INTERPRETING AND IMPROVING SUPERVISED FINE-TUNING FOR TOOL USE IN LLM AGENTS VIA INTERACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Supervised fine-tuning (SFT) improves the tool-use capabilities of large language model (LLM) agents, yet it remains unclear why tool-use capability varies with the SFT-based method, the model scale, and the amount of SFT data. With Harsanyi interactions, we investigate whether a common mechanism underlies these factors: for the two stages of tool calling, tool selection and argument generation, we decompose the LLM's output score into interaction effects among input words, where complex interactions involve many words and simple interactions involve only a few. We find that SFT, more effective SFT-based methods, larger model scales, and more SFT data all tend to strengthen the LLM's modeling of complex interactions and weaken its modeling of simple interactions. Motivated by this common mechanism, we propose Interaction Complexity Regularization (ICR), a plug-and-play auxiliary loss that directly enhances the strength of complex interactions while reducing that of simple interactions during SFT and can be added to existing SFT-based tool-use methods for tool selection and argument generation. Experiments with various backbones from five LLM families on three tool-use benchmarks, BFCL, StableToolBench, and τ^2-bench, show that ICR enhances SFT across model families and SFT-based tool-use methods.

1 INTRODUCTION

Tool use extends large language models (LLMs) beyond text generation, enabling them to act as agents that retrieve information, execute operations, and complete tasks in external environments. ReAct (Yao et al., 2023) demonstrates the effectiveness of interleaving reasoning with actions, while Toolformer (Schick et al., 2023) shows that language models can learn when to invoke APIs, which arguments to provide, and how to incorporate tool responses. These capabilities depend on two essential steps: selecting an appropriate tool and generating arguments that satisfy the user's request and the tool's specification. Improving both steps is central to building effective LLM agents.

Recent work typically relies on supervised fine-tuning (SFT) to improve the tool-use capabilities of LLM agents. Some studies construct new training data by collecting real-world APIs at scale (Qin et al., 2024) and synthesizing accurate, complex, and diverse data (Liu et al., 2025). Yet what a model learns from such data also depends on how the data is presented to it. Therefore, other studies propose transformations that can be applied to existing data to make it more effective for SFT, such as masking function names (Lin et al., 2025) and representing tools as dedicated tokens (Wang et al., 2025). Moreover, the effectiveness of SFT for tool use also depends on the model scale and on the amount of SFT data: the same method yields higher accuracy on larger models (Lin et al., 2025; Liu et al., 2025), and the same backbone reaches higher accuracy when it is fine-tuned on more data (Xu et al., 2026). However, it remains unclear why tool-use capability varies with factors including the SFT-based method, the model scale, and the amount of SFT data.

To this end, in this paper, we investigate an important problem: whether a common mechanism underlies how different factors enhance the tool-use capabilities of LLM agents. For example, do different SFT-based methods all strengthen the LLM's modeling of specific inference patterns? Recent studies have used Harsanyi interactions between input words to characterize the fine-grained inference patterns underlying the predictions of an individual LLM (Zheng et al., 2025; Wen et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

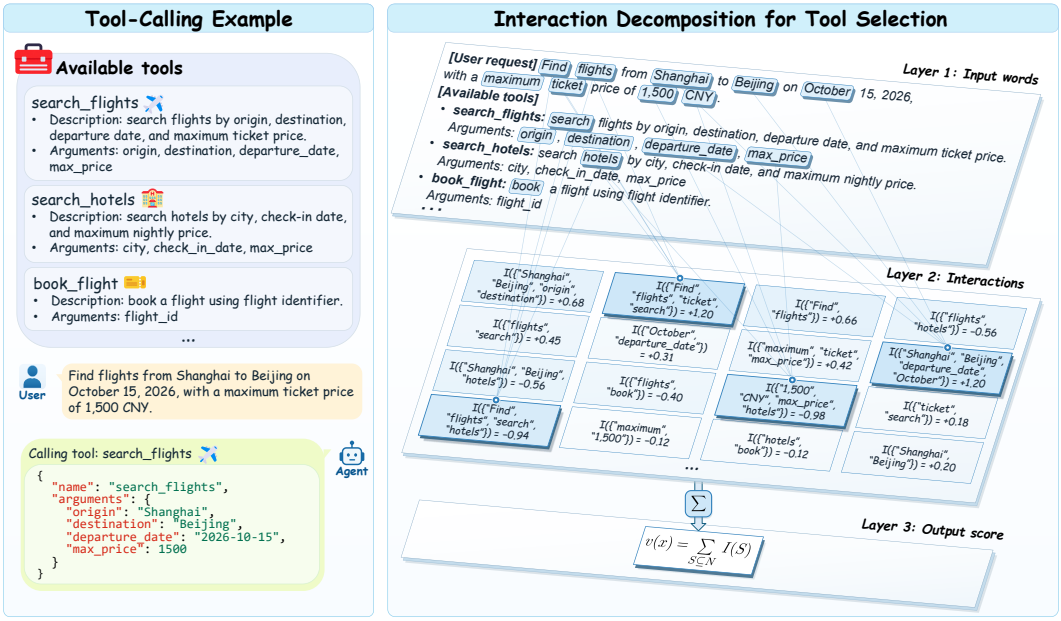


Figure 1: **Using interactions to explain tool selection.** Given the available tools and a user request, the agent calls `search_flights`. We explain this tool selection with the interaction decomposition framework, which proceeds in three layers. Layer 1 takes the words in the tool list and the user request as input variables. Layer 2 extracts the interactions among these words, where each effect $I(S)$ measures how the joint presence of the word set S pushes the agent toward or away from `search_flights`. Layer 3 shows that the sum of all active interaction effects equals the model's output score $v(x)$, which can be proved from Equation (2). Please see Appendix A.1 for the proof. *The same analysis applies to argument generation.*

2025; Chen et al., 2026; Qin et al., 2026). Inspired by these studies, we extend interaction analysis to tool selection and argument generation in LLM agents. As Figure 1 shows, given an example of tool calling, we extract interactions between input words in order to interpret how the joint presence of these words drives the tool selection and argument generation of an agent. For example, the interaction effect among the input words "*Find*," "*flights*," "*ticket*," and "*search*" pushes the agent to select the `search_flights` tool. It has been theoretically proven (Ren et al., 2023) that the output score of a deep neural network can be decomposed into a massive number of interactions between input variables. In this way, we can consider interactions between input words as the detailed inference patterns used by an LLM during tool selection or argument generation.

Based on interactions, we obtain the following three findings. First, compared with their initial checkpoints before SFT, LLMs fine-tuned with different SFT-based methods all encode stronger high-order complex interactions and weaker low-order simple interactions; this phenomenon is more pronounced for more effective methods. Here, a complex interaction involves many input words, and a simple interaction involves only a few. Second, the same phenomenon usually appears when comparing larger LLMs with smaller ones in the same model family. Third, it also appears when comparing the same backbone fine-tuned on more SFT data with that fine-tuned on less. In this way, we believe that the common mechanism behind the factors mentioned above is that **LLMs with better tool-use performance usually encode stronger complex interactions and weaker simple interactions**.

