

Small Language Model Tool-use Accuracy and Throughput Optimization for the Jetson AGX Orin

Bence Danko
dept. of applied data science
San Jose State University
San Jose, USA
bence.danko@sjsu.edu
0009-0000-5824-9296

Abstract—The deployment of agentic Artificial Intelligence on edge hardware is currently constrained by the trade-off between computational latency and reasoning capability. This preliminary survey proposes a comparative analysis of Small Language Models (SLMs), specifically focusing on recent sparse Mixture-of-Experts (MoE) architectures, to optimize tool-use throughput and accuracy on the NVIDIA Jetson AGX Orin. We examine the performance of emerging SLMs—including gpt-oss-20b, Qwen3-Coder-Next, and Nemotron 3 Nano—to determine their viability in resource-constrained, low-latency environments. The proposed methodology integrates QLoRA fine-tuning with rigorous ablations of prompt engineering frameworks (ReAct vs. ReflAct) and token-efficient serialization formats, contrasting standard JSON against Token-Oriented Object Notation (Toon). Performance is evaluated against the Berkeley Function Calling Leaderboard (BFCL) and tau2-bench, with specific attention to Time-to-First-Token (TTFT), decode rates, and power efficiency across the Orin’s variable thermal envelopes. This work aims to establish a methodology for maximizing agentic tool-calling density on consumer-grade embedded hardware.

Index Terms—small language models, mixture of experts, tool calling, Jetson AGX Orin, prompt engineering

I. SUMMARY

We intend to optimize and compare several small language models, specifically recently developed sparse mixture-of-expert models, on the Jetson AGX Orin [1]:

- gpt-oss-20b [2]
- Qwen3-Coder-Next [3]
- Nemotron 3 Nano 30B-A3B [4]

We will compare baseline performance on BFCL [5] and tau2-bench [6], two performance benchmarks for agentic tool-calling. We will perform ablations on prompting strategies, and QLoRA fine-tuning on ToolMind.

II. DATASET

ToolMind [7] consists of 160k synthetic data instances generated using over 20k tools, and 200k augmented open-source data sources.

An original synthetic portion contains 160k high-quality instances created using agent simulations, and judge pruning for higher quality samples [7]. ~200k samples are from open

source projects [8]–[14]. All are standardized to the same format.

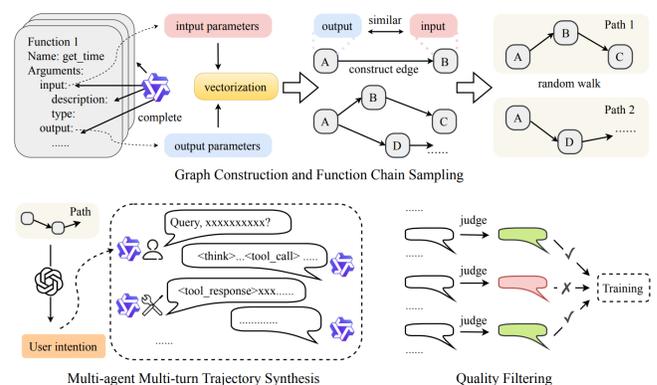


Fig. 1. ToolMind synthesis process. Source: Toolmind

In total, there are 368,611 tool calling chain samples. See Appendix A to see a sample. Data entries include:

- the pool of available tools
- the user request,
- the model thinking block
- tool evocation chain that satisfies the request.

III. BACKGROUND AND LITERATURE REVIEW

Despite the engineering overhead in sparse MoE training and tuning, recent work has argued that small language models (SLMs) are essential for agentic and tool-oriented workloads, citing advantages in lower latency, operational cost, reduced computational requirements, and task performance [15]. In our work we are hoping to demonstrate tool-use proficiency of small, sparse MoE models on efficient hardware, which will justify these claims. We are also hoping to test and demonstrate efficient prompting and tool use frameworks to optimize for efficient token use.

A. Sparse Mixture of Experts Fine Tuning

Mixture of Experts (MoE) architecture has recently undergone a revival under the domain of Large Language Models (LLMs), and particularly sparse MoEs have emerged

The author is with the Department of Applied Data Intelligence at San Jose State University.

in mainstream popularity [16]. However, training and fine-tuning large-scale MoE models to specific domains and tasks have been widely documented to prove difficult, raising issues such as router collapse, training divergence, and routing inefficiencies [17]–[21].

