MEGREZ2 TECHNICAL REPORT

Boxun Li[1] **Yadong Li**[1] **Zhiyuan Li**[1] **Congyi Liu**[1] **Weilin Liu**[1] **Guowei Niu**[1]
Zheyue Tan[1,2] **Haiyang Xu**[1] **Zhuyu Yao**[1] **Tao Yuan**[1] **Dong Zhou**[1] **Yueqing Zhuang**[1]
Bo Zhao[2] **Guohao Dai**[3,*] **Yu Wang**[4,*]
[1] Infinigence-AI[†] [2] Aalto University [3] Shanghai Jiao Tong University [4] Tsinghua University

`https://github.com/infinigence/Infini-Megrez`

ABSTRACT

We present Megrez2, a novel lightweight and high-performance language model architecture optimized for device native deployment. Megrez2 introduces a novel cross-layer expert sharing mechanism, which significantly reduces total parameter count by reusing expert modules across adjacent transformer layers while maintaining most of the model's capacity. It also incorporates pre-gated routing, enabling memory-efficient expert loading and faster inference. As the first instantiation of the Megrez2 architecture, we introduce the Megrez2-Preview model, which is pre-trained on a 5-trillion-token corpus and further enhanced through supervised fine-tuning and reinforcement learning with verifiable rewards. With only 3B activated and 7.5B stored parameters, Megrez2-Preview demonstrates competitive or superior performance compared to larger models on a wide range of tasks, including language understanding, instruction following, mathematical reasoning, and code generation. These results highlight the effectiveness of the Megrez2 architecture to achieve a balance between accuracy, efficiency, and deployability, making it a strong candidate for real-world, resource-constrained applications.

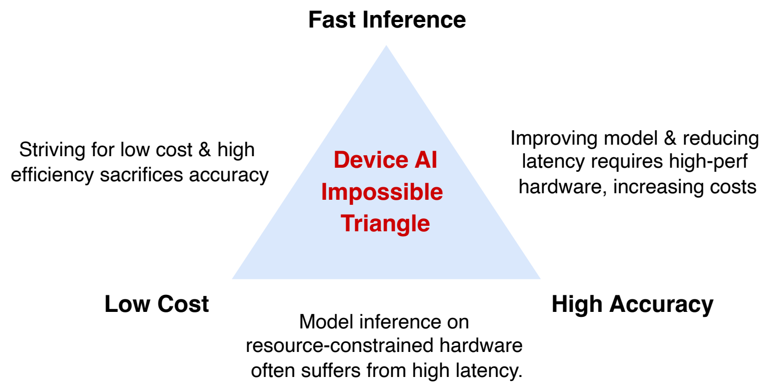
Fast Inference
Striving for low cost & high efficiency sacrifices accuracy
Device AI Impossible Triangle
Improving model & reducing latency requires high-perf hardware, increasing costs
Low Cost
High Accuracy
Model inference on resource-constrained hardware often suffers from high latency.

Figure 1: The Impossible Triangle of Device AI

1 Introduction

Large Language Models (LLMs) have progressed at an unprecedented pace, scaling to hundreds of billions of parameters and demonstrating remarkable advances toward Artificial General Intelligence (AGI). Recent foundation models such as GPT-4 [2], GPT-o3 [19], Gemini [23], Llama 4 [17], DeepSeekR1 [10], Kimi-K2 [3], and the Qwen3 series [27] have exhibited strong performance across complex tasks in multiple domains. This progress has been driven by massive

[*]Corresponding authors.
[†]The listing of authors is in alphabetical order based on their last names.

investments in data and compute, but such growth also intensifies the tension between model capacity and deployment practicality—particularly on devices with strict latency, memory, and power constraints.

