PiERN: Token-Level Routing for Integrating High-Precision Computation and Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Tasks on complex systems require high-precision numerical computation to support decisions, but current large language models (LLMs) cannot integrate such computations as an intrinsic and interpretable capability with existing architectures. Multi-agent approaches can leverage external experts, but inevitably introduce communication overhead and suffer from inefficiency caused by limited scalability. To this end, we propose **Physically-isolated Experts Routing Network** (PiERN), an architecture for integrating computation and reasoning. Instead of the tool-use workflows or function-calling, PiERN endogenously integrates computational capabilities into neural networks after separately training experts, a text-to-computation module, and a router. At inference, the router directs computation and reasoning at the token level, thereby enabling iterative alternation within a single chain of thought. We evaluate PiERN on representative linear and nonlinear computation-reasoning tasks against LLM finetuning and the multi-agent system approaches. Results show that the PiERN architecture achieves not only higher accuracy than directly finetuning LLMs but also significant improvements in response latency, token usage, and GPU energy consumption compared with mainstream multi-agent approaches. PiERN offers an efficient, interpretable, and scalable paradigm for interfacing language models with scientific systems.

Track: Research

1 Introduction

Decision-making processes in complex systems arising from scientific research and engineering practice often rely critically on high-precision numerical computation results (Kennedy & O'Hagan, 2002; Hennig et al., 2015). For example, power grid scheduling in modern power systems requires the accurate prediction of system loads and renewable energy generation to support reliable and optimal scheduling decisions (Hasan et al., 2025; Sharifhosseini et al., 2024; Biswal et al., 2024). Although large language models (LLMs) have recently achieved breakthrough progress in language understanding and logical reasoning, they still exhibit significant shortcomings in their intrinsic ability for high-precision numerical computation (Yang et al., 2025a). LLMs can generate seemingly reasonable chains of logic during reasoning, but once high-precision float operations, multi-step calculations, or partial differential equations (PDEs) solving are involved, they often yield incorrect or imprecise results (Huang et al., 2025; Jiang et al., 2025b). This deficiency severely constrains the applicability of LLMs in complex system decision-making scenarios (Alampara et al., 2025).

To compensate for this deficiency, existing studies have mainly pursued two approaches. The first approach is to perform end-to-end finetuning of LLMs, enabling them to directly learn numerical computation capabilities. However, it often remain insufficient when the task inputs and outputs take the form of high-dimensional numerical matrices or spatiotemporal grid solutions (e.g., PDE solution fields) (Bao et al., 2025; Spathis & Kawsar, 2024; Yang et al., 2025b), and may face catastrophic forgetting (Li et al., 2024a; Kotha et al., 2024). The second approach is based on multi-agent systems that invoke external experts, where LLMs act as the central brain (Schick et al., 2023; Wu et al., 2023; Li et al., 2025), responsible for functions such as task understanding and scheduling, while external experts are responsible for executing specific high-precision computations. Although this approach ensures the accuracy, it inevitably introduces additional communication and coordination overhead, leading to low reasoning efficiency, high response latency, large GPU energy consumption

(Chen et al., 2024b), and limited scalability in large-scale deployments. Recent work has explored directly integrating high-precision computation capabilities into language modeling (Wu et al., 2024; McLeish et al., 2024). Although these approaches demonstrate promising performance on specific tasks, they still remain difficult to extend to more complex computation–reasoning scenarios such as text-to-computation and multi-step computation, due to the lack of coupling computation with reasoning at the architectural level.

Therefore, how to faithfully and efficiently integrate high-precision numerical computation with language reasoning, has become the core scientific problem in advancing next-generation scientific intelligence systems.

To address this issue, we propose **Physically-isolated Experts Routing Network** (PiERN), and we compare the token usage and precision across representative open-source LLMs to further highlight the limitations of existing LLMs in high-precision computation–reasoning tasks. As shown in Figure 1, the two paradigms reveal complementary limitations when viewed along different axes. Along the x-axis, multi-agent systems can reach relatively high precision but only at the cost of extremely large token consumption, raising scalability concerns. Along the y-axis, fine-tuned LLMs consume fewer tokens but fail to achieve sufficient precision, showing poor stability. In contrast, PiERN simultaneously overcomes both drawbacks, achieving the highest precision with the fewest tokens and demonstrating clear advantages in efficiency and robustness.

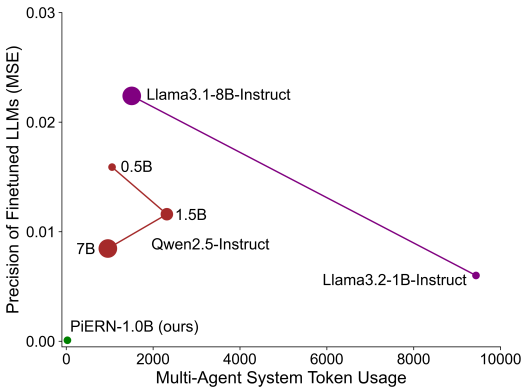

Figure 1: PiERN achieves high precision with low token usage. The horizontal axis represents the token usage of multi-agent systems with LLMs, and the vertical axis represents the precision of LLMs after finetuning.

As shown in Figure 2, the key idea of PiERN is to couple high-precision scientific computation with LLMs reasoning at the token level. Different from multi-agent approaches that rely on external function calls, PiERN internalizes expert invocation into a single reasoning chain, thereby ensuring both the high-precision of numerical computation and the efficiency and stability of reasoning-computation tasks. PiERN consists of three components: physically-isolated scientific computation experts, a text-to-computation module, and a token router, enabling dynamic expert switching and efficient coordination between high-precision computation and reasoning during inference (Appendix. A). We conducted systematic evaluations of PiERN on representative linear and nonlinear scientific computation–reasoning tasks (Sec. 2). The results show that PiERN significantly outperforms fine-tuned LLMs in prediction accuracy, and multi-agent baselines in inference cost.

2 Experiments and Results

2.1 Task and Data

We consider two representative battery-related tasks—a non-linear battery capacity prediction task and a linear battery profit calculation task—and construct task-specific numerical datasets paired with tailored language templates to support effective training of the text-to-computation module, with full details provided in Appendix B.

