was performed with a sequence length of 32k tokens and a hidden dimension of 3072. The MLA hyperparameters follow standard settings, with `qk_dim` set to 196 and `v_dim` set to 128. Low-rank adaptation is applied with `q_lora_rank` set to 1536, corresponding to half of the hidden dimension, and `kv_lora_rank` set to 512, corresponding to one-sixth of the hidden dimension. and FlashAttention-3 [7] is used as the attention backend. Under this setup, we observe that the configuration reaches a performance sweet spot at approximately the 36th Transformer layer, beyond which the achieved TFLOPs per second per GPU becomes saturated. With the evaluated 100B-scale model, this design yields an additional training efficiency gain of approximately 40%, demonstrating the practical benefits of the proposed attention architecture for large-scale MoE models. The empirical results are summarized in Table 2. Efficiency is computed as $(T_{\text{vanilla}} - T_{\text{interleaved}})/T_{\text{vanilla}}$, where T_{vanilla} and $T_{\text{interleaved}}$ denote the time per iteration of Vanilla MLA and Local–Global Interleaved MLA, respectively.

Feed-Forward Network and MoE Design. The feed-forward networks adopt a fine-grained Mixture-of-Experts (MoE) architecture [8], which has become a dominant design choice for large-scale language models. In the largest configuration, VAETKI-100B-A10B, each MoE layer consists of 128 experts together with a single shared expert. For each token, 8 experts are dynamically activated and their outputs are combined via a weighted sum based on routing probabilities. The final output is then derived by additively combining this result with the output of the shared expert. The resulting output representation is configured to be symmetric with respect to the dense FFN intermediate size, ensuring consistent dimensionality across dense and sparse layers.

Expert routing is performed based on affinity scores computed using a sigmoid function. Instead

of grouping-based top-k selection, we adopt the Global Auxiliary Loss [9] for expert routing, and further augment it with a conventional auxiliary loss to encourage balanced expert utilization. Both auxiliary losses are applied with a coefficient of 1×10^{-4}.

Normalization and FP8-ready Training Stability. Layer normalization applied to both attention modules and feed-forward networks following the structure introduced in Gemma 2 [3], which is also referred to as Peri-LN [10]. This structure has been widely adopted in recent open-weight large language models such as Gemma 3 and OLMo 2 [11], to improve early-stage training stability. Given that the scale of training tokens is critical to final performance, we utilize FP8 (E4M3) precision with NVIDIA Transformer Engine [12] for the linear layers within both MoE and dense feed-forward networks to accelerate training throughput.

During our exploration of FP8 training recipes, we observed that the blockwise scaling strategy, as presented in DeepSeek-V3 [13], could induce instabilities during the very early stages of training. To address this, we employ delayed scaling during the first stage of our three-stage pre-training curriculum (see Section 3.2), subsequently transitioning to blockwise scaling in the later stages to improve training efficiency. Furthermore, we employ a Zero-Centered RMSNorm [14] formulation to encourage activation distributions to cluster around zero, significantly enhancing numerical robustness in large-scale MoE settings under low-precision constraints.

2.2 Tokenizer

We employ a subword tokenizer based on a byte-fallback Byte Pair Encoding(BPE) algorithm with a vocabulary size of 135,168 (approximately 135K). The tokenizer is trained using SentencePiece on a corpus sampled to reflect the relative data proportions of the target languages. To ensure compatibility and consistent behavior across different implementations, the trained tokenizer is converted into a HuggingFace Transformers compatible tokenizer format before being used in training and inference.

Given that English constitutes the largest portion of the training data, more than half of the tokenizer vocabulary is allocated to Latin-based scripts. At the same time, to align with the original goal of building a sovereign language model, Korean accounts for approximately 20% of the vocabulary. To achieve this balance, subword units are first extracted via Unicode character mapping, classified into character classes, and then rebalanced according to the target language distribution.

For comprehensive Korean language coverage, the tokenizer includes the full Korean syllable block (approximately 11K syllables), as well as Hangul composition (Jamo) characters. This design supports both historical Korean text and modern Korean language applications. In addition, for code generation tasks, tab characters and consecutive whitespace tokens are explicitly reserved to ensure faithful tokenization of indentation-sensitive code.

The character-class distribution of the final subword vocabulary is summarized in Table 3.

Chat Template. We design the chat template to ensure that our model reaches the correct response to a user's query through a reasoning path in all scenarios with the exception of tool-use. This configuration acts as a default "thinking mode" effectively maximizing the utilization of the

Table 3: Character class distribution of the VAETKI tokenizer vocabulary.

Class	Latin	Hangul	Kana	Han	Cyrillic	Symbols	Digits	Others
Count	76,222	27,301	11,179	16,472	1,539	3,760	20	778
Ratio	55%	20%	8%	12%	1.13%	2.78%	0.01%	0.57%

internal knowledge of the model to achieve superior performance. Initiating with a <think> token, the model's output distinguishes the reasoning phase from the response phase via the </think> token emitted upon the completion of sufficient reasoning. This structural separation prevents the model's inherent reasoning from being conflated with the final response while providing an explicit boundary for the model to signal its transition. Furthermore, utilizing the </think> token as a clear delimiter facilitates the seamless extraction of the response phase, allowing for a more intuitive interpretation by the user.

In contrast, for tool-use scenarios featuring descriptions of available tools, we ensure that our model skips the reasoning phase to immediately enter the response phase. We explicitly adopt this "non-thinking mode" to accommodate the heavy computational burden and context complexity characteristic of tool-calling tasks involving a large number of candidate tools. Technically, this is implemented by configuring the template to immediately emit an empty <think> block, thereby bypassing explicit reasoning and generating the response directly.