Motivated by these findings, we propose **Interaction Complexity Regularization (ICR)**, a plug-and-play auxiliary loss that enhances the strength of complex interactions while reducing the strength of simple interactions during SFT. We incorporate ICR as an auxiliary loss into the original SFT objective and apply it to both tool selection and argument generation, using the corresponding output scores to guide training at each tool-calling stage. Experiments on various backbones from five LLM families and three benchmarks demonstrate improvements across model families and SFT-based tool-use methods for both tool selection and argument generation.

2 EXTRACTING INTERACTIONS FOR LLM AGENTS

Recently, Wang et al. (2026) proposed a method to analyze how interactions between input words affect an LLM's confidence in generating a given target sequence. We extend this method to analyze how interactions between input words affect tool selection and argument generation in LLM agents. Without loss of generality, we take tool selection as an example.

Given an input text x with n words to be analyzed, indexed by $N = \{1, \dots, n\}$, and an LLM with parameters θ, let $y^* = (y_1^*, \dots, y_L^*)$ denote the L-token sequence representing the target tool selection. We aim to analyze how combinations of input words[1] in N jointly affect the LLM's generation of y^*. Let $v(x) \in \mathbb{R}$ denote the LLM's confidence score for generating the correct tool selection y^*, defined as

$$v(x) = \log \frac{\bar{p}(y^* \mid x)}{1 - \bar{p}(y^* \mid x)}, \qquad \bar{p}(y^* \mid x) = \exp\left(\frac{1}{L}\sum_{\ell=1}^{L} \log p_\theta(y_\ell^* \mid x, y_{<\ell}^*)\right), \tag{1}$$

where $p_\theta(y_\ell^* \mid x, y_{<\ell}^*)$ denotes the conditional probability assigned by the LLM to the ℓ-th target token y_ℓ^* given the input x and the preceding target tokens $y_{<\ell}^* = (y_1^*, \dots, y_{\ell-1}^*)$. In this way, $\bar{p}(y^* \mid x) \in (0, 1)$ is the geometric mean of the target-token probabilities.

Harsanyi Interactions. Ren et al. (2023) adopted the Harsanyi dividend (Harsanyi, 1963) to quantify the effect $I(S)$ of the interaction among input variables in a set $S \subseteq N$, as follows:

$$I(S) = \sum_{T \subseteq S} (-1)^{|S|-|T|} v(x_T), \tag{2}$$

where x_T denotes the masked input in which all the words in $N \setminus T$ are masked[2] and all the words in T are kept unchanged. A nonempty interaction represents an AND relationship among the words in S: it is activated only when all words in S are present (unmasked) in x_T, and masking any of them deactivates it. For example, in Figure 1, the interaction among "*Shanghai*," "*Beijing*," "*departure_date*," and "*October*" contributes positively to the confidence score for selecting `search_flights`, whereas the interaction between "*flights*" and "*hotels*" contributes negatively to the same score. In the special case $S = \emptyset$, Equation (2) reduces to $I(\emptyset) = v(x_\emptyset)$, where $x_\emptyset$ represents the input in which all the words in N are masked.

Theorem 1 (Universal matching property)**.** *Let $\phi(x_T)$ denote the surrogate logical model based on interactions, as follows:*

$$\phi(x_T) = v(x_\emptyset) + \sum_{\emptyset \neq S \subseteq N} \mathbf{1}[S \subseteq T]\, I(S), \tag{3}$$

where $\mathbf{1}[S \subseteq T]$ indicates whether the AND relationship of S is activated in x_T. Under the setting of Equation (2), this model satisfies $\phi(x_T) = v(x_T)$ for every $T \subseteq N$, that is, it matches the LLM's output score exactly on all 2^n masked inputs (Ren et al., 2023).

Interaction Order and Strength. The order of an interaction is $m = |S|$, and its strength is measured by $|I(S)|$. We define the normalized average strength of interactions of order m as

$$\kappa(m) = \frac{\mathbb{E}_x \mathbb{E}_{S \subseteq N,\, |S|=m}[|I(S)|]}{Z}, \tag{4}$$

where $Z = \frac{1}{n}\sum_{m'=1}^{n} \mathbb{E}_x \mathbb{E}_{S \subseteq N,\, |S|=m'}[|I(S)|]$ is a normalization term. Strength concerns the magnitude of an interaction effect, independently of its sign. If $\kappa(m)$ at high orders is larger than that at low orders, the LLM tends to encode complex interactions among many input words rather than simple interactions among only a few words.

[1] Following Wen et al. (2025); Wang et al. (2026); Qin et al. (2026), we take words rather than tokens as input variables, because different LLMs may split the same word into different tokens. Treating all tokens of a word as a single variable keeps interactions comparable across LLMs with different tokenizers.

[2] Following Wang et al. (2026), we employ the standard masking technique that replaces each masked word with a dedicated [MASK] token at the embedding level. Please see Appendix B for an introduction to masking strategies and for the [MASK] embedding used for each LLM.

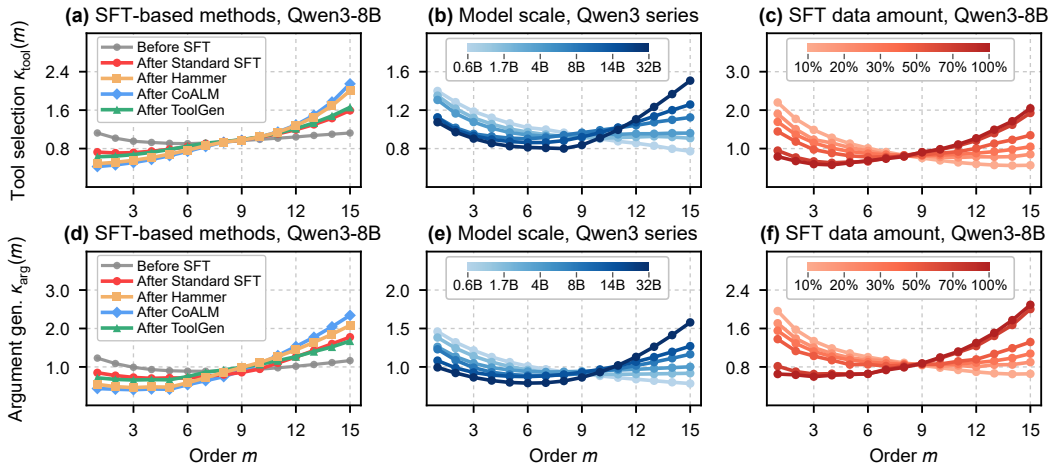


Figure 2: **Normalized average interaction strength $\kappa(m)$ versus the order m for tool selection (top) and argument generation (bottom).** (a, d) Qwen3-8B before SFT and after SFT with four methods. (b, e) The Qwen3 series from 0.6B to 32B parameters before SFT. (c, f) Qwen3-8B after SFT on 10%, 20%, 30%, 50%, 70%, and 100% of the 20,000-example controlled SFT set under a fixed optimizer-step budget.