2.2 Performance over LLM finetuning

Setups. Finetuning is a common solution to bring domain knowledge into LLMs, thereby it can also be used to enhance LLMs with high-precision computation capabilities. We compare the MSE among the finetuned LLMs and PiERN-1.0B on the test data of both Non-Linear Task and Linear Task. For fair comparison, LLMs are finetuned by the same training data used in the training of expert models. Meanwhile, only one language template is used to generate training data to let

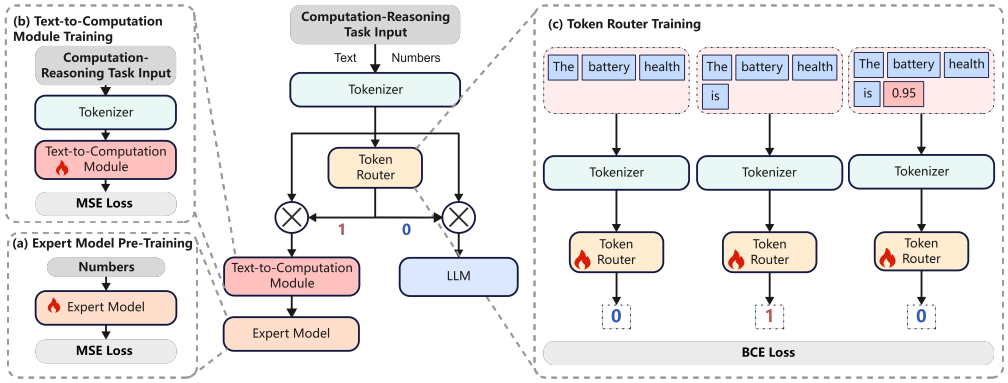

Figure 2: **(a):** Training of Expert Model for specific tasks. **(b):** Training the Text-to-Computation Module for text-computation alignment **(c):** Training the Token Router to determine experts for each token. **Middle:** The overall architecture of PiERN.

LLMs focus on computation. On the selection of LLMs, we used open-source models of various sizes, including Qwen and LLama series.

Table 1: Comparison of MSE among finetuned LLMs of different sizes and PiERN

Methods	PiERN-1.0B (Ours)	Qwen2.5			Llama	
		0.5B-Instruct	1.5B-Instruct	7B-Instruct	3.2-1B-Instruct	3.1-8B-Instruct
Non-Linear Task	0.000104	0.0159	0.0116	0.00847	0.00601	0.0224
Linear Task	0.000126	0.0712	0.0178	0.00238	0.129	0.000203

Results. Table 1 shows the accuracy of PiERN-1.0B on these two tasks is consistently better than all finetuned LLMs. Our method has the lowest MSE in all cases. The MSE of PiERN-1.0B can be one or two orders of magnitude lower, even compared with models whose parameter sizes are more than six times larger.

2.3 Performance over Multi-Agent System

Setups. We build multi-agent systems based on Qwen series and Llama series LLMs of different sizes, combined with two high-precision scientific computation experts: the battery capacity prediction expert and the battery profit calculation expert (Schick et al., 2023; Patil et al., 2025; Yao et al., 2023), on two Nvidia A800 GPUs with 80GB memory under the vLLM inference acceleration framework (Kwon et al., 2023). To highlight the performance advantages of the PiERN architecture, we design a series of comparative experiments to compare PiERN-1.0B with these multi-agent systems on the two tasks. We compare performance along four dimensions, which directly reflect the core differences between PiERN and the multi-agent systems, with detailed information of these metrics provided in Appendix C.1. More results are in Appendix C.2.

Latency. Table 2 presents the maximum, minimum, and average latency of models with different architectures and sizes on the corresponding tasks. PiERN-1.0B consistently maintains the lowest response time across all tasks. In the Non-Linear Task, the average latency of PiERN-1.0B is only 0.663s, more than 37 times faster than Qwen-1.5B-Instruct (25.04s) and nearly 60 times faster than Llama-1B (60.18s). Compared with larger models such as Qwen-32B (20.12s) and QwQ-32B (93.48s), PiERN-1.0B still maintains an advantage of one to even two orders of magnitude. In the Linear Task, the average latency of PiERN-1.0B is only 0.50s, while the latency of the multi-agent models ranges from 4.5s to over 100s. Even the baseline model with the lowest latency, Qwen-0.5B, still requires an average of 4.53s—almost 9 times slower than PiERN-1.0B. Overall, across the two tasks, PiERN-1.0B reduces latency by one to two orders of magnitude compared to current mainstream multi-agent systems, demonstrating significant and robust advantages in response speed.

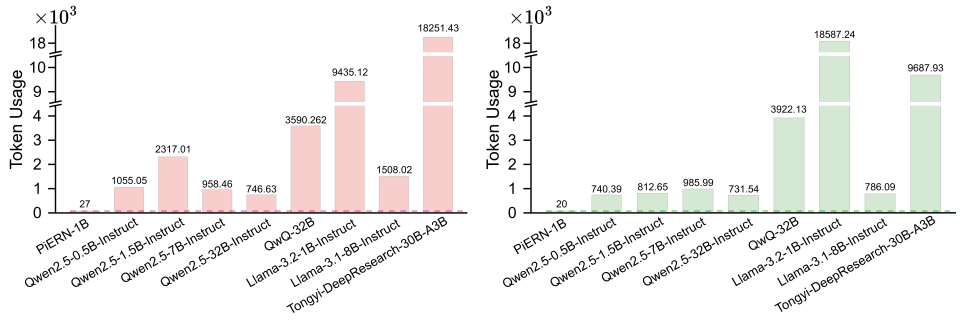

Figure 3: Token usage comparison among PiERN and multi-agent systems with LLMs: **(left)** Non-linear battery capacity estimation task, **(right)** Linear battery profit calculation task.

Table 2: Latency comparisons among PiERN and Multi-agent system baselines on Non-linear Task and Linear Task.

Architecture	PiERN (Ours)	Qwen2.5				QwQ	Llama		Tongyi-DeepResearch
	1.0B	0.5B-Instruct	1.5B-Instruct	7B-Instruct	32B-Instruct	32B	1B-Instruct	8B-Instruct	30B-A3B
Non-linear Task									
Max	**1.14**	195	315	12.3	33.7	596	320	286	358
Min	**0.632**	1.24	3.01	5.23	13.9	57.2	1.52	6.72	28.4
Avg	**0.663**	6.63	25.0	8.42	20.1	93.5	60.2	15.4	159
Linear Task									
Max	**0.525**	216	8.33	12.1	22.6	231	314	10.5	198
Min	**0.270**	1.33	3.37	6.42	18.2	67.2	3.25	6.67	21.2
Avg	**0.281**	4.53	5.21	8.87	20.2	108	106	7.71	81.2

Token Usage. As shown in Figure 3, PiERN-1.0B demonstrates significant advantages over the multi-agent system in terms of token efficiency. In the Non-Linear Task, PiERN-1.0B requires only 182 tokens on average, representing a 92% reduction compared with Qwen-1.5B (2.3k tokens) and a 98% reduction compared with Llama-1B (9.4k tokens). In the Linear Task, the average token usage of PiERN-1.0B is further reduced to 95 tokens, while Llama-1B consumes tens of thousands of tokens, yielding more than 99% savings for PiERN-1.0B. This efficiency improvement stems from the native text-to-computation architectural advantage of PiERN, which eliminates repeated context expansion and redundant cross-agent message passing. Therefore, PiERN not only achieves an order-of-magnitude reduction in user-side inference cost but also lays the foundation for the economic feasibility of large-scale deployment and applications.