Recent developments in fine tuning frameworks such as Unsloth directly aim to address sparse MoE fine-tuning [22], though stability is primarily achieved by simply freezing the router. Techniques such as selective PEFT expert tuning [23] have emerged. Large-scale testing and ablations for these techniques on modern MoE architectures, on capable edge hardware still require contribution and experimentation.

B. Prompt Engineering

Prompt engineering in agentic tool-using cases have been documented to come in planning, tool-selecting, calling, and response generation workflow stages [24]. ReAct [25] was foundational in introducing a widely accepted paradigm for contextual engineering, but there is still ongoing research being done into alternatives such as ReflAct, which introduces reasoning on reflecting on the agent’s state relative to its goal [26]. However, we hope to optimize in our experiments not just accuracy and enhanced reasoning, but the efficiency of those reason tokens and reducing times to action and evocation.

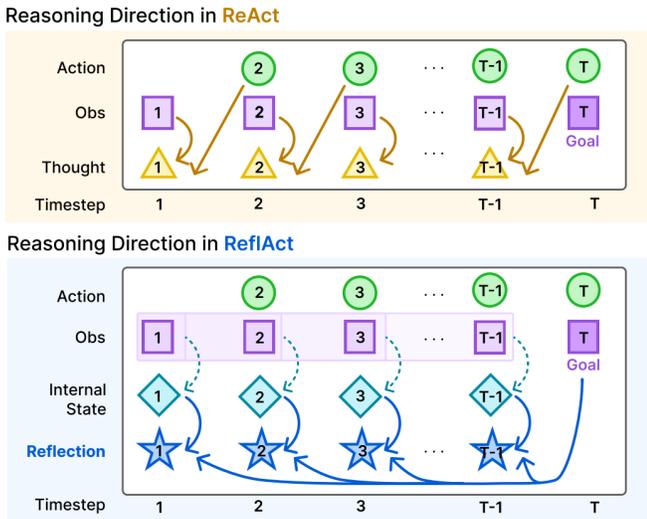


Fig. 2. ReAct versus ReflAct. Source: ReflAct.

MedReason [27] addresses efficient reasoning research on smaller models. By engineering Tree-of-Thought trees offline, and pruning for shortest-path reasonings, they created a high-quality reasoning dataset. By pruning for the shortest path, this allows for training on straightforward reasoning chains while still offering explainability [27], and still indirectly benefiting from the exploratory nature of ToT outputs. Synthetic Toolmind data also used a similarly-spirited approach, using offline chain-preparation. Random tool-walks are first generated, explained by an LLM agent role. These are presented to an LLM user-agent role that determines successful goal completion. A second

LLM-judge evaluation prunes for coherent toolchaining paths and associated reasoning [27]. In one limitation Toolmind data is not particularly curated for the most straightforward or efficient tool-calling pathways, relying mainly on synthetic pass-fail judgements by the two LLM judges.

ToolMind makes tool calls using JSON [7]. There is a lack in experimentation with possibly superior tool-calling paradigms, such as Natural Language Tools [28], and Toon (Token-Oriented Object Notation) a notation that can use up to ~40% less tokens for the same JSON representation [29]. There is yet to be significant work demonstrating ablations on these techniques.

C. Hardware: AGX Orin

The AGX Orin is manufactured using the Ampere architecture [1]. This offers exceptional performance on INT8 and INT4 operations and quantization methods. However, it’s important to consider that FP4 is likely a superior alternative for future researchers [30], and that our main contribution lies in the data preparation for such superior consumer hardware. The AGX Orin is only capable of 275 FLOPs on its highest setting, and despite the 64GB high-memory availability, quantization is necessary for low-latency performance.

IV. TECHNICAL APPROACH

A. Prompt Engineering

We intend to explore ablations on tool-calling techniques for our various models, including

- JSON-based (ToolMind [7])
- Natural Language Tools [28],
- Toon (Token-Oriented Object Notation) [29]

In addition, we intend to test various agentic frameworks for tool calling, and perform comparisons for efficiency and benchmark accuracy:

- ReAct [25]
- ReflAct [26]

In order to implement ReflAct, we can further synthesize ToolMind. Though the dataset contains valid tool-calling chains which are invaluable, we can further synthesize the dataset to include state-reflection. We can use superior frontier models to prepare state-awareness, and thus potentially yield the performance gains indicated by the ReflAct framework.