3 Explaining Tool-Use Performance via Interactions

3.1 Explaining the Factors behind Tool-Use Performance

In this section, we compare the interactions that an LLM models for tool selection and argument generation under different conditions, in order to reveal the common mechanism of how the SFT-based method, the model scale, and the amount of SFT data affect tool use. We first define the output scores of the two tool-calling stages. We define $v_{\text{tool}}(x)$ by Equation (1) with y^* being the target tool identifier, and $v_{\text{arg}}(x)$ by the same equation with y^* being the target arguments conditioned on the gold tool. Applying Equations (2) and (4) to the two scores yields two sets of interactions, $I_{\text{tool}}(S)$ and $I_{\text{arg}}(S)$, and two strength curves over the order m, $\kappa_{\text{tool}}(m)$ and $\kappa_{\text{arg}}(m)$. For each example, we select $n = 15$ meaningful words for tool selection and argument generation, respectively, and compute the interactions between them; the remaining words are treated as fixed, unmasked background context. The data, checkpoints, and training configurations are given in Appendix C.

Exploring the Effects of SFT-Based Methods on Interactions. Figures 2(a) and 2(d) show $\kappa_{\text{tool}}(m)$ and $\kappa_{\text{arg}}(m)$ of Qwen3-8B (Yang et al., 2025) before SFT and after SFT with four types of SFT methods, including Standard SFT, Hammer (Lin et al., 2025), ToolGen (Wang et al., 2025), and CoALM (Acikgoz et al., 2025). All four methods start from the same initial checkpoint, so the backbone is held fixed while the method varies (data in Appendix C). We find that all four SFT methods make the LLM encode weaker simple interactions and stronger complex interactions than its initial checkpoint, that is, κ becomes lower at low orders and higher at high orders for both tool selection and argument generation. Figure 7 in Appendix D shows the same phenomenon on Llama-3.1-8B (Grattafiori et al., 2024), Mistral-7B-v0.3 (Jiang et al., 2023), OLMo-2-7B (Team OLMo et al., 2024), and DeepSeek-LLM-7B (DeepSeek-AI et al., 2024).

Exploring the Effects of Model Scale on Interactions. Figures 2(b) and 2(e) show $\kappa_{\text{tool}}(m)$ and $\kappa_{\text{arg}}(m)$ of the Qwen3 series from 0.6B to 32B parameters before SFT. These models share the model series and the training recipe, so the series is held fixed while the parameter count varies. Larger models encode stronger complex interactions and weaker simple interactions than smaller ones in the same series, that is, κ at high orders increases with the parameter count, whereas κ at low orders decreases. Figure 8 in Appendix D shows the same phenomenon for Llama 2 (Touvron et al., 2023) (7B–70B), Ministral 3 (Liu et al., 2026) (3B–14B), OLMo 2 (1B–32B), and DeepSeek-R1-Distill (Guo et al., 2025) (1.5B–14B).

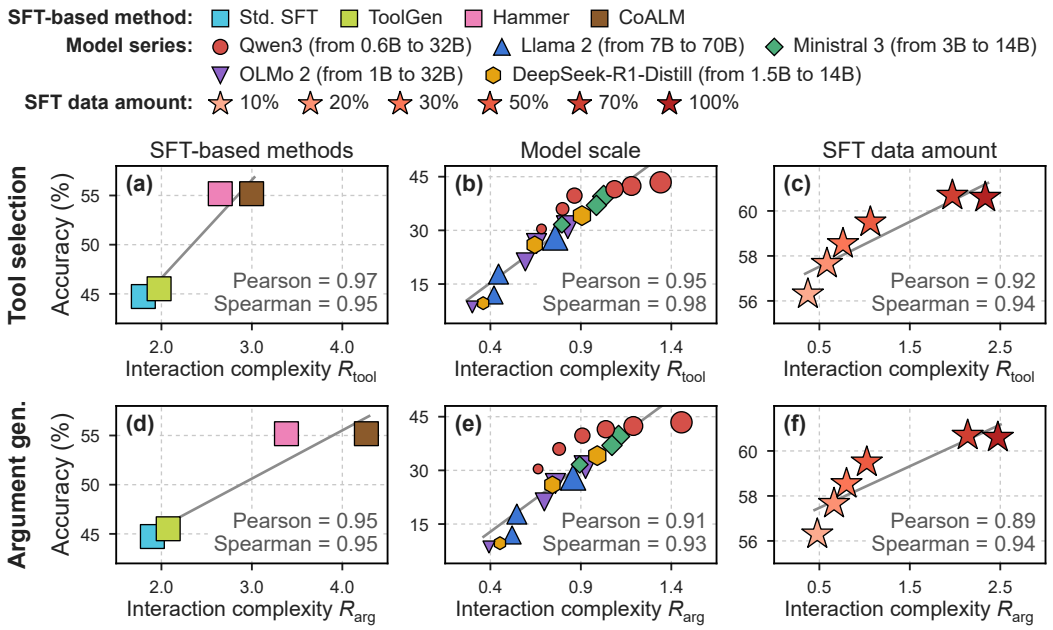


Figure 3: **BFCL v3 overall accuracy versus interaction complexity for tool selection (top, R_{tool}) and argument generation (bottom, R_{arg}).** (a, d) Qwen3-8B after SFT with four methods (Std. SFT: Standard SFT). (b, e) Models of five series before SFT; colors denote the series, and larger markers denote models with more parameters. (c, f) Qwen3-8B after SFT on 10%–100% of the 20,000-example controlled SFT set.

Exploring the Effects of the Amount of SFT Data on Interactions. Figures 2(c) and 2(f) show $\kappa_{\text{tool}}(m)$ and $\kappa_{\text{arg}}(m)$ of Qwen3-8B after SFT on 10%, 20%, 30%, 50%, 70%, and 100% of the 20,000-example controlled SFT set, *i.e.*, 2,000 to 20,000 examples, which contains multi-turn trajectories and no-call examples (Appendix C). The subsets are nested and share the backbone, the recipe, and the optimizer-step budget, so the backbone and the method are held fixed while the amount of data varies. Models fine-tuned on more data encode stronger complex interactions and weaker simple interactions than those fine-tuned on less data, that is, κ at high orders increases with the amount of data, whereas κ at low orders decreases. Table 7 in Appendix D lists the accuracy of these checkpoints.

3.2 Exploring the Common Mechanism behind Tool-Use Improvements

Interaction-Based Metrics for Analyzing Tool Use. Based on the phenomenon in Section 3.1, to compare the strength of the high-order interactions with that of the low-order interactions encoded by an LLM, we define the **Interaction Complexity** R_{tool} and R_{arg} as follows:

$$R_{\text{tool}} = \frac{\mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{high}})}\left[\kappa_{\text{tool}}(m)\right]}{\mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{low}})}\left[\kappa_{\text{tool}}(m)\right]}, \qquad R_{\text{arg}} = \frac{\mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{high}})}\left[\kappa_{\text{arg}}(m)\right]}{\mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{low}})}\left[\kappa_{\text{arg}}(m)\right]}, \tag{5}$$

where $\mathcal{O}_{\text{low}} = \{1, \ldots, \lceil n/3 \rceil\}$ and $\mathcal{O}_{\text{high}} = \{\lceil 2n/3 \rceil + 1, \ldots, n\}$ denote the sets of low orders and high orders, *i.e.*, the orders of simple interactions and complex interactions, respectively. Taking tool selection as an example, a large value of R_{tool} indicates that the LLM encodes stronger complex interactions and weaker simple interactions for tool selection, that is, the interactions used by the LLM are more complex. We therefore refer to R_{tool} and R_{arg} as the interaction complexity of the LLM for the two tool-calling stages.