3 Discussion

In this study, we propose PiERN, an architecture that unifies high-precision experts computation with LLMs reasoning. PiERN goes beyond the traditional workflow paradigm of tool invocation by enabling token-level alternating execution of expert computation and language reasoning within a single chain of thought. In computation–reasoning tasks, PiERN not only outperforms finetuned LLMs, but also significantly surpasses mainstream multi-agent methods in response latency, token usage, and GPU energy consumption. It should be noted that the statistical paradigm of current LLMs is mainly embodied in inductive reasoning and analogical reasoning, while PiERN endogenously integrates high-precision computation (deductive reasoning) into LLMs. The integration of inductive, analogical, and deductive reasoning paradigms introduces a new paradigm of intelligence. Nevertheless, the current evaluation of PiERN remains limited to specific computation–reasoning tasks. Future research will focus on three main directions: first, extending a single expert to multiple logically composable experts to support the alternating invocation of more complex high-precision computation tool call chains and language reasoning chains; second, exploring the reasoning capability of PiERN in scientific multimodal scenarios to realize a unified computation–reasoning framework across text, high-precision computation, images, equations, and code; finally, promoting the practical application of PiERN in complex system engineering tasks, such as power grid scheduling, drug discovery, and materials simulation, thereby verifying its feasibility and transformative potential as the infrastructure for next-generation scientific intelligence systems.

STATEMENT OF LLM USAGE

In our experiments, we used LLMs to assist in implementing parts of the technical pipeline code. We have carefully reviewed and verified all generated codes. In preparing the manuscript, we used LLMs to translate parts of our drafts that had been carefully prepared, and to polish the language. All generated content has been thoroughly checked by us to ensure accuracy. We take full responsibility for the validity of the research results and the final content of the paper.

REFERENCES

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. *arXiv preprint arXiv:2303.08774*, 2023.

Nawaf Alampara, Mara Schilling-Wilhelmi, Martiño Ríos-García, Indrajeet Mandal, Pranav Khetarpal, Hargun Singh Grover, N. M. Anoop Krishnan, and Kevin Maik Jablonka. Probing the limitations of multimodal language models for chemistry and materials research, 2025. URL https://arxiv.org/abs/2411.16955.

Jiajun Bao, Nicolas Boullé, Toni J. B. Liu, Raphaël Sarfati, and Christopher J. Earls. Text-trained llms can zero-shot extrapolate pde dynamics, revealing a three-stage in-context learning mechanism, 2025. URL https://arxiv.org/abs/2509.06322.

Biswajit Biswal, Subhasish Deb, Subir Datta, Taha Selim Ustun, and Umit Cali. Review on smart grid load forecasting for smart energy management using machine learning and deep learning techniques. *Energy Reports*, 12:3654–3670, 2024. ISSN 2352-4847. doi: https://doi.org/10.1016/j.egyr.2024.09.056. URL https://www.sciencedirect.com/science/article/pii/S2352484724006346.

Junyi Chen, Longteng Guo, Jia Sun, Shuai Shao, Zehuan Yuan, Liang Lin, and Dongyu Zhang. Eve: Efficient vision-language pre-training with masked prediction and modality-aware moe. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 1110–1119, 2024a.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. *arXiv preprint arXiv:2410.08115*, 2024b.

Mahmudul Hasan, Zannatul Mifta, Sumaiya Janefar Papiya, Paromita Roy, Pronay Dey, Nafisa Atia Salsabil, Nahid-Ur-Rahman Chowdhury, and Omar Farrok. A state-of-the-art comparative review of load forecasting methods: Characteristics, perspectives, and applications. *Energy Conversion and Management: X*, 26:100922, 2025. ISSN 2590-1745. doi: https://doi.org/10.1016/j.ecmx.2025.100922. URL https://www.sciencedirect.com/science/article/pii/S2590174525000546.

Philipp Hennig, Michael A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20150142, 2015.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, January 2025. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Xi Jiang, Jian Li, Hanqiu Deng, Yong Liu, Bin-Bin Gao, Yifeng Zhou, Jialin Li, Chengjie Wang, and Feng Zheng. Mmad: A comprehensive benchmark for multimodal large language models in industrial anomaly detection, 2025a. URL https://arxiv.org/abs/2410.09453.

Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li, and Dongsheng Li. Llms can find mathematical reasoning mistakes by pedagogical chain-of-thought, 2025b. URL https://arxiv.org/abs/2405.06705.

Marc C. Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 63(3):425–464, 01 2002. ISSN 1369-7412. doi: 10.1111/1467-9868.00294. URL https://doi.org/10.1111/1467-9868.00294.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference, 2024. URL https://arxiv.org/abs/2309.10105.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4297–4308, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.249. URL https://aclanthology.org/2024.findings-emnlp.249/.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025. URL https://arxiv.org/abs/2501.05366.

Xiaoyuan Li, Wenjie Wang, Moxin Li, Junrong Guo, Yang Zhang, and Fuli Feng. Evaluating mathematical reasoning of large language models: A focus on error identification and correction, 2024b. URL https://arxiv.org/abs/2406.00755.

Yunxin Li, Shenyuan Jiang, Baotian Hu, Longyue Wang, Wanqi Zhong, Wenhan Luo, Lin Ma, and Min Zhang. Uni-moe: Scaling unified multimodal llms with mixture of experts, 2024c. URL https://arxiv.org/abs/2405.11273.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *Advances in Neural Information Processing Systems*, 37: 108012–108041, 2024.

Dane Morgan and Ryan Jacobs. Opportunities and Challenges for Machine Learning in Materials Science. *Annual Review of Materials Research*, 50:71–103, July 2020. doi: 10.1146/annurev-matsci-070218-010015.