Regarding multi-turn interactions, our chat template ensures that the model does not reference the reasoning path from prior outputs. Given that reasoning generates a massive volume of tokens through extensive trial and error as well as path exploration, allowing the model to access only the response phase of previous outputs serves as a highly cost-efficient approach.

2.3 Infrastructure

To support the training and evaluation of the VAETKI models, a total of 1,016 NVIDIA H100 80GB PCIe GPUs were utilized, which were provided through the support of the Ministry of Science and ICT (MSIT) and the National IT Industry Promotion Agency (NIPA) of the Republic of Korea. The overall resource allocation for model development spanned approximately four months, during which the average GPU utilization reached 85.2%. These computational resources were provided with support from the Ministry of Science and ICT (MSIT) of the Republic of Korea and the National IT Industry Promotion Agency (NIPA). Through this program, we were granted access to the NAVER Cloud MLX Platform, on which all model development and experimentation were conducted.

Environments and Networking. The MLX Platform provides a Kubernetes-based cloud infrastructure tailored for large-scale distributed training. Within this environment, we had access to a total of six 400 Gb/s InfiniBand lanes. Among them, one lane was allocated for TCP over InfiniBand, and another was dedicated to high-throughput storage access using DDN systems. As a result, four InfiniBand lanes were available for tensor communication during distributed training.

Compared to frontier-level LLM development environments, where all eight InfiniBand lanes are dedicated to tensor exchange, this configuration may introduce potential communication bottlenecks. To mitigate this issue, we selectively rerouted a subset of inter-host collective operations to Gloo, PyTorch's IP-based collective communication backend. While the majority of GPU collectives

continued to rely on NCCL for high-performance device-level communication, Gloo was used as an auxiliary path for specific cross-host collectives, such as all-gather. This hybrid communication strategy allowed us to alleviate pressure on the InfiniBand fabric and reduce communication contention, resulting in more stable and efficient large-scale training.

Parallelism Strategy. We employ a four-dimensional parallelism strategy consisting of Tensor Parallelism (TP), Pipeline Parallelism (PP), Expert Parallelism (EP), and Data Parallelism (DP). For most stages of pretraining, where the context length is relatively moderate, a three-dimensional parallelism configuration based on PP, EP, and DP is applied to maximize throughput and simplify communication patterns.

During later phases of pretraining that require long-context handling, we additionally enable Tensor Parallelism together with Sequence Parallelism (SP) [15]. This combination mitigates memory growth in components where Tensor Parallelism is not directly applicable, such as LayerNorm, thereby improving memory efficiency and enabling stable training at extended sequence lengths.

3 Pre-Training

3.1 Training Data

The pretraining corpus consists of a mixture of Common Crawl–based web data, code data, and curated or synthetic datasets targeting mathematics and science domains. We prioritize publicly available datasets governed by permissive licenses such as CC-BY (e.g., Datacomp-LM [16]). These datasets are restricted to resources that are accessible through open platforms such as the HuggingFace Hub and ModelScope Datasets.

For the code domain, we rely on datasets collected from the Software Heritage Archive, including The Stack-v2 [17]. We source Korean-language data primarily from AIHub [18], the national AI data platform operated by the National Information Society Agency (NIA) of the Republic of Korea.

Based on these diverse sources, the total accumulated corpus is composed of approximately 10 trillion tokens from publicly accessible datasets, 2 trillion tokens from internally curated and refined datasets, and an additional 8 trillion tokens collected directly from the web. Due to constraints in available compute resources, we utilized a curated subset of 5 trillion tokens selected through a rigorous seven-stage refinement process, rather than training on the entire corpus. The pipeline begins with heuristic filtering to eliminate low-quality data, followed by document-level deduplication. Subsequently, we applied perplexity filtering and quality evaluation models to exclude anomalous documents and substandard content. The final stages involve the removal of harmful content, PII de-identification, and the elimination of spam and noise patterns to ensure the overall safety and precision of the dataset.

In addition, to prevent benchmark contamination, a 12-gram decontamination process was performed against well-known evaluation datasets. These steps are designed to improve data quality and to ensure fair and reliable downstream evaluation.

To facilitate subsequent alignment stages, we introduce a mid-training phase [19] in the later stages of pretraining. This phase leverages data originally curated for post-training but excluded from

alignment due to quality constraints, nevertheless retaining diverse knowledge and instruction-style conversational patterns. By incorporating this data during pretraining, the model is gradually exposed to instruction-following inputs and dialogue structures, thereby reducing the optimization burden in subsequent post-training stages.

3.2 Training Stages

Following the general training methodology established in Qwen 3 [20], our model is pre-trained using a multi-stage curriculum designed to progressively enhance general language competence, structured reasoning ability, and long-context modeling. Each stage is configured with distinct data compositions, sequence lengths, and optimization strategies to support stable scaling and efficient learning.

General Stage. In the first stage, the model is trained on large-scale and diverse corpora to acquire broad language proficiency and general world knowledge. This stage primarily focuses on general-domain data with moderate sequence lengths, serving as the foundation for subsequent specialization. To promote stable optimization dynamics and consistent representation learning across heterogeneous data sources, we employ a constant learning rate schedule during this stage.

Specifically, the learning rate is set to 2×10^{-4} and maintained throughout the training process. The sequence length is fixed at 4,096 tokens. For the 100B-scale model, we estimate the critical batch size [21] and set an upper bound of 55M tokens as the maximum effective batch size. Instead of adopting the maximum batch size from the outset, we employ a progressive batch scaling strategy to improve training stability and to prevent premature loss convergence caused by overly rapid increases in batch size. Training begins with an effective batch size of 8M tokens, which is gradually increased to 16M tokens and then to 32M tokens as training proceeds. Under this configuration, the General Stage consumes approximately 3.6T tokens in total.