The Relationship between Interaction Complexity and Tool-Use Accuracy. Figure 3 compares the tool-use accuracy of LLMs on the BFCL v3 dataset with their interaction complexity R_{tool} and R_{arg} across the four SFT-based methods on Qwen3-8B, across 19 models from five series with 0.6B to 70B parameters, and across Qwen3-8B checkpoints fine-tuned on 10%–100% of the SFT data. In

both tool selection and argument generation, the accuracy increases with the interaction complexity whether the SFT-based method, the model scale, or the amount of SFT data varies.

Common Mechanism across the Three Factors. Across the three factors, we observe the same phenomenon: **SFT makes an LLM encode stronger complex interactions and weaker simple interactions than its initial checkpoint** (Figure 2a,d), and the same holds for larger models in the same series (Figure 2b,e) and for SFT on more data (Figure 2c,f). Consistent with these findings, Figure 3 shows that **tool-use accuracy increases with interaction complexity across variations in the SFT-based method, model scale, and amount of SFT data**. We regard this phenomenon as the common mechanism behind the three factors, and use it to guide SFT in Section 4.

4 GUIDING SFT VIA INTERACTIONS

Inspired by the discovered common mechanism, we propose to design a plug-and-play auxiliary loss that directly enhances the strength of complex interactions and reduces the strength of simple interactions during SFT, while retaining the original SFT supervision. We term this loss **I**nteraction **C**omplexity **R**egularization (**ICR**) and apply it to both tool selection and argument generation.

4.1 INTERACTION COMPLEXITY REGULARIZATION

Let N_{tool} and N_{arg} index the input words for tool selection and argument generation, respectively. For each tool-calling stage, we define the average strength of the low-order simple interactions and that of the high-order complex interactions encoded by the LLM as

$$\begin{aligned} \mathcal{C}_{\text{tool}}^{\text{low}} &= \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{low}})}\mathbb{E}_{\substack{S\subseteq N_{\text{tool}}\\|S|=m}}\big[|I_{\text{tool}}(S)|\big], \quad \mathcal{C}_{\text{tool}}^{\text{high}} = \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{high}})}\mathbb{E}_{\substack{S\subseteq N_{\text{tool}}\\|S|=m}}\big[|I_{\text{tool}}(S)|\big], \\ \mathcal{C}_{\text{arg}}^{\text{low}} &= \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{low}})}\mathbb{E}_{\substack{S\subseteq N_{\text{arg}}\\|S|=m}}\big[|I_{\text{arg}}(S)|\big], \quad \mathcal{C}_{\text{arg}}^{\text{high}} = \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{high}})}\mathbb{E}_{\substack{S\subseteq N_{\text{arg}}\\|S|=m}}\big[|I_{\text{arg}}(S)|\big]. \end{aligned} \tag{6}$$

The proposed ICR is given as follows.

$$\mathcal{L}_{\text{ICR}}^{\text{tool}} = \mathcal{C}_{\text{tool}}^{\text{low}} - \mathcal{C}_{\text{tool}}^{\text{high}}, \qquad \mathcal{L}_{\text{ICR}}^{\text{arg}} = \mathcal{C}_{\text{arg}}^{\text{low}} - \mathcal{C}_{\text{arg}}^{\text{high}}. \tag{7}$$

Minimizing $\mathcal{L}_{\text{ICR}}^{\text{tool}}$ and $\mathcal{L}_{\text{ICR}}^{\text{arg}}$ suppresses simple interactions and enhances complex interactions in tool selection and argument generation, respectively.

To improve training efficiency, we adopt a sampling method to approximate $\mathcal{L}_{\text{ICR}}^{\text{tool}}$ and $\mathcal{L}_{\text{ICR}}^{\text{arg}}$ by following Zhang et al. (2021) and Wang et al. (2026). Specifically, given N_{tool}, we partition N_{tool} into b disjoint blocks $B_1, \ldots, B_b$, such that $\bigcup_{j=1}^{b} B_j = N_{\text{tool}}$ and $B_j \cap B_{j'} = \emptyset$, and let $U(K) = \bigcup_{j\in K} B_j$ for any $K \subseteq [b] = \{1, \ldots, b\}$. Based on this, we define the approximated loss term $\mathcal{L}_{\text{ICR}}^{\prime\,\text{tool}}$ as follows.

$$\mathcal{L}_{\text{ICR}}^{\prime\,\text{tool}} = \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{low}})}\,\mathbb{E}_{|U(K)|=m}\big[|J_{\text{tool}}(K)|\big] - \mathbb{E}_{m\sim\mathcal{U}(\mathcal{O}_{\text{high}})}\,\mathbb{E}_{|U(K)|=m}\big[|J_{\text{tool}}(K)|\big], \tag{8}$$

where $J_{\text{tool}}(K)$ denotes the interaction among the blocks in K (Appendix A.2). *The approximated term $\mathcal{L}_{\text{ICR}}^{\prime\,\text{arg}}$ is defined in the same way on N_{arg}.* Therefore, the approximation reduces the complexity from 2^n to 2^b.

Finally, the total training objective is given as follows.

$$\mathcal{L} = \mathcal{L}_{\text{SFT}} + \lambda_{\text{tool}}\mathcal{L}_{\text{ICR}}^{\prime\,\text{tool}} + \lambda_{\text{arg}}\mathcal{L}_{\text{ICR}}^{\prime\,\text{arg}}, \tag{9}$$

where $\mathcal{L}_{\text{SFT}}$ denotes the loss term for SFT, and λ_{tool} and λ_{arg} are positive weights.

4.2 EXPERIMENTS

Experimental Setup. We compare each recipe with and without ICR under paired initializations, data, and evaluation conditions. Controlled experiments use the controlled SFT set in Appendix C, which mixes single-turn calls, multi-turn trajectories, and no-call examples; method-specific experiments retain Hammer, ToolGen, and CoALM's respective representations and their own tool retrieval, whereas controlled experiments receive the ground-truth tools on StableToolBench. BFCL