OpenAI. Hello gpt-4o, 2024. URL https://openai.com/index/hello-gpt-4o/.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: large language model connected with massive apis. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, Stefano Ermon, Yun Fu, and Ran Xu. Unicontrol: A unified diffusion model for controllable visual generation in the wild, 2023. URL https://arxiv.org/abs/2305.11147.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231591445.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Seyed Mohammad Sharifhosseini, Taher Niknam, Mohammad Hossein Taabodi, Habib Asadi Aghajari, Ehsan Sheybani, Giti Javidi, and Motahareh Pourbehzadi. Investigating intelligent forecasting and optimization in electrical power systems: A comprehensive review of techniques and applications. *Energies*, 17(21), 2024. ISSN 1996-1073. doi: 10.3390/en17215385. URL https://www.mdpi.com/1996-1073/17/21/5385.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: solving ai tasks with chatgpt and its friends in hugging face. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Dimitris Spathis and Fahim Kawsar. The first step is the hardest: pitfalls of representing and tokenizing temporal data for large language models. *Journal of the American Medical Informatics Association*, 31(9):2151–2158, 07 2024. ISSN 1527-974X. doi: 10.1093/jamia/ocae090. URL https://doi.org/10.1093/jamia/ocae090.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning, 2024. URL https://arxiv.org/abs/2403.02884.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, et al. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2507.01006.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018. URL https://api.semanticscholar.org/CorpusID:49670925.

Hanchen Wang, Tianfan Fu, Yuanqi Du, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, August 2023. doi: 10.1038/s41586-023-06221-2. URL https://doi.org/10.1038/s41586-023-06221-2. Published online 2 August 2023; Issue date 3 August 2023.

Hengkui Wu, Panpan Chi, Yongfeng Zhu, Liujiang Liu, Shuyang Hu, Yuexin Wang, Chen Zhou, Qihao Wang, Yingsi Xin, Bruce Liu, Dahao Liang, Xinglong Jia, and Manqi Ruan. Scaling particle collision data analysis, 2024. URL https://arxiv.org/abs/2412.00129.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 3(4), 2023.

Haotong Yang, Yi Hu, Shijia Kang, Zhouchen Lin, and Muhan Zhang. Number cookbook: Number understanding of language models and how to improve it, 2025a. URL https://arxiv.org/abs/2411.03766.

Liu Yang, Siting Liu, and Stanley J. Osher. Fine-tune language models as multi-modal differential equation solvers. *Neural Networks*, 188:107455, 2025b. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2025.107455. URL https://www.sciencedirect.com/science/article/pii/S089360802500334X.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models, 2024. URL https://arxiv.org/abs/2401.13601.

Yiyuan Zhang, Kaixiong Gong, Kaipeng Zhang, Hongsheng Li, Yu Qiao, Wanli Ouyang, and Xiangyu Yue. Meta-transformer: A unified framework for multimodal learning, 2023. URL https://arxiv.org/abs/2307.10802.

A PiERN METHODOLOGY

In this section, we detail the methodology of PiERN. The PiERN architecture integrates high-precision scientific computation experts and LLMs as modules in the same model. The expert integration is interpretable and controlable, supporting efficient training and inference and allowing for dynamic expansion of experts. We first give an overview of the overall architecture of PiERN, then introduce the stepwise training method for each component module, and finally present the inference paradigm of alternating invocation of different experts at the token granularity.

A.1 ARCHITECTURE OVERVIEW

As shown in Figure 2, the PiERN architecture consists of three core components: (i) a set of high-precision scientific computation experts, which are trained on domain-specific data; (ii) a text-to-computation module, which aligns the inputs of language computation task inputs with expert input representations; and (iii) a token router, which dynamically decides whether to invoke an expert or the LLM for each token.

A.2 STEPWISE TRAINING

We propose a stepwise training method that decouples the training processes of different modules in PiERN, reduces the interference between heterogeneous optimization objectives of numerical computation and natural language, thereby improving training convergence stability, and ensures high-precision, interpretability, and dynamic scalability of experts.

Stage 1: Expert Model Pre-training. As shown in Figure 2(a), for neural network based high-precision scientific computation experts, the first stage involves training on fixed numerical input-output pairs, where the data come from a scientific or industrial domain. It should be noted that this stage can be skipped for non-neural network experts. Let the training data be $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{exp}}$, where $\mathbf{x}$ denotes the input conditions and $\mathbf{y}$ denotes the corresponding ground-truth high-precision numerical computation results. The expert model f_θ approximates the true mapping by minimizing the mean squared error (MSE):

$$\mathcal{L}_{\text{exp}} = \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(\mathbf{x}_i) - \mathbf{y}_i \|^2. \tag{1}$$

After convergence, the parameters of the expert model are frozen to maintain its high-precision scientific computation capability during subsequent PiERN training and inference.

Stage 2: Text-to-Computation Module Training. In the second stage, we optimize the text-to-computation module so that it can align inputs of language-computation task with the inputs of high-precision scientific computation experts. The training data are $(\mathbf{s}, \mathbf{x}) \in \mathcal{D}_{\text{text2comp}}$, where $\mathbf{s}$ denotes the natural language computation task inputs and $\mathbf{x}$ denotes the structured numerical inputs required by the experts. The mapping function g_ϕ learns to project text inputs into numerical input representations compatible with the experts, as shown in Figure 2(b), by minimizing the MSE loss:

$$\mathcal{L}_{\text{text2comp}} = \frac{1}{N} \sum_{i=1}^{N} \| g_\phi(\mathbf{s}_i) - \mathbf{x}_i \|^2. \tag{2}$$

To further strengthen the alignment between semantics and numerical values, we optionally introduce a contrastive loss (van den Oord et al., 2018), inspired by its successful application in cross-modal representation learning such as CLIP for vision–language alignment (Radford et al., 2021):

$$\mathcal{L}_{\text{contrastive}} = -\sum_{i=1}^{N} \log \frac{\exp(\text{sim}(g_\phi(\mathbf{s}_i), \mathbf{x}_i)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(g_\phi(\mathbf{s}_i), \mathbf{x}_j)/\tau)}, \tag{3}$$

where $\text{sim}(\cdot, \cdot)$ denotes the similarity function (e.g., cosine similarity), and τ denotes the temperature coefficient. The final training objective is defined as the weighted sum of equation 2 and equation 3:

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{text2comp}} + \lambda \, \mathcal{L}_{\text{contrastive}}, \tag{4}$$

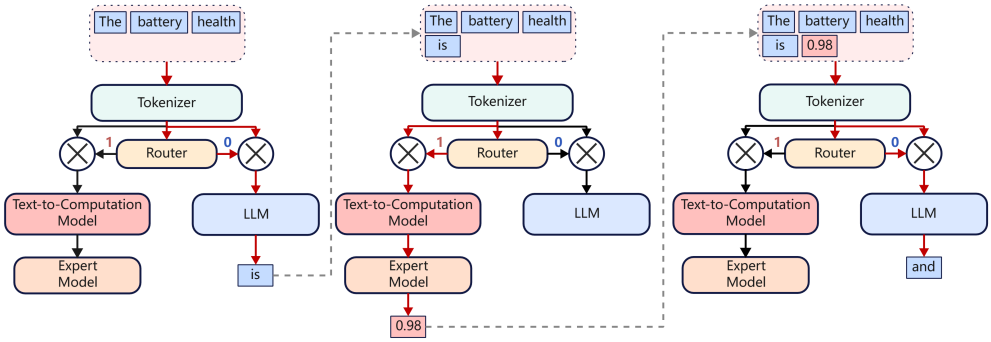

Figure 4: Token routing for reasoning-computation inference paradigm in PiERN. **Left:** Token Router decides to send tokenized inputs into LLM for generating the next token. **Middle:**Token Router decides to send tokenized inputs into the expert model for high-precision computation. **Right:** Token Router decides to send tokenized inputs with computation results to LLM for subsequent reasoning and planning.

where λ is the balancing coefficient. This joint training objective can distinguish correct and incorrect language–expert pairs, improve training efficiency, and promote the text-to-computation module to accurately regress language tasks to expert required numerical inputs.