Given the substantial costs involved, Mixture-of-Experts (MoE) architectures have become an increasingly attractive alternative. By dynamically activating only a subset of specialized experts per input, MoEs offer a compelling trade-off: they can match or even exceed the performance of dense models while significantly reducing inference-time computational demands. However, conventional MoE designs face notable deployment challenges on resource-constrained platforms. Sparse expert activations can lead to high memory usage and fragmented parameter utilization, thereby limiting the ability of on-device hardware to serve such models efficiently. Moreover, the design of device AI systems often encounters the *Impossible Triangle*: a three-way trade-off among speed, accuracy, and cost, where enhancing one aspect typically compromises the others. Figure 1 illustrates these inherent trade-offs in deploying models on device native hardware. Accelerating inference on constrained hardware usually demands aggressive pruning or quantization, which reduces model capacity and thereby degrades accuracy. On the other hand, increasing model size to boost accuracy leads to higher latency and greater hardware demands, which can be problematic for battery-powered or thermally constrained devices. This *device-model trilemma* delineates a design frontier where progress along one dimension often comes at the expense of the other two.

To address these challenges, we propose Megrez2, a novel language model architecture specifically designed for device native deployment. The core innovation of our approach is a **cross-layer expert sharing mechanism**: by reusing the same set of experts across multiple adjacent layers, Megrez2 significantly reduces the total parameter count while maintaining the number of activated parameters—crucial for preserving model performance. Combined with **pre-gated routing** and a novel routing strategy that encourages balanced expert utilization, our method achieves strong performance under tight computational constraints. This cross-layer sharing not only enhances parameter efficiency but also improves hardware utilization, making Megrez2 particularly well-suited for resource-constrained devices.

In the following sections, we first review related work. We then describe the Megrez2 architecture. To demonstrate the effectiveness of our approach, we present the training methodology and evaluation results of Megrez2-Preview, which serves as the first instantiation of the Megrez2 architecture. Finally, we conclude with a summary of our work.

2 Related Work

Mixture-of-Experts (MoE) models have emerged as a promising architecture for scaling large language models (LLMs) by decoupling the total parameter count from the computational cost per token. A key trend is the adoption of increasingly granular expert pools, often with shared experts across layers, enabling higher capacity and improved specialization while maintaining efficiency.

DeepSeekMoE [8] introduces a fine-grained MoE structure, segmenting each Feed-Forward-Network (FFN) into smaller experts. This enables a larger expert pool without increasing the total parameter count. By pioneering a system-level optimization stack tailored for MoEs, DeepSeekMoE achieves inference speeds up to 4.5 times faster and 9 times cheaper than comparable dense models. Additional optimization techniques such as auxiliary-loss-free routing [24] further enhance the model's routing efficiency. The Qwen series [28, 27] explores MoE architectures across various scales. For instance, Qwen1.5-MoE-A2.7B matches the performance of 7B dense models while activating only 2.7B parameters, combining 4 shared and 60 dynamic experts per MoE layer. Expanding on this, Qwen3 introduces models such as the 235B-A22B, providing 235B total parameters with only 22B active per token. These models illustrate how fine-grained MoE coupled with expert sharing significantly reduces computational cost without sacrificing accuracy. Skywork-MoE [26], initialized from a 13B dense model, employs a 146B MoE structure with 22B active parameters per token. Techniques like Gating Logit Normalization (GLN) and adaptive auxiliary losses enhance expert load balancing and specialization, positioning Skywork-MoE competitively alongside large-scale MoEs such as DBRX and Mixtral.

Deploying large language models (LLMs) on devices presents significant challenges due to limited memory and computational resources, making it impractical to run large-scale models directly. In addition to the Qwen series [27], smaller models such as Gemma [14] and Phi-4-Mini [1] offer viable alternatives tailored for memory-constrained environments. For instance, Phi-4-Mini, with its 3.8 billion parameters, demonstrates reasoning capabilities comparable to—or even surpassing—models with over 7 billion parameters. Google's Gemma 3 [14], which ranges from 1B to 27B parameters, supports quantized inference (e.g., INT4), making it suitable for single-GPU setups and mobile devices. Notably, the smaller variants, Gemma3-1B and Gemma3-4B, deliver performance on par with the previous generation Gemma2 across multiple benchmarks, illustrating that compact LLMs can offer large-model quality within an device-friendly footprint. Additionally, Megrez-3B-Instruct and Megrez-3B-Omni [15] provide strong small-scale dense models optimized for device native deployment.