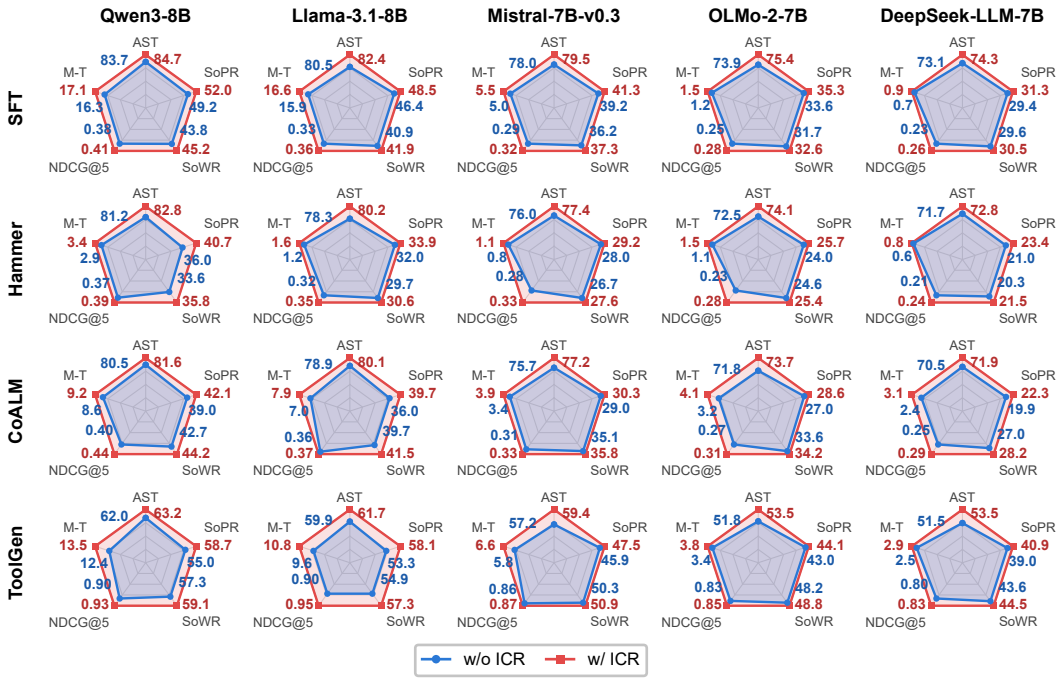


Figure 4: **SFT-based method comparisons without (blue) and with (red) ICR across five backbones.** The five axes represent five metrics, including the AST and the M-T for the BFCL dataset, the NDCG@5 for tool retrieval, and the SoPR and the SoWR for the StableToolBench dataset.

v4 (Patil et al., 2025) evaluates structured calling with the AST (the mean of the Non-Live and Live AST summary scores) and the M-T (the mean of the four multi-turn categories); StableToolBench (Guo et al., 2024) evaluates task completion with SoPR and SoWR; and τ^2-bench (Barres et al., 2026) evaluates interactive tasks with the per-domain pass[1]. NDCG@5 is the normalized discounted cumulative gain of the five tools that a method ranks highest among all StableToolBench tools, with the ground-truth tools of each query as the relevant ones; ToolGen ranks tools by generating its tool tokens, and Hammer, CoALM, and SFT rank them by the likelihood of calling each tool. Training schedules and the other evaluation settings are given in Appendix C.

Main Results. Figure 4 compares SFT, Hammer, CoALM, and ToolGen with and without ICR on five backbones. ICR improves all five metrics for every method on every backbone: across the 20 method–backbone pairs, the AST rises by 1.03 to 2.25 points and the SoPR by 1.11 to 4.80 points. Since the four methods differ in their training data or tool representations, these gains show that ICR is complementary to existing SFT-based methods. Table 1 reports results on the interactive tasks in τ^2-bench. ICR raises the pass[1] of SFT in all 15 backbone–domain pairs, by 0.16 to 0.95 points. Even on Telecom, which is out of domain for the controlled SFT set and on which SFT falls below Vanilla, ICR improves SFT on all five backbones. Table 2 in Appendix D further shows that ICR also improves the BFCL and StableToolBench results of Qwen3-4B and Qwen3-14B.

Interaction Changes after ICR. This experiment examines whether ICR indeed produces the interaction change that it is designed for. Figure 5 presents an independent held-out analysis of the normalized strength $\kappa(m)$ of SFT and SFT+ICR for each tool-calling stage. For both tool selection and argument generation, the LLM trained with SFT+ICR encodes stronger complex interactions and weaker simple interactions than the LLM trained with SFT alone, that is, $\kappa(m)$ is lower at low orders and higher at high orders. To verify that this difference is not an artifact of the normalization in Equation (4), we also compute the raw strength $\mathbb{E}_x\mathbb{E}_{S\subseteq N,\,|S|=m}[|I(S)|]$ before normalization; please see Figure 9 of Appendix D for the results.

Ablation Studies. We ablate the two effects of ICR and the two tool-calling stages on Qwen3-8B, and rule out alternative explanations with additional controls.

Table 1: τ^2-bench pass[1] of the controlled evaluation across backbones and scales, in percent. Vanilla is the checkpoint before SFT. The "Δ vs. SFT" rows give the improvement of SFT + ICR over SFT. Please see Figure 4 and Table 2 for additional BFCL and StableToolBench results on more models.

	Qwen3-4B	Qwen3-8B	Qwen3-14B	Llama-3.1-8B	Mistral-7B-v0.3
τ^2-bench Retail					
Vanilla	28.07	41.01	42.98	5.26	2.85
SFT	35.09	49.12	50.15	20.18	16.52
SFT + ICR (Ours)	35.75	50.07	50.66	20.76	17.18
Δ vs. SFT	**+0.66**	**+0.95**	**+0.51**	**+0.58**	**+0.66**
τ^2-bench Airline					
Vanilla	12.00	12.50	14.50	22.00	16.00
SFT	14.50	15.67	17.83	24.17	18.67
SFT + ICR (Ours)	14.67	15.83	18.17	24.67	19.00
Δ vs. SFT	**+0.17**	**+0.16**	**+0.34**	**+0.50**	**+0.33**
τ^2-bench Telecom					
Vanilla	17.54	19.08	29.82	20.61	13.16
SFT	15.79	17.32	26.97	15.57	10.67
SFT + ICR (Ours)	16.15	17.62	27.19	16.52	10.89
Δ vs. SFT	**+0.36**	**+0.30**	**+0.22**	**+0.95**	**+0.22**

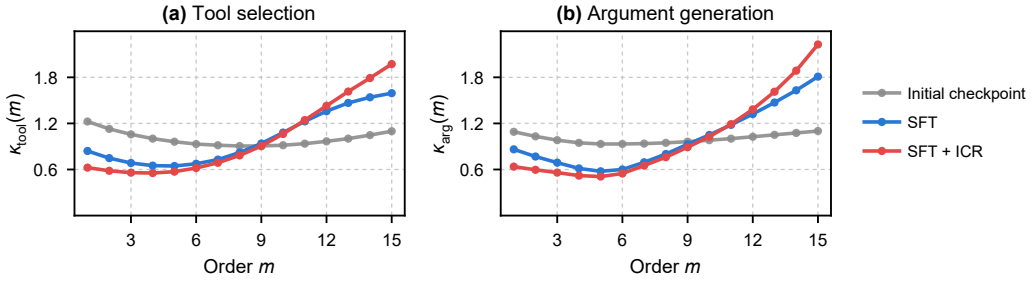


Figure 5: **Testing the intended interaction change after ICR.** Held-out interactions of Qwen3-8B for tool selection (a) and gold-tool-conditioned argument generation (b): normalized strength $\kappa(m)$ versus the order m for the initial checkpoint, SFT, and SFT+ICR. In both stages, SFT+ICR lies below SFT at low orders and above it at high orders, which is the change encouraged by Equation (9).