Stage 3: Token Router Training. In the final stage, we train the token router to dynamically decide at each time step whether to invoke a high-precision scientific computation expert or the LLM. Its input is the hidden representation $\mathbf{h}_t$ of all tokens at the current time step, and its output is a probability distribution $p(e \mid \mathbf{h}_t)$ over the set of experts and the LLM $\mathcal{E}$, which indicates which expert model or the LLM should be selected in the next-token prediction process. The training data are $(\mathbf{h}_t, e_t) \in \mathcal{D}_{\text{router}}$, where e_t denotes the corresponding token-level invocation label, as shown in Figure 2(c). The router is optimized using a cross-entropy (CE) loss:

$$\mathcal{L}_{\text{router}} = -\sum_{t}\sum_{e\in\mathcal{E}} y_{t,e} \log p(e \mid \mathbf{h}_t), \tag{5}$$

where $y_{t,e}$ is a one-hot vector. In particular, when $\mathcal{E}$ contains only one expert and the LLM, this objective naturally degenerates into the binary cross-entropy (BCE) loss.

A.3 Inference Paradigm

During the inference paradigm shown in Figure 4, PiERN integrates the pre-trained expert model, text-to-computation module, and token router into a LLM via neural network connections. This integrated model is then used to execute computational tasks, and subsequent inference and planning are carried out based on the high-precision computation results. Since both the LLM and the text-to-computation module are based on Qwen2.5-0.5B, and the token router as well as the expert model are implemented as lightweight custom neural networks, the total parameter count of the current PiERN is approximately 1.0B, denoted as PiERN-1.0B.

Building on this architecture, PiERN-1.0B dynamically switches between standard language reasoning and high-precision computation of expert model. For example, given inputs “The battery health”, the token router detects no computation requirement (as the next token is likely to be ”is”) and forwards the sequence to the LLM for ordinary next-token prediction. In contrast, given “The battery health is” when a concrete numeric value is required, the router invokes the text-to-computation module to transform the sequence into expert inputs; then the expert model returns a high-precision value (e.g., 0.95), which is appended to the sequence as “The battery health is 0.95.” The computed value is seamlessly incorporated into the context, enabling the LLM to continue reasoning (e.g., “, which is in a relatively good state for daily use.”). In this way, PiERN-1.0B preserves numerical accuracy while maintaining coherent language reasoning, planning, and decision making.

B TASK AND DATA

B.1 DATA DESCRIPTION OF BATTERY CAPACITY PREDICTION TASK

The main objective of this task is to predict the remaining state of health of a battery based on time-series data of current. Battery health is a core indicator for measuring battery performance degradation, typically defined as the ratio of the current remaining capacity to the initial capacity of the battery. The battery degradation is a complex non-linear process. It is affected by multiple coupled physicochemical reactions and mechanisms such as electrode material aging, electrolyte decomposition, and solid electrolyte interphase growth, usually modelled with PDEs.

The data input include two main information. First is the time-series current data, which refers to 11 current values collected over a 2-hour period with a sampling interval of 12 minutes. Second is the time of the to-be-predicted time point and its corresponding current value. The above information constitutes 13 input feature values. In addition, the initial health status of the battery is set to 1 by default. In terms of data scale, the training dataset contains 7,200 matching samples of "current data - time point - state of health", and the test dataset contains 2,400 samples of the same type. Both of the training dataset and the test dataset are generated based on the P2D model constructed via COMSOL Multiphysics modeling. All training data are used in the training process of the expert model and the fine-tuning task of the large language model.

B.2 DATA DESCRIPTION OF BATTERY PROFIT CALCULATION TASK

The core of this task is to calculate the battery profit from arbitrage in electricity markets or electricity bill saving. Among these, the key parameters are defined as follows. α: Battery degradation coefficient; Δp: Price difference between charging and discharging; P : Battery charge-discharge power; c_a: Marginal degradation cost. This calculation can be formulated as $R = \Delta p \cdot P - \alpha \cdot c_a \cdot 1200$, a simple linear calculation task.

Data input includes four parameters: α, Δp, P, c_a. and the output is the final profit R. There are a total of 10,000 data entries, with training data accounting for 90% and the remaining being test data. Consistent with the Battery Capacity Prediction Task, all training data are used in the training process of the expert model and the fine-tuning task of the large language model.

B.3 LANGUAGE TEMPLATES AND DATA COMBINATION METHODS

To better train the text-to-computation module, we have designed multiple language templates specifically for the above two tasks. These templates are combined with the numeric data for each task for model training, enabling the text-to-computation module to better understand semantics and generate numbers accurately across various scenarios.

The specific composition methods of the language templates, task data, and large model fine-tuning data are shown in Figure 5 and Figure 6.

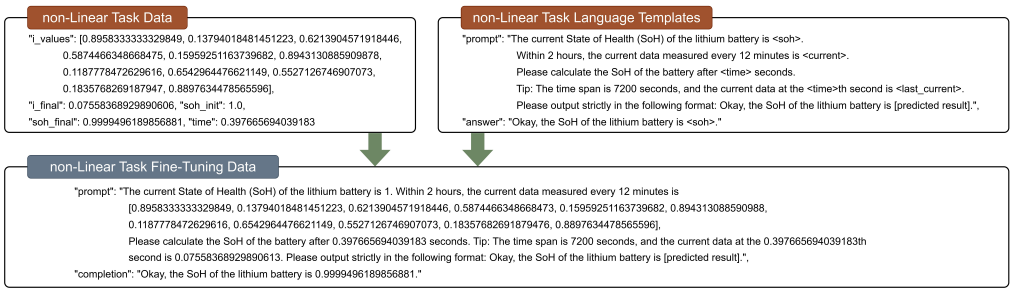

Figure 5: The combination of fine-tuning data for non-linear tasks, including time-series current data, time, battery health data, and language templates.