Reasoning Stage. The second stage focuses on strengthening the model's reasoning capabilities. Inspired by the Qwen 3 training pipeline, the data mixture in this stage is adjusted to increase the proportion of STEM-related content, coding data, and reasoning-oriented samples. Compared to the first stage, this stage emphasizes higher-quality and more structured data, and adopts a more aggressive learning rate decay strategy to stabilize convergence.

Training in Stage 2 employs a linear learning rate decay scheduler, with the learning rate decayed to a minimum value of 1×10^{-4}. The total training volume for this stage amounts to approximately 1.5T tokens. After processing roughly 500B tokens, the effective batch size is increased from 32M tokens to 46M tokens to stabilize the variance in batch-wise gradients and facilitate loss reduction.

To further enhance reasoning and bidirectional context modeling, we incorporate Fill-in-the-Middle (FIM) training scheme [22] during this stage. Before Stage 2 training, we assign some special tokens (`<|fim_prefix|>`, `<|fim_middle|>`, `<|fim_suffix|>`, and `<|fim_eod|>`) within the reserved region of the tokenizer and used consistently throughout training. FIM reformulation is applied to 60% of the Stage 2 data, while the remaining 40% retain the standard left-to-right autoregressive format. We adopt the FIM scheme of Variant 2, following established prior work.

When applying FIM, different strategies are used depending on the data domain. For general-domain

text, prediction targets are constructed by randomly selecting middle spans at the character level. In contrast, for code-domain data, the middle segments are formed based on whitespace and line-break boundaries, preserving syntactic structure and indentation patterns critical for code understanding.

Long Context Stage. In the final stage, the training corpus is extended to include long-context data in order to expand the model's effective context length. Data from earlier stages are combined with high-quality long-sequence samples, and the model is trained with substantially increased sequence lengths. Consistent with prior work such as Qwen 2.5 and Qwen 3, positional encoding configurations are adjusted to support long-context training.

Specifically, approximately 20B tokens are trained with increased sequence lengths of 8K, 16K, and 32K in a staged manner, enabling stable adaptation to long-context modeling. To compensate for the increased context length, we apply Position Interpolation (PI) [23] with a ratio of 8 exclusively to the rotary positional embeddings used in global attention. This selective application enables effective long-context generalization while preserving the stability of local attention patterns.

4 Post-Training

4.1 Training Data

To bridge the gap between pre-training representations and practical user interaction, we directly construct and curate a large-scale instruction tuning dataset comprising approximately 8 million raw query-reasoning path-response triplets. We design a comprehensive three-stage post-training pipeline: **Reasoning-centric SFT**, **High-Quality SFT**, and **Direct Preference Optimization (DPO) [24]**. While the initial corpus is vast, we apply distinct filtering and synthesis strategies appropriate for each phase to progressively refine the model's capabilities.

4.1.1 Synthesized Data

Reasoning-Centric SFT and Chat Transition. The first subset is designed to seamlessly transition the model into a chat-based interaction mode while conducting reasoning-centric Supervised Fine-Tuning (SFT). We curate approximately 3 million samples specifically targeting logic-heavy domains such as Code, Mathematics, and Logical Reasoning. This foundational SFT phase serves to instill core reasoning structures and adaptability across diverse problem types. By prioritizing the establishment of generalizable reasoning paths and conversational formats over strict stylistic constraints, we ensure the model develops a comprehensive understanding of complex logical dependencies, laying a robust foundation for the high-precision tuning in subsequent stages.

High-Quality SFT. For the SFT stage, we employ a rigorous selection process to curate approximately 2 million high-quality samples. A core innovation in this stage is the *Correctness-Guaranteed Reasoning Path Synthesis*. For deterministic domains like Mathematics and Coding, we did not simply rely on external model outputs; instead, we synthesized and rigorously verified reasoning chains using our proprietary high-capability models to ensure the intermediate logic faithfully leads to known correct answers.

- **Quality Filtering:** To ensure the highest data standards, we implement a rigorous filtering mechanism applied strictly to both queries and model responses. Query-based filtering relies on a proprietary Clarity Tagging Model that evaluates inputs based on six weighted metrics: *Task Definition* (30%), assessing action specificity; *Necessary Context* (20%), checking for background information; *Specificity & Constraints* (15%), ensuring explicit limitations; *Ambiguity & Terminology* (15%), detecting undefined terms; *Self-Containment* (10%), verifying information completeness; and *Scope & Feasibility* (10%), confirming executability. Only queries achieving high clarity scores are retained. Conversely, for Response-based filtering, we apply distinct verification strategies tailored to the task type. In objective domains, we utilize Code Execution and Pass@N metrics to ensure correctness and strictly filter out responses exhibiting excessive repetition, language mixing, or instability. In open-ended scenarios, we employ a Qwen-based evaluator, retaining only responses with quality scores exceeding 0.85.

- **Linguistic Diversity and Tone-Aware Translation:** The dataset is linguistically diverse, comprising English (58%), Korean (20%), Japanese (10%), Chinese (10%), and other languages (2%). A key feature of our multilingual strategy is *Tone-Aware Translation*. When translating user queries, we apply specific stylistic weights to simulate diverse user personas. For instance, Korean queries are distributed across *formal_polite* (40%), *casual_polite* (30%), *casual_informal* (25%), and *mixed* (5%) styles. Similar weighted distributions were applied to Chinese (e.g., formal vs. casual) and Japanese (e.g., standard polite vs. casual), ensuring the model is robust to various conversational nuances across languages.