Exploring the individual effects of enhancing complex interactions and suppressing simple interactions. To examine whether each of the two effects contributes to the improvement on its own, we remove one of them at a time and reverse both of them. Writing the auxiliary term of Equation (9) as $\mathcal{L}_{\text{ICR}} = \lambda_{\text{tool}}\mathcal{L}_{\text{ICR}}^{\text{tool}} + \lambda_{\text{arg}}\mathcal{L}_{\text{ICR}}^{\text{arg}}$, we compare four losses: (1) $\mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{ICR}}$; (2) $\mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{low}}$ with $\mathcal{L}_{\text{low}} = \lambda_{\text{tool}}\mathcal{C}_{\text{tool}}^{\text{low}} + \lambda_{\text{arg}}\mathcal{C}_{\text{arg}}^{\text{low}}$, which removes high-order enhancement; (3) $\mathcal{L}_{\text{SFT}} + \mathcal{L}_{\text{high}}$ with $\mathcal{L}_{\text{high}} = -\lambda_{\text{tool}}\mathcal{C}_{\text{tool}}^{\text{high}} - \lambda_{\text{arg}}\mathcal{C}_{\text{arg}}^{\text{high}}$, which removes low-order suppression; and (4) $\mathcal{L}_{\text{SFT}} - \mathcal{L}_{\text{ICR}}$, which reverses both auxiliary signs. Figure 6(a) shows that, on Qwen3-8B, each of the two single-effect losses raises the AST of SFT from 83.66% (to 84.14% with $\mathcal{L}_{\text{low}}$ and to 84.28% with $\mathcal{L}_{\text{high}}$), but less than the full loss (84.74%), whereas reversing both terms lowers it to 82.97%. Therefore, both effects contribute to the improvement and complement each other.

Exploring the individual effects of applying ICR to tool selection and to argument generation. To examine whether each term mainly improves its own stage, we apply ICR to one stage at a time by setting $\lambda_{\text{arg}} = 0$ (tool-only ICR) or $\lambda_{\text{tool}} = 0$ (argument-only ICR) in Equation (9). Figure 6(b) reports two official BFCL v4 categories: a Simple entry has a single candidate function and thus tests only argument generation, whereas a Multiple entry has two to four candidates and thus also tests tool selection. Tool-only ICR raises Multiple by 0.60 points but Simple by only 0.08 points, so it mainly improves tool selection. Argument-only ICR raises Simple by 0.90 points and Multiple by only 0.52 points, no more than its gain on Simple, so it mainly improves argument generation rather than tool selection. The official BFCL error types of the failed Multiple entries confirm this (Table 5 in Appendix D): tool-only ICR mainly removes calls to a wrong function, whereas argument-only ICR mainly removes wrong arguments. Applying ICR to both stages yields the largest AST gain, so the two terms improve different stages and are complementary.

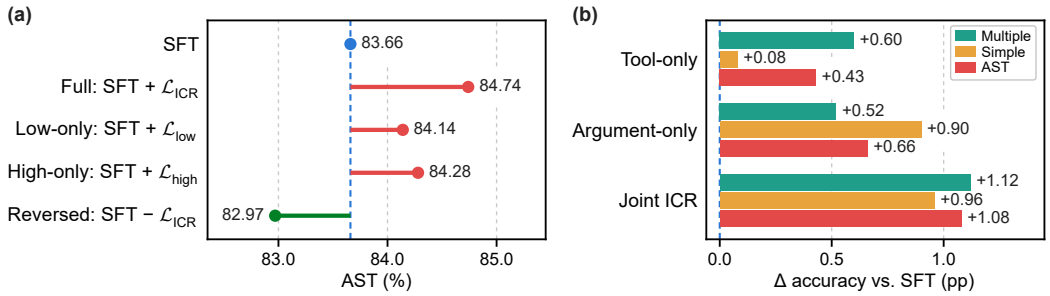


Figure 6: **Ablation studies of ICR on Qwen3-8B.** (a) AST of SFT and of SFT with each of the four losses in Section 4.2. (b) Change in the Multiple, Simple, and AST scores of BFCL v4 relative to SFT, when ICR is applied only to tool selection, only to argument generation, or to both.

Additional controls. Equal-compute SFT, SFT on masked inputs x_T, and generic regularization controls in Appendix D test whether the gains can be explained by additional computation or by exposure to masked inputs. All of them stay well below ICR, raising the AST of SFT by at most 0.38 points on Qwen3-8B against 1.08 points for ICR (Tables 4 and 6), which confirms that the gains come from ICR rather than from these factors.

5 Related Work

Tool Use of LLM Agents. Besides the SFT-based methods reviewed in Section 1, reinforcement learning methods such as ToolRL (Qian et al., 2025) and Tool-N1 (Zhang et al., 2026) improve tool calling with rule-based rewards. Most analyses of tool use examine the model outputs: ToolBeHonest (Zhang et al., 2024) and When2Call (Ross et al., 2025) diagnose when agents hallucinate tools or fail to abstain from calling, and Faghih et al. (2025) show that tool selection changes substantially when only the tool descriptions are edited. Other studies examine internal signals, relating tool-use errors to the troughs of the attention waveform (Chen et al., 2024) or to collapsed representations of tool reliability (Yin et al., 2026). However, these studies do not explain which combinations of input words drive tool use, or what different factors change in common when they improve it.

Using Game-Theoretic Interactions to Explain DNNs. Ren et al. (2023) proposed to explain DNNs with Harsanyi interactions (Harsanyi, 1963) between input variables, and showed that a few salient interactions suffice to reproduce the output of a DNN on exponentially many masked inputs. Li & Zhang (2023) provided evidence that such interactions represent transferable and discriminative concepts, and Liu et al. (2023) found that complex interactions are more difficult to learn than simple ones. The interaction framework has been used to explain the learning dynamics of DNNs (Ren et al., 2024), dropout (Zhang et al., 2021), and adversarial transferability (Wang et al., 2021), as well as the inference of LLMs in legal judgment (Chen et al., 2026), arithmetic reasoning (Wen et al., 2025), prompt sensitivity (Qin et al., 2026), and knowledge distillation (Wang et al., 2026). We extend it to explain how different factors improve the tool use of LLM agents.

6 Conclusion

We use Harsanyi interactions to explain how the SFT-based method, the model scale, and the amount of SFT data affect the tool use of LLM agents. In both tool selection and argument generation, LLMs with better tool-use performance usually encode stronger complex interactions and weaker simple interactions. Based on this common mechanism, we propose ICR, which enhances the effectiveness of SFT across different SFT-based methods and backbones on three tool-use benchmarks.

Limitation. Interaction-based analysis and regularization require additional computation beyond standard SFT, and the current analysis over a fixed number of input words may be less suitable for more complex tool-use settings. Future work may explore more efficient and adaptive interaction modeling for long-context, large-tool-set, and multi-turn scenarios.

AI USE STATEMENT

In this work, generative AI tools were used to support code debugging, language polishing, and LaTeX formatting. We did not use these tools to generate research ideas, draft the related work section, or engage in any plagiarism-related activities. We assume full responsibility for the final content of this work, including the text, claims, experimental results, and all artifacts created with the assistance of generative AI.

ETHICS STATEMENT

This work investigates tool selection and argument generation in LLM agents using existing datasets and benchmarks. The use and redistribution of third-party datasets, models, and tools are subject to their respective licenses and access conditions. Improved tool-use performance does not guarantee safe or authorized actions; deployment requires appropriate permission controls, argument validation, and human oversight for consequential actions.