Specialized optimizations are critical for deploying MoEs on devices. While MoEs offer improved performance with reduced computation, their overall memory footprint remains significant. Earlier approaches employed offloading

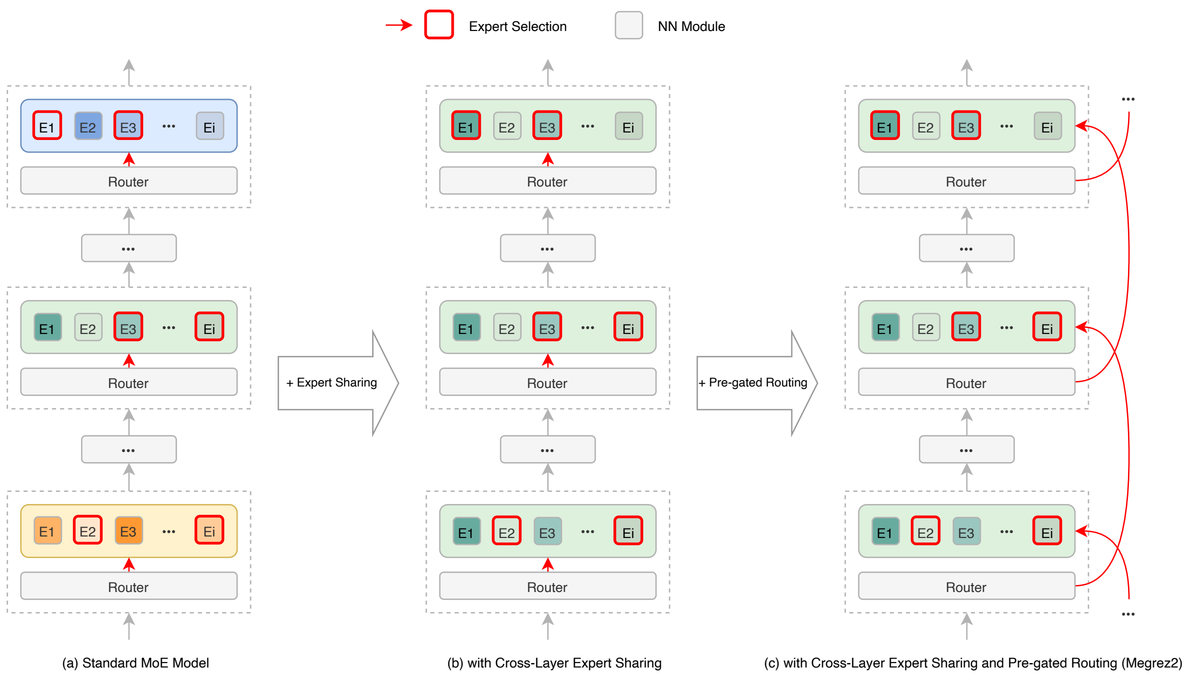
Expert Selection
NN Module
E1
E2
E3
Ei
Router
+ Expert Sharing
+ Pre-gated Routing
(a) Standard MoE Model
(b) with Cross-Layer Expert Sharing
(c) with Cross-Layer Expert Sharing and Pre-gated Routing (Megrez2)

Figure 2: Expert Sharing and Pre-gated Routing in Megrez2 Model

techniques to facilitate serving MoE models on memory-constrained devices [9], but such methods remain constrained by memory bandwidth. Pre-gated MoE [13] addresses this challenge by introducing a pre-gating function that reduces the dynamic nature of sparse expert activation, thus managing MoE's large memory footprint more effectively. Similarly, Read-ME [5] applies pre-gating strategies in its router, enabling expert-aware batching and lookahead scheduling based on expert workload dispatching, significantly reducing memory usage during MoE inference, especially beneficial for device native deployments.

3 Model Architecture

In this section, we present the architecture of Megrez2, which incorporates a novel cross-layer expert sharing strategy and pre-gated routing within the Mixture-of-Experts (MoE) framework.