- **Multi-turn and Agent Capabilities:** To support long-context interactions, 6% of the dataset consists of multi-turn dialogues, constructed via a two-phase process: initial seed generation with human verification, followed by expansion using a dedicated consecutive query generator model. Tool-use capabilities are significantly enhanced by restructuring existing datasets through two strategies: *Cross-lingual Adaptation*, where tool descriptions and arguments were translated to simulate non-English tool usage; and *Complexity Injection*, where irrelevant "distractor" tools were added to the context. On average, the model is trained to select the correct tool from a set of over 30 candidates.

- **Identity, Safety, and Cultural Alignment:** We curate approximately 3,000 samples to establish the model's self-cognition, ensuring that it correctly identifies itself as "VAETKI" and understands its development purpose. Beyond identity, safety alignment is strictly tailored to the South Korean cultural and legal context while adhering to global standards. We apply *Cultural Context* filtering, ensuring that historical events (e.g., Tiananmen Square) are described with factual neutrality aligned with domestic sentiments. Conversely, content violating the National Security Law or containing specific forms of domestic hate speech is strictly classified as "Unsafe", aligning the model with Korean societal values.

Offline Preference Optimization. Finally, to align the model with human preferences, we construct a Direct Preference Optimization (DPO) dataset consisting of approximately 200,000 pairs generated from the model at an earlier training stage. "Chosen" responses comprise synthesized samples with verified correctness or outputs achieving high scores from an LLM-as-a-judge. In contrast, "Rejected" responses capture specific model failure modes, including incorrect answers, repetition loops, and excessively verbose reasoning paths, effectively steering the model away from hallucinations and instability.

4.1.2 In-House Curated Data

In addition to the general instruction-tuning pipeline, we constructed an additional dataset totaling 17.6 billion tokens for the development of VAETKI, with the support of the Ministry of Science and ICT (MSIT) and the National Information Society Agency (NIA). This effort was aimed at reinforcing the model's capabilities in areas that require specialized knowledge and cultural specificity. The constructed datasets includes:

- **Multilingual Dataset:** A corpus of Chinese and Japanese open-source data was established to improve the model's linguistic adaptability and generation performance in both languages.
- **Korean Cultural Heritage and Q&A:** To instill an understanding of the Korean context, we generated specialized Q&A datasets based on authoritative Korean dictionaries and academic literature in Korean studies.
- **High-Quality Mathematical Reasoning:** We developed a reasoning dataset featuring middle and high school mathematics problems along with their detailed, step-by-step solution paths to enhance the model's logical rigor.
- **Instruction-Style Multi-turn Dialogue:** A collection of multi-turn dialogue data was established based on open-source queries to further diversify the model's conversational patterns.

By integrating these datasets alongside our three-stage post-training pipeline, we ensured that VAETKI possesses not only general conversational proficiency but also good performance in regional languages, cultural understanding, and complex reasoning.

4.2 Supervised Fine-Tuning

Following post-training data construction, we perform large-scale SFT to further align the pre-trained model with high-quality instruction-following behavior and robust reasoning performance. This stage focuses on stabilizing generation quality, improving instruction adherence, and reinforcing correctness across diverse domains, while preserving the long-context capabilities acquired during pre-training.

The SFT objective is formulated as a standard next-token prediction loss, applied selectively through loss masking. Specifically, loss is computed only over model-generated segments such as the *thinking path* and *final answer*, while all remaining components including system instructions, user prompts, and formatting tokens are fully masked. This design concentrates optimization pressure on reasoning quality and response correctness, rather than surface-level prompt imitation.

Since the volume of SFT data is relatively small compared to pre-training, we configure training with a small batch size to improve sample efficiency. To account for the increased gradient noise induced by small-batch optimization [25], we adopt a conservative learning rate regime. Training is conducted with an effective batch size of approximately 17M tokens, and the total SFT training volume amounts to roughly 430B tokens. We employ a linear learning rate decay schedule, decaying the learning rate from an initial value of 6×10^{-6} to a minimum of 3×10^{-7}, enabling steady refinement of high-level reasoning behaviors while minimizing catastrophic forgetting.

To ensure stable SFT optimization in large-scale MoE models, we adopt both initialization- and training-level stabilization strategies. Specifically, the SFT model is initialized from a linear merge of pre-trained checkpoints with identical weights, yielding a stable and unbiased starting point that reduces variance across runs [26, 27], which is particularly beneficial under the small-batch setting. During training, router parameters are frozen throughout SFT [28], and the auxiliary load-balancing loss is down-weighted by a factor of 20 compared to pre-training, reflecting the reduced emphasis on expert utilization diversity during alignment-focused optimization.

For improved training efficiency, domain-aware data packing is employed, allowing samples from similar domains to be concatenated within the same sequence [29]. This strategy reduces padding overhead and improves token utilization without compromising data integrity. To preserve the 32K context length learned during long-context pre-training, training is executed using a hybrid parallelism configuration that combines TP, PP, EP, DP, and SP.

Given the relatively small batch size per pipeline stage, naive pipeline parallelism would introduce significant pipeline bubbles and reduce throughput. To address this, we adopt an interleaved pipeline parallelism strategy, which effectively overlaps computation across multiple micro-batches and substantially improves hardware utilization [30].