REPRODUCIBILITY STATEMENT

Sections 2–4 define the interaction-based analysis, the stage-specific output scores, and the Interaction Complexity Regularization objective. Appendix A provides theoretical derivations and clarifies the relationship between word-level interactions and grouped computations. Appendix B describes the masking procedure and the construction of the mask embedding. Appendix C specifies the experimental protocol, including data sources and splits, model and method comparisons, training configurations, interaction estimation, and evaluation procedures. These descriptions identify the methodological choices needed to reproduce the analysis and training procedure.

REFERENCES

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Emmanouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. Can a single model master both multi-turn conversations and tool use? CoALM: A unified conversational agentic language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12370–12390, 2025. doi: 10.18653/v1/2025.acl-long.605. URL https://aclanthology.org/2025.acl-long.605/.

Marco Ancona, Cengiz Öztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for Shapley value approximation. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 272–281, 2019.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OC2z7iSQKa.

Lu Chen, Yuxuan Huang, Yixing Li, Dongrui Liu, Qihan Ren, Shuai Zhao, Kun Kuang, Zilong Zheng, and Quanshi Zhang. Can LLMs reason soundly in law? auditing inference patterns for legal judgment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5T0BXtJxzN.

Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11160–11174, 2024. doi: 10.18653/v1/2024.acl-long.601.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Kazem Faghih, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. Tool preferences in agentic LLMs are unreliable. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20954–20969, 2025. doi: 10.18653/v1/2025.emnlp-main.1060.

Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2950–2958, 2019.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11143–11156, 2024. doi: 10.18653/v1/2024.findings-acl.664. URL https://aclanthology.org/2024.findings-acl.664/.

John C. Harsanyi. A simplified bargaining model for the n-person cooperative game. *International Economic Review*, 4(2):194–220, 1963.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Mingjie Li and Quanshi Zhang. Does a neural network really encode symbolic concepts? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20452–20469. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23at.html.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. Robust function-calling for on-device language models via function masking. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yVQcr4qjD6.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Dongrui Liu, Huiqi Deng, Xu Cheng, Qihan Ren, Kangrui Wang, and Quanshi Zhang. Towards the difficulty for a deep neural network to learn concepts of different complexities. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.00920.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pp. 54463–54482, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/61cce86d180b1184949e58939c4f983d-Abstract-Datasets_and_Benchmarks_Track.html.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 38, 2025. doi: 10.52202/085713-2184. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5e3661f7fe4c8ac5652d62eb3d3c96ea-Abstract-Datasets_and_Benchmarks_Track.html.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, 2025.

Ruiyang Qin, Qingzhuo Wang, Tian Wang, Zhihua Wei, and Wen Shen. Evaluating and explaining prompt sensitivity of LLMs using interactions. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.16789.

Jie Ren, Mingjie Li, Qirui Chen, Huiqi Deng, and Quanshi Zhang. Defining and quantifying the emergence of sparse concepts in DNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20280–20289, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Ren_Defining_and_Quantifying_the_Emergence_of_Sparse_Concepts_in_DNNs_CVPR_2023_paper.html.

Qihan Ren, Junpeng Zhang, Yang Xu, Yue Xin, Dongrui Liu, and Quanshi Zhang. Towards the dynamics of a DNN learning symbolic interactions. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50653–50688, 2024.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to call tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3391–3409, 2025. doi: 10.18653/v1/2025.naacl-long.174.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328, 2017.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, et al. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Qingzhuo Wang, Ruiyang Qin, Zhenxin Qin, Wen Shen, and Zhihua Wei. A unified approach to interpreting knowledge distillation for large language models via interactions. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65719.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. ToolGen: Unified tool retrieval and calling via generation. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XLMAMmowdY.

Xin Wang, Jie Ren, Shuyun Lin, Xiangming Zhu, Yisen Wang, and Quanshi Zhang. A unified approach to interpreting and boosting adversarial transferability. In *International Conference on Learning Representations*, 2021.

Leilei Wen, Liwei Zheng, Hongda Li, Lijun Sun, Zhihua Wei, and Wen Shen. Interpreting arithmetic reasoning in large language models using game-theoretic interactions. In *Advances in Neural Information Processing Systems*, volume 38, pp. 77401–77425. Curran Associates, Inc., 2025. doi: 10.52202/085713-2596. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6fb3465e95fbfbde193d3df17769f59c-Paper-Conference.pdf.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. TOUCAN: Synthesizing 1.5M tool-agentic data from real-world MCP environments, 2026. URL https://openreview.net/forum?id=UgFmrYcLOt.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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*, 2023. URL https://arxiv.org/abs/2210.03629.

Chenlong Yin, Zeyang Sha, Shiwen Cui, Changhua Meng, and Zechao Li. The reasoning trap: How enhancing LLM reasoning amplifies tool hallucination. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8310–8328, 2026. doi: 10.18653/v1/2026.acl-long.376.

Hao Zhang, Sen Li, YinChao Ma, Mingjie Li, Yichen Xie, and Quanshi Zhang. Interpreting and boosting dropout from a game-theoretic view. In *International Conference on Learning Representations*, 2021.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-Research-Tool-N1: Exploring tool-using language models with reinforced reasoning. In *International Conference on Learning Representations*, 2026.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11388–11422, 2024. doi: 10.18653/v1/2024.emnlp-main.637.

Xinhao Zheng, Huiqi Deng, and Quanshi Zhang. Towards attributions of input variables in a coalition. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78115–78138. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zheng25d.html.

A What Grouping Preserves: Exact and Conditional Guarantees

We distinguish four questions: whether grouping preserves the score of a fixed model, whether it preserves a decision made from that score, whether it preserves the ranking of interaction statistics across models, and whether training with an approximate loss produces the same model as training with the exact loss. These are different statements. Below we prove exact reconstruction and invariance of direct evaluation for a fixed checkpoint, followed by sufficient conditions for decision and ranking preservation. We also give a counterexample to an unconditional ranking guarantee.

A.1 Exact Word-Level Reconstruction

Fix an input x, the set N of words to be analyzed, the unmasked background context, the target y^*, and the [MASK] embedding. Recall from Equation (2) that $I(\emptyset) = v(x_\emptyset)$. By the Harsanyi definition in Equation (2), for every $T \subseteq N$,

$$\sum_{S \subseteq T} I(S) = \sum_{S \subseteq T} \sum_{U \subseteq S} (-1)^{|S|-|U|} v(x_U) \tag{10}$$

$$= \sum_{U \subseteq T} v(x_U) \sum_{U \subseteq S \subseteq T} (-1)^{|S|-|U|} \tag{11}$$

$$= \sum_{U \subseteq T} v(x_U)(1-1)^{|T|-|U|} = v(x_T). \tag{12}$$

The inner sum is one when $U = T$ and zero otherwise. This proves universal matching. In particular, it applies separately to v_{tool} and v_{arg}. These are exact algebraic identities for the chosen scalar scores, not assumptions about the model architecture.