3.1 Mixture-of-Experts Block

We first briefly review the widely adopted $\mathrm{top}\text{-}k$ MoE structure [27, 8, 16]. At layer i, a private gating network G_i selects, from its dedicated pool of M experts $\{E_{i,1}, \ldots, E_{i,M}\}$, the k most relevant experts ($k \ll M$). Given the layer input $\mathbf{h}_i$, the gating network produces routing scores $\mathbf{s}_i = G_i(\mathbf{h}_i)$. The aggregated hidden state h_{i+1} is then computed as

$$\mathbf{h}_i' = \sum_{j=1}^{M} \left[\mathrm{top}\text{-}k(\mathbf{s}_i)\right]_j E_{i,j}(\mathbf{h}_i), \tag{1}$$

where $\mathrm{top}\text{-}k(\cdot)$ retains only the k largest entries of $\mathbf{s}_i$, setting the others to zero. Each expert, implemented as a feed-forward network, computes $E_{i,j}(\mathbf{h}_i)$; these outputs are then combined with their corresponding routing weights to produce the block's output. For clarity, we omit the *shared experts* introduced in DeepSeekMoE [8], which are always active in every layer. As shown in Figure 2(a), the router in each layer computes the gating scores independently and selects the corresponding experts to produce the output of this block.

3.2 Cross-Layer Expert Sharing

To reduce the parameter count of a standard MoE, we share each expert's parameters across n consecutive layers, which lowers the total parameters by roughly a factor of n. Specifically, Megrez2 partitions the L-layer transformer into $G = L/n$ contiguous *groups* of length n. Every layer i in group $g = \lfloor i/n \rfloor$ *shares* the same expert pool $E_{g,1}, \ldots, E_{g,M}$ while retaining its own gating network G_i and projection weights. Layer i then updates its hidden state as

$$\mathbf{h}_i' = \sum_{j=1}^{M} \big[\text{top-}k(\mathbf{s}_i)\big]_j \, E_{g,j}\big(\mathbf{h}_i\big), \qquad g = \big\lfloor i/n \big\rfloor, \tag{2}$$

where $\mathbf{s}_i = G_i(\mathbf{h}_i)$ denotes the routing scores. As illustrated in Figure 2(b), cross-layer expert sharing reduces the number of unique parameters by roughly a factor of n while keeping the activated parameters unchanged.

3.3 Pre-gated Routing

To support deployment on devices with limited memory, Megrez2 adopts Pre-gated Routing [13]. By shifting the gating computation to the preceding layer, the model can load the parameters of the selected experts in advance, which lowers the memory cost of sparse activation. As shown in Figure 2(c), the router in the preceding layer, G_{i-1}, generates the routing decision for layer i.

$$\mathbf{h}_i' = \sum_{j=1}^{M} \big[\text{top-}k(G_{i-1}(\mathbf{h}_i))\big]_j \, E_{g,j}\big(\mathbf{h}_i\big), \qquad g = \big\lfloor i/n \big\rfloor, \tag{3}$$

Combining pre-gated routing with cross-layer expert sharing yields two practical advantages on device native deployment. First, when the router predicts the experts required by the next group of layers that do not share parameters, the design behaves like a standard pre-gated MoE, so the computation and weights loading can be overlapped and pipelined to reduce latency. Second, within a group whose layers share a common expert pool, an expert that has already been selected need not be re-loaded, allowing the cache size and replacement policy to be tuned to the available hardware. Together, these properties make it feasible to serve large MoE models in memory-constrained environments.

3.4 Architecture

In addition to the aforementioned two improvements, Megrez2-Preview adopts a dense-layer-first architecture similar to DeepSeekV2 [16]. The model consists of 31 layers in total, with the first layer being a dense layer. Every three layers form a group that shares the expert parameters of the MoE module, which includes 64 experts per group. Among these, the top-6 experts are selected via routing. Additionally, each layer incorporates 4 shared experts. The hidden size of the dense layers is set to 10,944, and each expert has a hidden dimension of 1,408. The model employs the tokenizer of Megrez-3B series [15].

4 Training Methodology

In this section, we present the training methodology of Megrez2-Preview, which is pre-trained on a diverse dataset containing 5 trillion tokens from multiple domains. Following pre-training, we perform supervised fine-tuning on millions of samples. Finally, we apply reinforcement learning with verifiable rewards to enhance the model's reasoning capabilities.

4.1 Pre-training

Pre-training dataset curation. Our dataset curation process follows a similar approach to the Megrez-3B series [15], but with significant expansions in both scale and diversity. The dataset comprises 5 trillion tokens drawn from a wide range of domains, including web text, cleaned GitHub code, STEM (Science, Technology, Engineering, and Mathematics) content, books, and synthetic reasoning data.