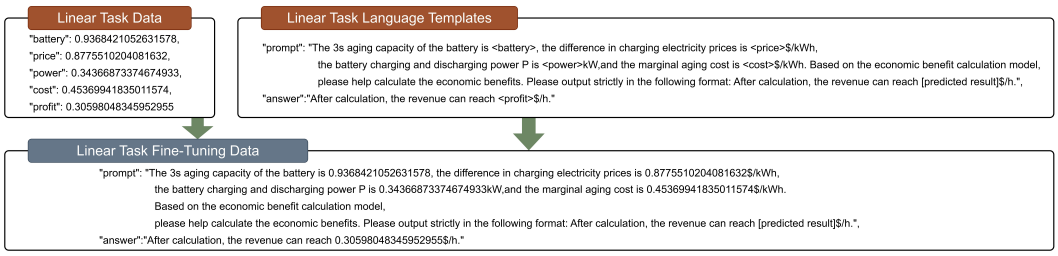

Figure 6: The combination of fine-tuning data for linear tasks, including calculation data related to profit and language templates.

C Metrics and More results for PiERN and Multi-Agent Systems

C.1 Metrics

We compare performance along four dimensions, which directly reflect the core differences between PiERN and the multi-agent systems: **(i) Latency**: determines the response speed of the system in interactive scenarios, directly affecting user experience and the feasibility of real-time decision-making tasks; **(ii) Token Usage**: measures the additional overhead in inference caused by long-context understanding and cross-agent communication, directly reflecting the user-side inference cost and the overall economic efficiency of task execution; **(iii) GPU Energy Consumption**: reflects resource utilization and energy efficiency, serving as a key metric for evaluating deployment scalability and sustainability; **(iv) Success Rate**: measures the proportion of correct high-precision results obtained in computation–reasoning tasks, directly reflecting the system's reliability and task completion capability.

C.2 More results

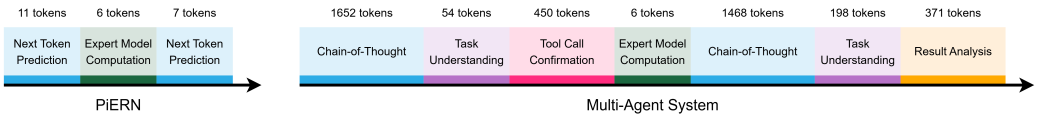

Figure 7: Token usage decomposition for PiERN and QwQ-32B based multi-agent systems on Non-Linear Task.

To further illustrate why the token usage gap between the PiERN architecture and the multi-agent system is so huge, Figure 7 presents the token usage decomposition of PiERN and the multi-agent system during the execution of a single task (Shen et al., 2023; Li et al., 2023). In PiERN, inference always alternates between next-token prediction and high-precision expert computation, thereby eliminating redundant communication overhead. In contrast, the multi-agent system must sequentially perform task understanding, tool invocation confirmation, data alignment, expert computation, and result analysis, with each step introducing additional communication and synchronization costs. This decomposition clearly reveals why PiERN can achieve significantly faster inference.

GPU Energy Consumption. As shown in Figure 8, we also report the GPU energy consumption of PiERN and LLM function calling in the Non-Linear and Linear Tasks, and the results consistently show that PiERN-1.0B is one–two orders of magnitude more efficient than the multi-agent systems. In the Non-Linear Task, PiERN-1.0B consumes only 271J on average, whereas Qwen-1.5B consumes 5.8kJ, Qwen-32B exceeds 11.7kJ, and QwQ-32B exceeds 55kJ. Even Llama-1B, whose parameter size is smaller than that of QwQ-32B, consumes more than 14.5kJ, nearly two orders of magnitude higher than PiERN. In the Linear Task, PiERN-1.0B consumes only 99J on average, while Qwen-1.5B exceeds 1.4kJ, Qwen-32B consumes 11.7kJ, Llama-1B consumes 26kJ, and QwQ-32B reaches as high as 63kJ. This means that GPU energy consumption is reduced by 93–99.8%. These results indicate that the PiERN architecture significantly reduces GPU utilization.

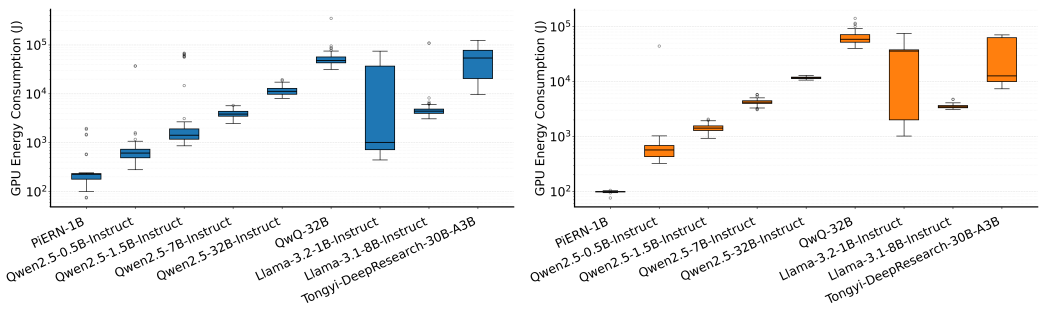

Figure 8: GPU energy consumption comparison between PiERN and multi-agent systems: **(left)** Non-linear battery capacity estimation task, **(right)** Linear battery profit calculation task.

Beyond direct energy savings, this efficiency improvement also translates into stronger scalability and lower carbon emissions.

Success Rate. As shown in Figure 9, PiERN-1.0B achieves 100% success rate on both two tasks. In contrast, multi-agent systems show instability on non-linear tasks, same-scale LLMs such as Qwen2.5-0.5B perform poorly, and even larger LLMs fail to consistently maintain 100% success.

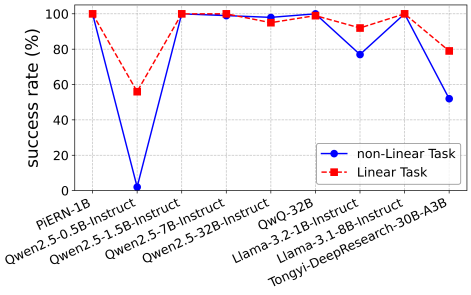

Figure 9: Success Rate between PiERN and multi-agent systems on inference tasks.