B. Fine-Tuning Techniques

Frameworks like Unsloth have developed to specialize in MoE fine-tuning, most recently to accelerate LoRA training times [22]. We will take advantage of these frameworks to help us conduct the fine-tuning process. However, most effort in quality tuning comes from ensuring clean and quality data representations.

V. PERFORMANCE EVALUATION

ToolMind itself was evaluated on BFCL [5], tau-bench [31], and tau2-bench [6]. We’ll be evaluating on the same benchmarks in order to establish consistent and comparable baselines our own developments.

TABLE I
BASELINE METRICS TEMPLATE

Model	TTFT ^a (m/s)	PreFill ^b (tok/sec)	Decode ^c (tok/sec)	Accuracy
x	x	x	x	x

^a Time to first token.

^b How fast a model can load the context.

^c How fast a model can create new tokens

Baseline metrics to run for the base MoE models.

We hope to construct these ablations specifically that will demonstrate performance on the AGX Orin:

These will be carried out on 15W, 30W, 45W and 60W modes.

VI. REFERENCES

REFERENCES

- [1] NVIDIA, "Nvidia jetson agx orin technical brief," 2022. [Online]. Available: <https://www.nvidia.com/content/dam/en-zz/Solutions/gtcfd21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf>
- [2] O. S. A. Team, "gpt-oss-120b and gpt-oss-20b release card," *arXiv preprint arXiv:2508.10925*, 2025. [Online]. Available: <https://arxiv.org/abs/2508.10925>
- [3] Q. Team, "Qwen3-coder-next technical report," 2026. [Online]. Available: https://github.com/QwenLM/Qwen3-Coder/blob/main/qwen3_coder_next_tech_report.pdf
- [4] NVIDIA, "Nvidia-nemotron-3-nano-technical-report," 2025. [Online]. Available: <https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Nano-Technical-Report.pdf>
- [5] G. O. F. Team, "BFCL v4: Berkeley function calling leaderboard with web search," https://gorilla.cs.berkeley.edu/blogs/15_bfcl_v4_web_search.html, 2024, accessed: 2026-02-25.
- [6] S. Research, "Tau2-bench: Next-generation tool-use benchmarking," 2025. [Online]. Available: <https://github.com/sierra-research/tau2-bench>
- [7] T. Authors, "ToolMind: A comprehensive benchmark for tool-use in llms," *arXiv preprint arXiv:2511.15718*, 2025. [Online]. Available: <https://arxiv.org/abs/2511.15718>
- [8] A. Authors, "APIGen: Automated pipeline for generating high-quality datasets for tool-use," *arXiv preprint arXiv:2406.18518*, 2024. [Online]. Available: <https://arxiv.org/abs/2406.18518>
- [9] Glaive AI, "Glaive function calling V2," 2023. [Online]. Available: <https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2>
- [10] C. Zhang *et al.*, "ToolACE: Deterministic tool use with trace-level post-training," *arXiv preprint arXiv:2409.00920*, 2024. [Online]. Available: <https://arxiv.org/abs/2409.00920>
- [11] N. Author *et al.*, "When2Call: Optimizing tool usage in large language models," *arXiv preprint arXiv:2504.18851*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.18851>
- [12] C. Yuan *et al.*, "BUTTONInstruct: Bounding User-defined Tasks with Objective Navigation," 2024. [Online]. Available: <https://arxiv.org/abs/2410.12952>
- [13] L. Zheng *et al.*, "APIGen-MT-5k: A multi-turn API execution dataset," 2025. [Online]. Available: <https://arxiv.org/abs/2504.03601>
- [14] T. Zhu *et al.*, "Tau-bench: A Benchmark for Tool-learning Agents in Real-world Scenarios," 2024. [Online]. Available: <https://arxiv.org/abs/2406.12045>
- [15] N. Research, "Small language models (slms) are the future," *arXiv preprint arXiv:2506.02153*, 2025. [Online]. Available: <https://arxiv.org/abs/2506.02153>
- [16] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang, "A survey on mixture of experts in large language models," *IEEE Transactions on Knowledge and Data Engineering*, 2025. [Online]. Available: <https://arxiv.org/pdf/2407.06204>
- [17] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022. [Online]. Available: <https://arxiv.org/abs/2101.03961>
- [18] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus, "St-moe: Designing stable and transferable sparse expert models," 2022. [Online]. Available: <https://arxiv.org/abs/2202.08906>
- [19] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017. [Online]. Available: <https://arxiv.org/abs/1701.06538>
- [20] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.16668>
- [21] C. Hwang, W. Cui, Y. Xiong, Z. Yang, Z. Liu, H. Hu, Z. Wang, R. Salas, J. Jose, P. Ram, J. Chau, P. Cheng, F. Yang, M. Yang, and Y. Xiong, "Tutel: Adaptive mixture-of-experts at scale," *arXiv preprint arXiv:2206.03382*, 2022. [Online]. Available: <https://arxiv.org/abs/2206.03382>
- [22] U. AI, "Unsloth specialized moe fine-tuning," 2026. [Online]. Available: <https://unsloth.ai/docs/new/faster-moe>
- [23] Y. T. et al., "Exploring expert concentration for parameter-efficient fine-tuning of mixture-of-expert llms," *OpenReview*, 2025. [Online]. Available: <https://openreview.net/forum?id=zBgjWTWgCh>
- [24] C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J.-R. Wen, "Tool learning with large language models: A survey," *arXiv preprint arXiv:2405.17935*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.17935>
- [25] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: <https://arxiv.org/abs/2210.03629>
- [26] J. Kim, S. Rhee, M. Kim, D. Kim, S. Lee, Y. Sung, and K. Jung, "Reflect: World-grounded decision making in llm agents via goal-state reflection," *arXiv preprint arXiv:2505.15182*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.15182>
- [27] M. Authors, "MedReason: Reasoning-focused medical large language models," *arXiv preprint arXiv:2504.00993*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.00993>
- [28] R. T. Johnson, M. D. Pain, and J. D. West, "Natural language tools: A natural language approach to tool calling in large language agents," *arXiv preprint arXiv:2510.14453*, 2025. [Online]. Available: <https://arxiv.org/abs/2510.14453>
- [29] TOON Format Contributors, "Token-oriented object notation (toon)," <https://github.com/toon-format/toon>, 2025. [Online]. Available: <https://github.com/toon-format/toon>
- [30] onekq, "Nvfp4 vs int4," Hugging Face Blog. [Online]. Available: <https://huggingface.co/blog/onekq/nvfp4-int4>
- [31] S. Research, "Tau-bench: A benchmark for tool-use agents," 2024. [Online]. Available: <https://github.com/sierra-research/tau-bench>