4.3 Preference Optimization

Following the supervised fine-tuning stage, we implement Direct Preference Optimization (DPO) [24] to further align the behavior of the model with human values and robust reasoning patterns. Unlike reinforcement learning approaches that require a separate reward model [31, 32, 33], DPO directly optimizes the policy by leveraging the preference pairs constructed in the post-training data phase. This stage is critical for sharpening the capability of the model to distinguish between high-quality reasoning and plausible but incorrect hallucinations.

We utilize the preference dataset $\mathcal{D} = \{(x, y_w, y_l)\}$ described in Section 4.1.1, where x represents the prompt, y_w is the synthesized correctness-guaranteed response (chosen), and y_l is the rejected response. The training objective is to maximize the margin between the likelihood of the chosen response and the rejected response subject to a Kullback-Leibler (KL) divergence [34] constraint relative to the reference model π_{ref}. The loss function is formulated as:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right] \tag{1}$$

Here, π_{ref} is initialized from the final checkpoint of the SFT stage, and β is a hyperparameter controlling the strength of the KL penalty. We set β to a conservative value to prevent excessive deviation from the reasoning capabilities established during SFT. This setting allows the model to sufficiently penalize the specific failure modes present in y_l, such as repetition loops and unnecessarily verbose reasoning paths, without degrading general performance.

To ensure stability during preference learning, we maintain the *router freezing* strategy adopted in the SFT stage. By keeping the MoE gating mechanisms fixed, we ensure that the specialization of experts learned during pre-training and SFT is preserved. Consequently, the optimization is directed solely towards the refinement of expert representations and output logits. Furthermore, to mitigate the risk of reward hacking or forgetting, we employ a lower learning rate compared to the

SFT phase with a warm-up period comprising 3% of the total training steps followed by a cosine decay schedule.

A key advantage of our data construction strategy involves pairing synthesized and verified reasoning paths (y_w) against model-generated errors (y_l). This structure is effectively realized in this stage as the optimization process explicitly suppresses the generation of "hallucinated reasoning" where the model produces convincing but logically flawed steps. It also penalizes the linguistic instabilities captured in the rejected samples, including code mixing or stuttering. This results in a final model that adheres to instructions and exhibits a preference for concise, logically sound, and structurally coherent outputs.

5 Evaluation

5.1 Benchmark Tasks

To assess the performance of VAETKI-100B-A10B, we evaluate the model on a set of widely used Korean and English benchmarks that capture general knowledge, reasoning, and instruction following capabilities.

For Korean benchmarks, general knowledge and language competence are evaluated using KMMLU-Pro [35], CLIcK [36], and KoBALT [37], which collectively assess professional knowledge, linguistic understanding, and cultural context in Korean. Reasoning ability in Korean is measured with HRM8K [38], a benchmark designed to test structured mathematical reasoning.

For English benchmarks, general knowledge is assessed using MMLU-Pro [39] and GPQA-Diamond [40], covering broad academic knowledge and graduate-level scientific reasoning. Reasoning performance is measured via HLE (text-only) [41], which focuses on high-level expertise. Instruction-following capability is measured using IFBench [42] and IFEval [43], capturing the model's ability to understand and execute explicit task instructions.

In addition to task performance, we evaluate the safety and alignment behavior of VAETKI-100B-A10B using a set of established safety benchmarks. Exaggerated or inappropriate safety responses are assessed with XSTest [44], which identifies over-safety tendencies in large language models, while OR-Bench [45] is used to measure over-refusal behavior on benign requests, capturing the balance between safety enforcement and utility. Robustness to more complex and adversarial scenarios is evaluated using SaladBench [46], a hierarchical benchmark designed to assess model behavior under compositional attack patterns, and Strong Reject [47], which measures the consistency of refusal for empty or underspecified jailbreak attempts.

5.2 Results and Discussion

Evaluation of the 100B-Scale Model. Table 4 reports the benchmark results of VAETKI-100B-A10B in comparison with an existing open MoE baseline, gpt-oss-120b (medium) [48], across a set of commonly used Korean and English benchmarks. All evaluations cover general knowledge, reasoning, instruction following under a MoE setting with different activated-parameter budgets.

On Korean benchmarks, VAETKI-100B-A10B demonstrates superior linguistic and cultural compe-

tence, outperforming the baseline on CLIcK and KoBALT. However, it shows a performance gap in structured reasoning tasks, as seen in KMMLU-Pro and HRM8K. These results indicate that while the model excels in understanding the nuances of the Korean language, further optimization is required for complex Korean reasoning.

On English general knowledge and reasoning benchmarks, VAETKI-100B-A10B exhibits a stable performance profile while maintaining a gap compared to the gpt-oss-120b baseline. Specifically, the performance disparity is more pronounced on reasoning-intensive tasks such as GPQA-Diamond and HLE, suggesting that the current MoE configuration requires further optimization for complex, multi-step logical chains. However, VAETKI-100B-A10B achieves a significant result on IFEval, scoring 86.0 and surpassing the baseline's 83.6. This performance demonstrates the model's robust instruction-following capabilities and its practical utility in real-world conversational scenarios. These results indicate that while there is room for improvement in deep reasoning, VAETKI-100B-A10B successfully balances fundamental linguistic proficiency with strong adherence to user instructions.

Overall, the performance profile of VAETKI-100B-A10B reflects a design that prioritizes Korean linguistic competence and efficient MoE scaling over competitive performance on English-centric reasoning benchmarks. Rather than aiming for state-of-the-art results on global reasoning tasks, VAETKI-100B-A10B positions itself as a language-focused foundation model, with clear opportunities for further optimization in complex reasoning tasks.

Table 4: Comparison between VAETKI-100B-A10B and an open-source MoE baseline on commonly used Korean and English benchmarks.