Multi-stage training. We adopt a three-stage training paradigm, progressively increasing the diversity and complexity of the data to enhance Megrez2-Preview's performance across various tasks.

- **Foundational Stage.** This initial stage focuses on building a robust foundation for general language modeling. The model is trained on 1.5 trillion tokens, using a sequence length of 4,096 tokens. The training data in this stage consists primarily of publicly available web text and code.
- **Knowledge & Reasoning Augmentation Stage.** To enrich the model's knowledge and reasoning abilities, this stage introduces more refined and diverse data sources, including high-quality web text, cleaned code, and reasoning tasks with detailed solution processes. We also incorporate proprietary book datasets, specialized mathematical content, and distilled long-context data. A substantial portion of this stage's data consists of synthesized knowledge and reasoning prompts. The model is further pre-trained on approximately 3 trillion tokens, maintaining the 4K sequence length. The learning rate is reduced by an order of magnitude compared to the previous stage, and a cosine annealing schedule is employed to enable fine-grained learning from premium sources.
- **Long-Context Extension Stage.** The final stage focuses on extending the model's context length to 32K tokens. Following the approach of the Megrez-3B series [15], we increase the base frequency of RoPE [22] to 1,000,000. Most of the training corpus in this stage is curated by concatenating samples with distributions similar to those in the previous stages. In addition, we introduce specialized long-context datasets to facilitate effective learning over extended contexts. This stage involves more than 600 billion tokens.

4.2 Post-training

4.2.1 Supervised Fine-Tuning

The dataset used in the Supervised Fine-Tuning (SFT) stage consists of millions of high-quality samples, meticulously curated to enhance both general conversational capabilities and specialized skills across diverse domains, such as mathematics, code generation, tabular data processing, and information extraction. Additionally, this dataset extensively incorporates distilled data and synthesized reasoning samples to refine and optimize data distributions, thereby improving model performance on common tasks and achieving closer alignment with human preferences and task-specific requirements.

The Megrez2-Preview model is fine-tuned for 2 epochs on the SFT dataset using a cosine learning rate scheduler, with an initial learning rate one order of magnitude lower than that used during the pre-training stage. Except for the learning rate, most hyperparameters remain consistent with those of the pre-training phase. To enhance training efficiency and model performance, particularly in handling longer multi-turn conversations, training samples are concatenated into fixed-length sequences of either 4K or 32K tokens. Crucially, attention masks are configured to ensure mutual invisibility between distinct samples within concatenated sequences, and positional encodings are likewise reset to prevent information leakage. This strategy enables efficient batching while maintaining the integrity of individual training instances.

Furthermore, we introduce a novel *turn-level loss* mechanism, analogous to the per-sample loss, specifically tailored for multi-turn dialogue data. In this approach, the loss for each conversational turn is computed independently and normalized by the number of tokens in that turn, rather than by the total sequence length. This *turn-level loss* strategy significantly improves model accuracy, especially in complex multi-turn conversational settings.

4.2.2 Reinforcement Learning

We employ **Reinforcement Learning with Verified Reward (RLVR)** to enhance Megrez2-Preview's reasoning abilities. Our approach begins by constructing a large-scale, open-source dataset comprising math and reasoning tasks, which undergoes rigorous filtering for both quality and difficulty. In particular, we discard questions whose answers cannot be reliably verified through string matching, and we convert multiple-choice questions into open-ended formats to minimize the likelihood of correct guesses.

To enrich data diversity, we employ a model-based synthesis pipeline to generate additional samples, while excluding simpler questions from the prompt set. To ensure a balanced difficulty distribution, we further adjust the dataset such that approximately 50% of the examples are questions the model fails to answer correctly in all attempts. The final dataset contains roughly 60,000 samples.

For reinforcement learning, we utilize a modified version of the Generalized Reinforcement Learning with Policy Optimization (GRPO) algorithm [21], replacing the original GRPO loss with a Proximal Policy Optimization (PPO) loss. We adopt Generalized Advantage Estimation (GAE) [20] with both λ and γ set to 1. The reward function is binary, assigning a positive reward for correct answers and a negative reward for incorrect ones. To encourage exploration during training, we set the model temperature to 1.0 in rollout stage and disable KL regularization.