These results represent a paradigm shift: PiERN eliminates the costly external communication and coordination overhead in multi-agent systems by internally unifying the integration of high-precision scientific computation experts, the text-to-computation module, and the token router, thereby establishing a unified computation–inference workflow. The PiERN architecture demonstrates that efficiency and flexibility are not conflicting goals, but can be simultaneously achieved within the same design. By tightly coupling high-precision computation with LLMs reasoning in a single architecture, PiERN not only provides a more efficient, economical, scalable, and sustainable solution, but also outlines a blueprint for next-generation scientific intelligence systems (Wang et al., 2023; Morgan & Jacobs, 2020).

D Related Work

Multimodal Capabilities of LLMs Multimodal learning has become a central direction in today's AI, aiming to integrate text, vision, and audio within unified architectures (Zhang et al., 2024; 2023). Recent breakthroughs such as GPT-4o (OpenAI, 2024) and GLM-4.5V (Team et al., 2025) highlight these rapid progress. Mixture-of-Experts (MoE) has emerged as an efficient paradigm, demonstrating strong performance in multimodal generation, alignment, and controllable content creation (Li et al., 2024c; Chen et al., 2024a; Qin et al., 2023). Despite these advances, their utility in specialized scientific and industrial computation remains limited (Jiang et al., 2025a).

Mathematical and Reasoning Abilities LLMs have achieved impressive results in mathematical reasoning, solving problems at or beyond high-school level (Achiam et al., 2023; Hurst et al., 2024; Tang et al., 2024), but often fail at basic arithmetic and numerical consistency (Huang et al., 2025; Li et al., 2024b). End-to-end finetuning can not fundamentally improve the numerical understanding and processing abilities (NUPA). Multi-agent systems mitigate this gap by leveraging external experts through function call (Schick et al., 2023; Patil et al., 2025; Wu et al., 2023), significantly enhancing high-precision problem-solving capabilities. Yet, they still suffer from communication overhead, and limited scalability (Chen et al., 2024b).

Integrating High-Precision Computation with Language Recent attempts have explored directly embedding numerical representations into Transformers (McLeish et al., 2024; Wu et al., 2024), and benchmarks such as NUPA (Yang et al., 2025a) have been introduced to evaluate progress. Nonetheless, existing methods still struggle with multi-step calculations, solving complex nonlinear PDEs systems, and generalization beyond trained ranges, leaving the challenge of real-world high-precision computation unresolved.

E Additional Analysis on Token-Level Routing and Tool-Use Paradigms

E.1 Connection to Mixture-of-Experts

The relationship between PiERN and Mixture-of-Experts (MoE) deserves precise clarification, because the surface similarity conceals a fundamental difference.

The MoE lineage, from Jacobs (1991) to Shazeer's Sparsely-Gated MoE (2017), and later to Switch Transformer and GShard, is built on a common architectural premise: experts are internal sub-networks learned through end-to-end joint optimization. Rather than encoding pre-existing domain knowledge, they are simply parameterized modules whose functional roles arise during training. In this sense, "expert" is primarily a technical label rather than a literal indication of externally grounded expertise. Such modules generally lack a stable and interpretable identity, cannot be validated in isolation as independent components, and are not easily replaced or upgraded without re-optimizing the broader system. Their specialization is therefore better understood as an emergent consequence of training dynamics than as an explicitly designed or controlled property.

By contrast, the experts in PiERN are fundamentally different. They are physically isolated domain-specific computational units, which may be either neural or non-neural. These experts exist as independent components with well-defined functionality and verifiable behavior. For example, the FNO solver has been systematically validated on PDEBench, while GCAM is an established climate–economy simulation model. These are not merely sub-networks labeled as "experts," but genuine computational specialists with explicit physical semantics, independently verifiable accuracy, and modular replaceability.

This distinction for PiERN has three concrete consequences MoE cannot provide:

Interpretability: each expert corresponds to an identifiable scientific function. We know what it does and can audit it independently.

Stability: frozen experts do not drift during training. Their behavior is fully predictable and does not degrade.

Modularity: experts can be upgraded, replaced, or validated independently without touching the LLM.

E.2 Comparison with Agentic Tool-Use Strategies

To make the comparison with multi-agent systems more rigorous, we introduced more additional PDEBench variants. The first is Lean Function-Calling, a single-agent setting that produces direct JSON outputs without chain-of-thought, intended to approximate the lowest possible overhead of an external invocation framework. The second is Reduced CoT MAS, a multi-agent variant in which each stage contains only 1–2 sentences of brief reasoning. Together with the original AutoGen MAS baseline, which uses a five-stage chain-of-thought design, these variants cover a broad range of representative agentic tool-use configurations. The corresponding results on all three PDEBench tasks and six backbone scales are in Table 3.

Table 3: Comparison of diverse agentic tool-use strategies vs. PiERN on three PDEBench tasks. SR = Success Rate (%). Lat = Latency (s). Tok = tokens per call. PiERN model sizes include the base LLM, T2C module, and the token router.

		Diff-Reaction			Diff-Sorption			Burgers		
Method	Size	SR	Lat	Tok	SR	Lat	Token	SR	Lat	Tok
AutoGen MAS (5-stage CoT reasoning)										
	0.6B	1.5	9.8	649	14.5	7.7	1336	4	5.3	489
	1.7B	1	15.3	758	64.5	20.0	1598	25	22.5	386
	4B	93.5	16.4	485	77.5	29.5	1630	88.5	22.0	495
	8B	57.5	29.4	487	83.5	53.8	1622	92	32.8	494
	14B	97	43.8	486	92	90.0	1640	98	51.2	499
	32B	98	92.4	482	94	154.7	1655	97	96.1	499
Lean FC (no reasoning)										
	0.6B	59.5	7.7	1059	41.5	5.8	815	70.5	1.9	282
	1.7B	83	2.2	203	57.5	7.2	665	83	1.3	133
	4B	51	2.7	151	46.5	13.8	724	85	2.4	138
	8B	54.5	8.1	272	47	32.0	1096	86	3.5	134
	14B	93.5	5.9	128	53	75.7	414	86	5.9	132
	32B	95	13.3	126	52	69.8	587	90.5	13.3	133
Reduced CoT MAS (1–2 sentence brief reasoning)										
	0.6B	7	3.5	494	13	10.4	914	70	7.2	282
	1.7B	16	9.1	482	25.5	19.3	945	85	8.8	339
	4B	28.5	8.4	472	25.5	20.9	970	91	9.2	337
	8B	20	296.1	461	24.5	220.8	960	72	120.9	313
	14B	44.7	309.0	1752	90	105.5	1126	97.5	26.9	343
	32B	20	147.8	461	24.5	183.3	960	72	101.8	313
PiERN (ours)										
	2.4B	**100**	**1.0**	**20**	**100**	**2.0**	**20**	**100**	**0.7**	**20**
	3.5B	**100**	**1.9**	**20**	**100**	**3.5**	**20**	**100**	**1.4**	**20**
	5.8B	**100**	**3.7**	**20**	**100**	**7.4**	**20**	**100**	**2.8**	**20**
	9.8B	**100**	**6.3**	**20**	**100**	**12.4**	**20**	**100**	**4.7**	**20**
	15.8B	**100**	**1.9**	**20**	**100**	**2.6**	**20**	**100**	**1.1**	**20**
	33.8B	**100**	**2.9**	**20**	**100**	**4.8**	**20**	**100**	**2.0**	**20**

In brief, PiERN consistently achieves the highest success rate, the lowest token cost, and the lowest latency across different model scales and tasks, and these advantages persist regardless of how aggressively CoT is reduced in external-invocation approaches.