Benchmark (Metric)		**gpt-oss-120b (medium)**	**VAETKI -100B-A10B**
Architecture		MoE	MoE
# Total Params		117B	112B
# Activated Params		5.1B	10B
Korean benchmarks			
General	KMMLU-Pro	61.9	58.4
	CLIcK	73.0	75.5
	KoBALT	46.0	47.5
Reasoning	HRM8K	83.3	70.6
English benchmarks			
General	MMLU-Pro	79.1	71.0
Reasoning	GPQA-Diamond	73.1	53.2
	HLE (text only)	8.6	5.9
	IFBench	63.1	52.3
	IFEval	83.6	86.0

Table 5: Comparison of Safety and Alignment Performance between VAETKI-100B-A10B and gpt-oss-120b.

Benchmark (Metric)		**gpt-oss-120b**	**VAETKI -100B-A10B**
Resistance to Adversarial Attacks	XSTest (unsafe)	99.5	100
	OR-Bench (toxic)	99.7	98.1
	SaladBench	99.4	40.2
	Strong Reject	100	99.6
Response Rate to Safe Prompts	XSTest (safe)	88.0	95.7
	OR-Bench (safe)	20.2	65.9

Safety and Alignment Evaluation. In addition to task performance, we evaluate the safety and alignment properties of VAETKI-100B-A10B using a set of recently established benchmarks that reflect contemporary standards in model safety assessment. Across four widely recognized benchmarks introduced in international venues over the past two years, VAETKI-100B-A10B achieves an overall safety level comparable to that of the gpt-oss-120b baseline, as summarized in Table 5, indicating that the model maintains robust defenses against unsafe and malicious inputs under standard evaluation settings.

Beyond robustness to harmful prompts, we further assess model utility under safety constraints, focusing on the ability to respond appropriately to benign user requests without excessive refusal. On OR-Bench (safe), which measures successful task completion on safe prompts, VAETKI-100B-A10B substantially outperforms the baseline, exhibiting a markedly higher response rate. This result suggests that the model effectively balances safety enforcement with usability, reducing over-refusal while preserving correct execution of user intent in non-adversarial scenarios.

However, on benchmarks targeting highly sophisticated and compositional adversarial attacks, such as SaladBench, VAETKI-100B-A10B shows a notable performance gap relative to the baseline. This indicates that while the model is well-aligned for common unsafe inputs and everyday usage scenarios, additional alignment refinement and defensive mechanisms are required to address complex, multi-step attack patterns. These findings highlight clear directions for future work on strengthening adversarial robustness without compromising the model's demonstrated utility on safe requests.

Auxiliary Evaluation of the 7B-Scale SLM. While the primary focus of this study is the evaluation of the 100B-scale model, we additionally conduct a supplementary evaluation on a 7B-scale MoE language model to examine whether similar design principles exhibit consistent behavior in a smaller model regime. This experiment is intended as an auxiliary analysis rather than a comprehensive study, providing preliminary insight into scalability under constrained training conditions.

Table 6: Comparison among VAETKI-7B-A1B and comparable open-source MoE baselines on Korean and English benchmarks. All three models were evaluated under the same experimental setup to ensure a fair and consistent comparison.

Benchmark (Metric)		**Granite-4.0 -H-Tiny**	**OLMoE-1B-7B -0125-Instruct**	**VAETKI -7B-A1B**
Architecture		MoE	MoE	MoE
# Total Params		7B	7B	7B
# Activated Params		1B	1.3B	1.2B
# Pre-trained Tokens		23T	4.07T	1.86T
Korean benchmarks				
General	KMMLU-Pro	27.1	15.1	24.2
	CLIcK	47.7	28.2	40.6
	KoBALT	12.1	7.7	11.9
Reasoning	HRM8K	39.7	3.0	26.5
English benchmarks				
General	MMLU-Pro	41.9	14.0	34.6
Reasoning	GPQA-Diamond	27.8	30.3	27.2
	MATH500	62.4	25.0	61.8
	IFBench	21.2	18.1	20.9

The 7B-A1B model is trained under substantially tighter time and resource constraints than the 100B-A10B model, resulting in a markedly smaller pre-training corpus (1.86T tokens) compared to other models of similar scale such as Granite-4.0-H-Tiny (23T) and OLMoE-1B-7B (4.07T). Despite these limitations, the model achieves competitive performance on several Korean and English benchmarks relative to contemporaneous open MoE models, while maintaining reasonable general knowledge and reasoning capabilities. These results suggest that the core architectural and data-composition choices employed in the 100B-A10B model remain effective at smaller scales, and motivate further investigation into resource-efficient training strategies. Detailed results are provided in Table 6.

6 Conclusion

In this report, we have detailed the development process of the VAETKI series, consisting of 100B, 20B, and 7B parameter versions of foundation models optimized for the Korean language and cultural understanding. The primary objectives of this project were to establish technological self-reliance, safeguard national data sovereignty, and accelerate AI Transformation (AX) across domestic industries.

Technically, the VAETKI model family maximizes inference efficiency by integrating the Multi-Latent Attention (MLA) mechanism with Local-Global Interleaving techniques. Furthermore, it achieves efficient parameter utilization and performance scaling through a fine-grained Mixture-of-Experts

(MoE) architecture. By maintaining the proportion of Korean language data at approximately 20% in both the tokenizer and the training corpora, the models have demonstrated robust linguistic and cultural understanding on major benchmarks such as CLIcK and KoBALT. Additionally, VAETKI-100B-A10B outperforms open-source baselines on IFEval, indicating strong instruction-following capability. Ultimately, VAETKI has proven its value as a practical foundation model that effectively balances general linguistic proficiency with domain-specific knowledge, even within a constrained activated-parameter regime.