5 Evaluation

We evaluate Megrez2-Preview across a diverse range of domains, including general language comprehension, instruction-following, mathematical reasoning, and coding tasks. For each benchmark, we compare Megrez2-Preview against strong baselines with either a comparable number of activation parameters or similar total parameter counts. These include several open-source dense models, Qwen2.5-3B, Qwen2.5-7B [28], Qwen3-4B, Qwen3-8B [27], Gemma-3-4B [14], as well as a proprietary model, GPT-4o-mini [18] (version dated 2024-07-18). Detailed evaluation results are presented in Table 1.

Table 1: Comparison among Megrez2 and other representative models.

		Megrez2	**Qwen2.5-3B**	**Qwen2.5-7B**	**Qwen3-4B**	**Qwen3-8B**	**Phi-4-mini**	**Gemma-3-4B**	**GPT-4o-mini**
# Activated Params (B)		**3.0**	3.1	7.6	4.0	8.2	3.8	4.3	-
# Stored Params (B)		7.5	3.1	7.6	4.0	8.2	3.8	4.3	-
General Tasks	C-EVAL	**91.7**	68.2	76.2	72.2	77.9	40.0	-	66.3
	MMLU-Pro	**67.6**	43.7	56.3	-	-	52.8	43.6	-
Instruction Tasks	IFEval	80.2	58.2	71.2	81.2	83.0	68.6	**90.2**	80.4
Math Tasks	MATH-500	81.6	65.9	75.5	84.8	**87.4**	64.0	75.6	78.2
	GSM8K	83.6	86.7	91.6	-	**93.2**	88.6	89.2	-
Coding Tasks	HumanEval	74.4	74.4	84.8	-	85.9	74.4	71.3	**87.2**
	MBPP	**88.0**	72.7	79.2	-	77.0	65.3	63.2	-

Overall, Megrez2 exhibits a notably efficient parameter design. With only 3B activated parameters and a total of 7.5B parameters, it delivers performance that matches or even exceeds that of substantially larger models. For instance, it stores fewer parameters than Qwen2.5-7B and Qwen3-8B, yet outperforms them in several benchmarks. This underscores Megrez2's ability to balance model capacity and computational efficiency, making it particularly well-suited for deployment in resource-constrained environments.

General Tasks. To assess general language understanding, we evaluate performance on C-EVAL (ZH) [12] and MMLU-Pro [25], using *Exact Match (EM)* as the metric. Despite using fewer activated parameters, Megrez2-Preview consistently outperforms or matches the performance of significantly larger models in these benchmarks. This reflects its strong general reasoning and language comprehension capabilities, achieved with a lightweight architecture.

Instruction Tasks. To evaluate instruction-following ability, we use IF-Eval [29] under the *Prompt Strict* setting. Megrez2 achieves highly competitive results, closely tracking or surpassing models with larger sizes. While one model achieves a higher peak score, it does so with reduced consistency across task types. In contrast, Megrez2-Preview strikes a solid balance between model size and instruction-following quality.

Math Tasks. For mathematical reasoning, we evaluate the model on MATH-500 [11] and GSM8K [7], using *Exact Match (EM)* as the evaluation metric. Megrez2-Preview demonstrates strong performance in both datasets, often rivaling or exceeding larger models. Even when slightly outperformed in some cases, it achieves competitive accuracy with far fewer parameters, showcasing its strength in numerical and symbolic reasoning.

Coding Tasks. To assess code generation, we use two widely adopted benchmarks—HumanEval [6] and MBPP [4]—with *Pass@1* as the evaluation metric. Megrez2-Preview achieves strong results relative to its size, outperforming all other models in at least one benchmark. These outcomes suggest that the model effectively captures programming logic and structure, making it well-suited for code generation tasks.

Across all evaluated domains, the Megrez2-Preview model delivers consistently strong performance, often surpassing larger models on both general and specialized tasks. These results demonstrate the effectiveness of its architecture and training methodology in achieving high performance with lower computational cost, positioning Megrez2 architecture as an efficient and versatile solution for a wide range of applications, particularly in device native scenarios.