E.3 Gradient Conflict Between Reasoning and Tool-Use

Recent studies (e.g., LEAS/DART) have demonstrated that reasoning token gradients and tool-use token gradients are nearly orthogonal in parameter space. Forcing them through shared weights produces a seesaw effect, improving tool-calling degrades reasoning, and vice versa. This is not an artifact of particular training setups; it is a fundamental geometric property from an optimization perspective.

A common response is to separate the two functions in weight space, as in DART, which uses distinct LoRA adapters for reasoning and tool use. This helps alleviate the interference, but it does not fully resolve the underlying issue. Even under such a design, the LLM must still express tool parameters as text tokens. The burden of mapping linguistic intent to structured numerical inputs therefore remains within the language generation process itself. In this sense, the conflict is reduced, but not removed.

A parallel line of work on token-level routing (e.g., FusionRoute) discovers the value of routing at the decoding step rather than at the sequence level. But these approaches route between language model variants, they never cross the text-to-computation modality boundary, and they have no mechanism for cross-modal alignment. They find the right level of granularity, but stop short of the right abstraction.

E.4 Degradation Experiments Under High-Dimensional Numerical Context

To make the above problem more concrete, we conducted two controlled degradation experiments, directly measuring how LLM native routing and text-based parameter transmission degrade under high-dimensional numerical context.

Experiment A: Routing degradation. We constructed a 6-solver routing task where the LLM must identify the correct PDE expert from physically similar descriptions. After the task description, we appended N random floats to simulate real MAS invocation context. Routing accuracy results are in Table 4.

Table 4: Routing accuracy (%) vs. numerical context length N (6-solver PDE task, 120 trials each).

N (floats)	0.6B	1.7B	4B	8B	14B	32B
0 (control)	83	100	100	100	100	100
128	75	100	100	100	100	100
512	82	100	100	100	100	100
1024	83	100	100	100	100	100
2048	83	86	100	100	100	100
3000	48	84	100	100	100	100
4000	32	82	92	98	100	100

Routing degradation is systematic and scale-dependent. For 1.7B, accuracy is perfect at $N = 1024$ but drops to 86% at $N = 2{,}048$. For 0.6B, accuracy collapses to 48% at $N = 3{,}000$, and the dominant failure mode shifts from *wrong expert* to *no call*, which the model abandons the tool-call format entirely and outputs free text.

Experiment B: Parameter transmission accuracy degradation. We evaluate the accuracy of LLMs in generating parameters when numerical inputs are transformed through linguistic semantics and passed to an expert model. Specifically, the model is given an array of N random floats together with an arithmetic operation. It must first interpret the operation at the semantic level, and then output the transformed numerical results as text, thereby simulating the "language intent $\rightarrow$ numerical parameters" conversion in a typical MAS invocation.

We report the **full-match rate**, defined as the fraction of outputs for which all N elements have relative error within 0.1%. The metric is averaged over four operations of increasing complexity ($\times 2.0$, $\div 3.0$, $+1.5$, $\times 0.5 - 0.25$), with 30 trials per operation. Full-match rate reflects the operational reliability of this process: in a real expert invocation, a single incorrect parameter can invalidate the entire computation.

Table 5: Full-match rate (%) for numerical array transmission vs. array length N.

N (floats)	0.6B	1.7B	4B	8B	14B	32B
5	43	53	66	66	75	75
10	17	41	54	53	73	73
20	1	5	47	48	62	73
50	0	0	7	32	53	66
100	0	0	0	13	48	51
200	0	0	0	0	40	40
500	0	0	0	0	2	12

The full-match rate shows that text-based parameter transmission deteriorates rapidly as the parameter dimension increases. It drops to 0% at $N = 20$ for 0.6B and 1.7B, at $N = 100$ for 4B, and at $N = 200$ for 8B; even for 14B and 32B, it is already close to 0% at $N = 500$. The reason is straightforward: if the per-element accuracy is p, then the probability that all elements are simultaneously correct is p^N, which decays exponentially with N. Larger models can only delay this degradation, but cannot eliminate it.

E.5 PiERN's Architectural Solution

PiERN removes the cross-modal alignment burden from the LLM entirely. Once the token router fires, the LLM produces zero numerical parameter tokens. The Text-to-Computation (T2C) module handles the mapping from linguistic semantics to expert input space directly in the hidden-state geometry. The LLM's gradient contains only $\nabla L_{\text{reasoning}}$. There is no $\nabla L_{\text{tool\_use}}$ in the LLM at all. This is not weight-space decoupling; it is the complete architectural elimination of the conflict.

It is worth being precise about what "T2C" represents at the architectural level. The module is not specifically about "text-to-computation" as a modality pair; it is about having a dedicated alignment module between any two computationally incompatible domains that need to be bridged. The specific case of T2C is one instance. The general principle is: whenever a router separates two computational paths, a dedicated alignment module is needed to handle the cross-domain translation without delegating it back to the language generation process. This is precisely what the token-level routing works lack — they find the right routing granularity but miss the need for an explicit alignment module, and as a result still implicitly force the LLM to handle the translation through generation.

Zero MMLU/GLUE degradation across six backbone scales is the direct empirical signature of this elimination. A system that still generates tool parameters through language (whether via plain tool-calling or DART-style LoRA) cannot provide this guarantee without additional regularization effort.

This separation also has a direct implication for training efficiency that weight-space approaches cannot match. In PiERN's architecture, three components can be optimized independently: the LLM via reinforcement learning on reasoning trajectories, T2C via supervised learning on language-to-expert-parameter alignment, and the expert remains frozen throughout. There is no gradient interference between these paths by construction.

This opens a broader design question: for any system pairing language reasoning with specialized domain computation, the critical question is not how do we teach the LLM to call the tool, but how do we build an architecture so the LLM never has to. PiERN is one answer. As scientific AI systems grow more complex, more experts, higher-dimensional state spaces, and tighter latency requirements, this architectural separation becomes not an optimization but a necessity.