6.1 Limitations

Despite its strengths, the VAETKI-100B-A10B model developed in this study exhibits the following technical limitations:

- **Reasoning Performance Gap**: The model recorded lower scores compared to the baseline model on high-difficulty English reasoning benchmarks such as GPQA-Diamond and HLE. This indicates that further optimization is required for tasks involving complex, multi-step logical reasoning.
- **Infrastructure and Communication Bottlenecks**: Due to environmental constraints, only four InfiniBand lanes were available for tensor communication during the training process. This necessitated a hybrid strategy using Gloo as an auxiliary communication backend, which may have functioned as a potential bottleneck in communication efficiency compared to frontier-level models.
- **Limited Coverage of Domain-Specific Evaluation**: Evaluations in this report focused primarily on general benchmark performance. Performance verification for highly specific domain knowledge required in actual industrial sectors, such as manufacturing and finance, remains in the early stages.
- **Incompatibility with Context Parallelism (CP)**: The current implementation of our hybrid attention mechanism is constrained by a mismatch between the communication requirements of MLA and existing CP frameworks. While MLA necessitates a specialized P2P communication pattern to handle latent KV representations, standard CP implementations like Ring Attention are built on synchronous, circulative communication patterns. The lack of an asynchronous, window-aware communication backend in current implementation makes it difficult to integrate sliding-windowed attention without incurring prohibitive overhead. As a result, we have opted to disable the CP dimension and operate under a 4D parallelism setup, which limits our sequence length scalability compared to a full 5D parallelism configuration.

6.2 Future Works

To enhance the performance of the VAETKI models and expand the ecosystem, we plan to pursue the following research initiatives:

- **Advanced Reasoning Data and Training**: To improve high-difficulty reasoning metrics such as GPQA, we will enhance techniques for generating high-quality synthetic data and apply more rigorous verification processes to reasoning paths to ensure logical consistency.

- **Model Scaling and Multimodal Expansion**: Building on our experience developing the 100B MoE LLM, we plan to challenge the development of a 200B MoE LLM. Furthermore, we aim to extend our development into Large Multimodal Models (LMMs) by leveraging the diverse multimodal datasets held by the NC-AI Consortium.
- **Industry-Specific AX Solutions**: We will develop fine-tuned models tailored to the specific requirements of actual industrial sites, such as manufacturing, finance, and services, to provide concrete AX guidelines that enable domestic enterprises to adopt high-performance AI at a lower cost.

Acknowledgement

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-25-080043). This research was 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), S.Korea. (Grant No. 2025-AIData-WII42).

References

[1] DeepSeek-AI et al. *DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model*. 2024. arXiv: 2405.04434 [cs.CL]. URL: https://arxiv.org/abs/2405.04434.

[2] Joshua Ainslie et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. arXiv: 2305.13245 [cs.CL]. URL: https://arxiv.org/abs/2305.13245.

[3] Gemma Team et al. *Gemma 2: Improving Open Language Models at a Practical Size*. 2024. arXiv: 2408.00118 [cs.CL]. URL: https://arxiv.org/abs/2408.00118.

[4] Petar Veličković et al. *Softmax is not Enough (for Sharp Size Generalisation)*. 2025. arXiv: 2410.01104 [cs.LG]. URL: https://arxiv.org/abs/2410.01104.

[5] Ken M. Nakanishi. *Scalable-Softmax Is Superior for Attention*. 2025. arXiv: 2501.19399 [cs.CL]. URL: https://arxiv.org/abs/2501.19399.

[6] Gemma Team et al. *Gemma 3 Technical Report*. 2025. arXiv: 2503.19786 [cs.CL]. URL: https://arxiv.org/abs/2503.19786.

[7] Jay Shah et al. *FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision*. 2024. arXiv: 2407.08608 [cs.LG]. URL: https://arxiv.org/abs/2407.08608.

[8] Damai Dai et al. *DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models*. 2024. arXiv: 2401.06066 [cs.CL]. URL: https://arxiv.org/abs/2401.06066.

[9] Zihan Qiu et al. *Demons in the Detail: On Implementing Load Balancing Loss for Training Specialized Mixture-of-Expert Models*. 2025. arXiv: 2501.11873 [cs.LG]. URL: https://arxiv.org/abs/2501.11873.

[10] Jeonghoon Kim et al. "Peri-LN: Revisiting Normalization Layer in the Transformer Architecture". In: *Forty-second International Conference on Machine Learning*. 2025. URL: https://openreview.net/forum?id=ci1S6wmXf0.

[11] Team OLMo et al. *2 OLMo 2 Furious*. 2025. arXiv: 2501.00656 [cs.CL]. URL: https://arxiv.org/abs/2501.00656.

[12] NVIDIA Corporation. *Transformer Engine*. https://github.com/NVIDIA/TransformerEngine. Accessed: 2025-09-01. 2023.

[13] DeepSeek-AI et al. *DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437.

[14] Yi Liu, Tom Costello, and Sean Costello. *Zero-centered Re-parameterization of LayerNorm*. Accessed: 2025-05-31. May 2025. URL: https://ceramic.ai/blog/zerocentered.

[15] Vijay Korthikanti et al. *Reducing Activation Recomputation in Large Transformer Models*. 2022. arXiv: 2205.05198 [cs.LG]. URL: https://arxiv.org/abs/2205.05198.

[16] Jeffrey Li et al. *DataComp-LM: In search of the next generation of training sets for language models*. 2025. arXiv: 2406.11794 [cs.LG]. URL: https://arxiv.org/abs/2406.11794.