APPENDIX A
EXAMPLE TOOLMIND ENTRY

```
{
  "conversations": [
    {
      "role": "user",
      "content": "Could you help me ensure a flight status for route KY1230 on string
        → \"Monday\" and search for a person named \"John Doe\" in the US?
        → Additionally, could you retrieve information about an auditor named \"KPMG
        → Global\" in the Business Software domain?"
    },
    {
      "tool_calls": [
        {
          "function": {
            "name": "ensure-flight",
            "arguments": {
              "route_id": "KY1230",
              "string": "Monday"
            }
          }
        },
        {
          "function": {
            "name": "Search Person by Name",
            "arguments": {
              "q": "John Doe",
              "countryCode": "us"
            }
          }
        },
        {
          "function": {
            "name": "Get Auditor Information",
            "arguments": {
              "roac": "KPMG Global"
            }
          }
        }
      ],
      "role": "assistant",
    }
  ]
}
```

```

"content": "<think>\nOkay, let's break down what the user is asking for. They
→ have three main requests here. First, they want to ensure the flight status
→ for route KY1230 on the string \"Monday\". Second, they need to search for a
→ person named \"John Doe\" in the US. Third, they want information about an
→ auditor named \"KPMG Global\" in the Business Software domain.\n\nStarting
→ with the flight status. Looking at the tools provided, there's an API called
→ \"ensure-flight\" which requires route_id and string parameters. The user
→ provided route KY1230 and string \"Monday\", so that's straightforward. The
→ passenger_count has a default of 1, so even if they didn't mention it, we
→ can use the default. So I'll call ensure-flight with route_id KY1230 and
→ string Monday.\n\nNext, searching for \"John Doe\" in the US. The tool
→ \"Search Person by Name\" needs a query (q) and optionally a countryCode.
→ The user specified US, which is the default, but it's safe to include it. So
→ parameters would be q: \"John Doe\", countryCode: \"us\".\n\nThird,
→ retrieving auditor info for \"KPMG Global\". The tool \"Get Auditor
→ Information\" requires roac, which is the auditor's name or ID. The user
→ provided \"KPMG Global\", so roac should be that.\n\nI need to make sure all
→ required parameters are present for each tool. For ensure-flight, route_id
→ and string are required, which are given. For Search Person, q is required,
→ which is provided. For Get Auditor, roac is required. All set.\n\nNo
→ parameters are missing, so three separate tool calls are needed here. Let me
→ structure each in the correct JSON format as specified.\n</think>"
}
],
"tools": [
{
  "type": "function",
  "function": {
    "name": "System Availability Check",
    "description": "This API checks the availability of systems to ensure
→ everything is operational.",
    "parameters": {
      "type": "dict",
      "properties": {
        "system_id": {
          "description": "The unique identifier of the system to check",
          "type": "string"
        },
        "timeout": {
          "description": "The time in seconds to wait for the system to respond",
          "type": "int"
        }
      }
    },
    "required": [
      "system_id"
    ]
  },
  "required": null
},
{
  "type": "function",
  "function": {
    "name": "Search Person by Name",
    "description": "Search for a person by name and returns a list of matching
→ persons with additional information",