6 Conclusion

In this technical report, we present Megrez2, a novel language model architecture that introduces expert sharing across consecutive layers and pre-gated routing within the Mixture-of-Experts (MoE) framework. We evaluate Megrez2-Preview, the first instantiation of the Megrez2 architecture, on a diverse set of benchmarks, including general language understanding, instruction following, mathematical reasoning, and code generation. Despite activating only 3B parameters per token and having a total of 7.5B parameters, Megrez2-Preview matches or even surpasses the performance of significantly larger models. It demonstrates strong generalization, solid reasoning abilities, and effective task-specific adaptation, all while requiring only a fraction of the computational cost. These results highlight the effectiveness of our approach, making it a cost-efficient solution for both research and deployment in resource-constrained environments. Overall, the balanced design of the Megrez2 architecture provides a compelling trade-off between model size and performance, making it well suited for device native deployment and a wide range of real-world applications.

References

[1] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. “Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras”. In: *arXiv preprint arXiv:2503.01743* (2025).

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).

[3] Moonshot AI. *Kimi-K2: Open-Source Models by Moonshot AI*. `https://github.com/MoonshotAI/Kimi-K2`. 2025.

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

[5] Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, Zhangyang Wang, et al. “*Read-ME*: Refactorizing LLMs as Router-Decoupled Mixture of Experts with System Co-Design”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 116126–116148.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. “Evaluating Large Language Models Trained on Code”. In: *arXiv preprint arXiv:2107.03374* (2021).

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. “Training Verifiers to Solve Math Word Problems”. In: *arXiv preprint arXiv:2110.14168* (2021).

[8] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. “Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models”. In: *arXiv preprint arXiv:2401.06066* (2024).

[9] Artyom Eliseev and Denis Mazur. “Fast inference of mixture-of-experts language models with offloading”. In: *arXiv preprint arXiv:2312.17238* (2023).

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning”. In: *arXiv preprint arXiv:2501.12948* (2025).

[11] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. “Measuring mathematical problem solving with the math dataset”. In: *arXiv preprint arXiv:2103.03874* (2021).

[12] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. “C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[13] Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. “Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference”. In: *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2024, pp. 1018–1031.

[14] Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram’e, Morgane Rivière, Louis Rouillard, et al. “Gemma 3 Technical Report”. In: *ArXiv* abs/2503.19786 (2025).

[15] Boxun Li, Yadong Li, Zhiyuan Li, Congyi Liu, Weilin Liu, Guowei Niu, Zheyue Tan, Haiyang Xu, Zhuyu Yao, Tao Yuan, et al. “Megrez-omni technical report”. In: *arXiv preprint arXiv:2502.15803* (2025).

[16] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. “Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model”. In: *arXiv preprint arXiv:2405.04434* (2024).

[17] Meta AI. *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`. Apr. 2025.

[18] OpenAI. *GPT-4o mini: advancing cost-efficient intelligence*. OpenAI blog post. Released July 18, 2024. July 2024. URL: `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`.

[19] OpenAI. *Introducing OpenAI o3 and o4-mini*. May 2025. URL: `https://openai.com/index/introducing-o3-and-o4-mini/`.

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

[21] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. "Deepseekmath: Pushing the limits of mathematical reasoning in open language models". In: *arXiv preprint arXiv:2402.03300* (2024).

[22] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024), p. 127063.

[23] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, et al. *Gemini: A Family of Highly Capable Multimodal Models*. 2025. arXiv: `2312.11805 [cs.CL]`. URL: `https://arxiv.org/abs/2312.11805`.

[24] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. "Auxiliary-loss-free load balancing strategy for mixture-of-experts". In: *arXiv preprint arXiv:2408.15664* (2024).

[25] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024". In: *URL https://arxiv. org/abs/2406.01574* (2024), p. 21.

[26] Tianwen Wei, Bo Zhu, Liang Zhao, Cheng Cheng, Biye Li, Weiwei Lü, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Liang Zeng, et al. "Skywork-moe: A deep dive into training techniques for mixture-of-experts language models". In: *arXiv preprint arXiv:2406.06563* (2024).

[27] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. "Qwen3 Technical Report". In: *ArXiv* abs/2505.09388 (2025).

[28] Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. "Qwen2.5 Technical Report". In: *ArXiv* abs/2412.15115 (2024).

[29] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. "Instruction-following evaluation for large language models". In: *arXiv preprint arXiv:2311.07911* (2023).