[17] Anton Lozhkov et al. *StarCoder 2 and The Stack v2: The Next Generation*. 2024. arXiv: 2402.19173 [cs.SE]. URL: https://arxiv.org/abs/2402.19173.

[18] National Information Society Agency (NIA), Republic of Korea. *AIHub: National Artificial Intelligence Data Platform*. Accessed: 2025-09-01. 2024. URL: https://www.aihub.or.kr.

[19] Zengzhi Wang et al. *OctoThinker: Mid-training Incentivizes Reinforcement Learning Scaling*. 2025. arXiv: 2506.20512 [cs.CL]. URL: https://arxiv.org/abs/2506.20512.

[20] An Yang et al. *Qwen3 Technical Report*. 2025. arXiv: 2505.09388 [cs.CL]. URL: https://arxiv.org/abs/2505.09388.

[21] Hanlin Zhang et al. *How Does Critical Batch Size Scale in Pre-training?* 2025. arXiv: 2410.21676 [cs.LG]. URL: https://arxiv.org/abs/2410.21676.

[22] Mohammad Bavarian et al. *Efficient Training of Language Models to Fill in the Middle*. 2022. arXiv: 2207.14255 [cs.CL]. URL: https://arxiv.org/abs/2207.14255.

[23] Shouyuan Chen et al. *Extending Context Window of Large Language Models via Positional Interpolation*. 2023. arXiv: 2306.15595 [cs.CL]. URL: https://arxiv.org/abs/2306.15595.

[24] Gaon An et al. "Direct preference-based policy optimization without reward modeling". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 70247–70266.

[25] Sam McCandlish et al. *An Empirical Model of Large-Batch Training*. 2018. arXiv: 1812.06162 [cs.LG]. URL: https://arxiv.org/abs/1812.06162.

[26] Pavel Izmailov et al. *Averaging Weights Leads to Wider Optima and Better Generalization*. 2019. arXiv: 1803.05407 [cs.LG]. URL: https://arxiv.org/abs/1803.05407.

[27] Shi Jie Yu and Sehyun Choi. *Parameter-Efficient Checkpoint Merging via Metrics-Weighted Averaging*. 2025. arXiv: 2504.18580 [cs.LG]. URL: https://arxiv.org/abs/2504.18580.

[28] Damai Dai et al. *StableMoE: Stable Routing Strategy for Mixture of Experts*. 2022. arXiv: 2204.08396 [cs.LG]. URL: https://arxiv.org/abs/2204.08396.

[29] Shuhe Wang et al. *Packing Analysis: Packing Is More Appropriate for Large Models or Datasets in Supervised Fine-tuning*. 2024. arXiv: 2410.08081 [cs.LG]. URL: https://arxiv.org/abs/2410.08081.

[30] Deepak Narayanan et al. *Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM*. 2021. arXiv: 2104.04473 [cs.CL]. URL: https://arxiv.org/abs/2104.04473.

[31] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[32] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[33] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744.

[34] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[35] Seokhee Hong et al. "From KMMLU-Redux to Pro: A Professional Korean Benchmark Suite for LLM Evaluation". In: *Findings of the Association for Computational Linguistics: EMNLP 2025*. Ed. by Christos Christodoulopoulos et al. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 19067–19096. ISBN: 979-8-89176-335-7. DOI: 10.18653/v1/2025.findings-emnlp.1038. URL: https://aclanthology.org/2025.findings-emnlp.1038/.

[36] Eunsu Kim et al. "CLIcK: A benchmark dataset of cultural and linguistic intelligence in Korean". In: *arXiv preprint arXiv:2403.06412* (2024).

[37] Hyopil Shin et al. "KoBALT: Korean Benchmark For Advanced Linguistic Tasks". In: *arXiv preprint arXiv:2505.16125* (2025).

[38] Hyunwoo Ko, Guijin Son, and Dasol Choi. "Understand, Solve and Translate: Bridging the Multilingual Mathematical Reasoning Gap". In: *arXiv preprint arXiv:2501.02448* (2025).

[39] Yubo Wang et al. "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 95266–95290.

[40] David Rein et al. "Gpqa: A graduate-level google-proof q&a benchmark". In: *First Conference on Language Modeling*. 2024.

[41] Long Phan et al. "Humanity's last exam". In: *arXiv preprint arXiv:2501.14249* (2025).

[42] Valentina Pyatkin et al. "Generalizing Verifiable Instruction Following". In: *arXiv preprint arXiv:2507.02833* (2025).

[43] Jeffrey Zhou et al. "Instruction-following evaluation for large language models". In: *arXiv preprint arXiv:2311.07911* (2023).

[44] Paul Röttger et al. *XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models*. 2024. arXiv: 2308.01263 [cs.CL]. URL: https://arxiv.org/abs/2308.01263.

[45] Justin Cui et al. *OR-Bench: An Over-Refusal Benchmark for Large Language Models*. 2025. arXiv: 2405.20947 [cs.CL]. URL: https://arxiv.org/abs/2405.20947.

[46] Lijun Li et al. *SALAD-Bench: A Hierarchical and Comprehensive Safety Benchmark for Large Language Models*. 2024. arXiv: 2402.05044 [cs.CL]. URL: https://arxiv.org/abs/2402.05044.

[47] Alexandra Souly et al. *A StrongREJECT for Empty Jailbreaks*. 2024. arXiv: 2402.10260 [cs.LG]. URL: https://arxiv.org/abs/2402.10260.

[48] Sandhini Agarwal et al. “gpt-oss-120b & gpt-oss-20b model card”. In: *arXiv preprint arXiv:2508.10925* (2025).