```

```

"parameters": {
  "type": "dict",
  "properties": {
    "q": {
      "description": "The name of the person to search for",
      "type": "string"
    },
    "countryCode": {
      "description": "Optional parameter to filter results by country code",
      "type": "string",
      "default": "us"
    }
  },
  "required": [
    "q"
  ]
},
"required": null
}
},
{
  "type": "function",
  "function": {
    "name": "ensure-flight",
    "description": "This API ensures the flight schedule and availability for a
→ given route and string.",
    "parameters": {
      "type": "dict",
      "properties": {
        "route_id": {
          "description": "The unique identifier of the flight route.",
          "type": "string"
        },
        "string": {
          "description": "The string for which the flight schedule is required.",
          "type": "string"
        },
        "passenger_count": {
          "description": "The number of passengers for which the flight schedule
→ is required.",
          "type": "int",
          "default": 1
        }
      },
      "required": [
        "route_id",
        "string"
      ]
    },
    "required": null
  }
},
{
  "type": "function",
  "function": {
    "name": "Get Auditor Information",

```

```

"description": "Retrieve information about auditors from the Business Software
→ domain.",
"parameters": {
  "type": "dict",
  "properties": {
    "roac": {
      "description": "The name or identifier of the auditor or group of
→ auditors.",
      "type": "string"
    }
  },
  "required": [
    "roac"
  ]
},
"required": null
}
},
{
  "type": "function",
  "function": {
    "name": "create_client",
    "description": "Create a new client in the Business Software domain",
    "parameters": {
      "type": "dict",
      "properties": {
        "client_email": {
          "description": "Email address of the new client",
          "type": "string"
        },
        "client_lastname": {
          "description": "Last name of the new client",
          "type": "string"
        },
        "client_firstname": {
          "description": "First name of the new client",
          "type": "string"
        },
        "client_pass": {
          "description": "Password for the new client",
          "type": "string"
        },
        "client_login": {
          "description": "Login credentials for the new client",
          "type": "string"
        }
      },
      "required": [
        "client_email",
        "client_lastname",
        "client_firstname",
        "client_pass",
        "client_login"
      ]
    },
    "required": null
  }
}

```

```

},
{
  "type": "function",
  "function": {
    "name": "Get Finance Rates",
    "description": "Retrieves a list of finance rates for a specific location.",
    "parameters": {
      "type": "dict",
      "properties": {
        "loc": {
          "description": "The location for which to retrieve finance rates.",
          "type": "string",
          "default": "94105"
        }
      }
    },
    "required": [
      "loc"
    ]
  },
  "required": null
}
},
{
  "type": "function",
  "function": {
    "name": "Get User Document Status Summary",
    "description": "Retrieves a summary of a user's document status within the
    ↪ business software domain.",
    "parameters": {
      "type": "dict",
      "properties": {
        "user_id": {
          "description": "Unique identifier of the user",
          "type": "string"
        },
        "document_id": {
          "description": "Unique identifier of the document",
          "type": "string"
        }
      }
    },
    "required": [
      "user_id",
      "document_id"
    ]
  },
  "required": null
}
}
]
}

```