A.2 Exact Reconstruction on Grouped Inputs

Let $\mathcal{B} = (B_1, \ldots, B_b)$ be a partition of N into disjoint nonempty word sets, and let $U(K) = \bigcup_{j \in K} B_j$. Define

$$J_{\text{tool}}(K) = \sum_{A \subseteq K} (-1)^{|K|-|A|} v_{\text{tool}}(x_{U(A)}), \qquad J_{\text{tool}}(\emptyset) = v_{\text{tool}}(x_\emptyset). \tag{13}$$

The corresponding $J_{\text{arg}}(K)$ is defined from v_{arg}. We use J to distinguish the computational proxy from a single word-level coefficient I.

Theorem 2 (Grouped reconstruction)**.** *For every retained block set $A \subseteq [b]$, $\sum_{K \subseteq A} J_{\text{tool}}(K) = v_{\text{tool}}(x_{U(A)})$. The same identity holds for argument generation. In particular, taking $A = [b]$ reconstructs the full unmasked score exactly.*

Proof. Substitute Equation (13) and exchange the finite sums. The coefficient of $v_{\text{tool}}(x_{U(C)})$ is $\sum_{C \subseteq K \subseteq A} (-1)^{|K|-|C|} = (1-1)^{|A|-|C|}$, which vanishes unless $C = A$. This leaves exactly $v_{\text{tool}}(x_{U(A)})$. The argument uses every grouped coefficient, including the empty-set term $I(\emptyset) = v(x_\emptyset)$. Retaining only coefficients appearing in the training penalty is insufficient for this reconstruction identity. □

Relation to Word Interactions. Let $h(S) = \{j : S \cap B_j \neq \emptyset\}$. Substituting word-level reconstruction into the grouped definition yields, for nonempty K,

$$J_{\text{tool}}(K) = \sum_{S \subseteq U(K)} I_{\text{tool}}(S) \sum_{h(S) \subseteq A \subseteq K} (-1)^{|K|-|A|} \tag{14}$$

$$= \sum_{\substack{S \subseteq U(K) \\ h(S) = K}} I_{\text{tool}}(S). \tag{15}$$

Only sets touching every participating block survive. Consequently, the grouped coefficient contains word interactions of orders from $|K|$ to $|U(K)|$, with possible cancellation between signs.

Word-Count Convention. For selecting the bands of the training proxy, we assign a grouped coalition its covered-word count $m_{\mathcal{B}}(K) = |U(K)|$. Thus, with 24 words in four disjoint six-word blocks, $K = \{1, 3\}$ is assigned $m_{\mathcal{B}}(K) = 12$, not 2. This respects a word-count axis. It does not imply $J_{\text{tool}}(\{1, 3\}) = I_{\text{tool}}(B_1 \cup B_3)$: the left side includes orders 2 through 12. Only the exact word interaction on $B_1 \cup B_3$ has exclusively order 12. This distinction is essential to the guarantees below.

A.3 When Accuracy Is Unchanged

Theorem 3 (Invariance of direct evaluation)**.** *Fix a trained checkpoint, prompts, decoder, tool environment, and evaluator. If grouping is used only for post-hoc analysis and is not supplied to the decoder or evaluator, replacing exact interaction analysis by grouped analysis leaves generated calls and their evaluation scores unchanged. With stochastic decoding or environments, the evaluation distribution is unchanged; samplewise equality additionally requires matched randomness.*

Proof. The inputs to generation, the parameters, and the evaluation functions are identical. For deterministic decoding, the next token is therefore identical; induction over generated tokens gives the same call. Applying the same tool and environment transition to the same call produces the same next state, and induction gives the same trajectory and metric. For stochastic components the conditional distributions are identical at every step. Coupling their random draws gives the samplewise statement. Interaction coefficients are never used in this computation, so their approximation cannot change this fixed-checkpoint evaluation. □

This result applies to the direct evaluation used in this paper. It does not claim that ICR-trained and ordinary SFT-trained checkpoints have equal accuracy. Training deliberately changes the checkpoint. Also, an exact gold-target score reconstruction alone is not a theorem about generated-call accuracy: decisions require scores of competing candidates or all logits used by the decoder.

Theorem 4 (Decision preservation under bounded score error)**.** *For a finite candidate set, let $f_x(c)$ be the decision score and $\widetilde{f}_x(c)$ its approximation. Suppose $\max_c |\widetilde{f}_x(c) - f_x(c)| \le \epsilon_x$. If the exact winner c_x^* is unique and its margin $\gamma_x = f_x(c_x^*) - \max_{c \neq c_x^*} f_x(c)$ exceeds $2\epsilon_x$, the approximate winner is also c_x^*.*

Proof. For every competitor, $\widetilde{f}_x(c_x^*) - \widetilde{f}_x(c) \ge f_x(c_x^*) - f_x(c) - 2\epsilon_x \ge \gamma_x - 2\epsilon_x > 0$. □

For D examples with binary correctness, this implies

$$|\widetilde{\text{Acc}} - \text{Acc}| \le \frac{1}{D} \sum_{x=1}^{D} \mathbf{1}\{\gamma_x \le 2\epsilon_x\}. \tag{16}$$

For greedy autoregressive generation, the margin condition must hold at every visited prefix for all next-token logits; induction then preserves the complete call. Stochastic sampling and an arbitrary sequence-scoring rule cannot be substituted for this condition without an additional argument.

Where a Score-Error Bound Can Come From. If every computed grouped coalition score differs from its exact value by at most η, then $|\widetilde{J}_{\text{tool}}(K) - J_{\text{tool}}(K)| \le 2^{|K|}\eta$ by the triangle inequality. If a reconstruction on A retains only $\mathcal{R} \subseteq 2^A$, its error is at most

$$\sum_{K \in \mathcal{R}} |\widetilde{J}_{\text{tool}}(K) - J_{\text{tool}}(K)| + \sum_{K \subseteq A,\, K \notin \mathcal{R}} |J_{\text{tool}}(K)|. \tag{17}$$

This is a numerical and truncation bound. It is not a bound on recovering the word-order strength distribution from grouped scores.

A.4 When Interaction-Ratio Rankings Are Preserved

For a fixed stage and model, denote the exact raw mean strengths in the high and low bands by H and $L > 0$. Since the common normalizer cancels, the ratio in Equation (5) is $R = H/L$. Let $\widetilde{H}$ and $\widetilde{L}$ be proxies with certified errors $|\widetilde{H} - H| \le \delta_H$ and $|\widetilde{L} - L| \le \delta_L < L$. These are assumptions to verify, not consequences of grouping.

Theorem 5 (Ratio and ordering stability)**.** *The ratio error satisfies*

$$|\widetilde{R} - R| \le E := \frac{L\delta_H + H\delta_L}{L(L - \delta_L)}. \tag{18}$$

For two models M and Q, $R_M - R_Q > E_M + E_Q$ guarantees $\widetilde{R}_M > \widetilde{R}_Q$.

Proof. Write $\widetilde{H} = H + e_H$ and $\widetilde{L} = L + e_L$. Then $\widetilde{R} - R = (Le_H - He_L)/(L(L + e_L))$. Use $|e_H| \le \delta_H$, $|e_L| \le \delta_L$, and $L + e_L \ge L - \delta_L > 0$ to obtain Equation (18). The pairwise statement follows from $\widetilde{R}_M - \widetilde{R}_Q \ge R_M - R_Q - E_M - E_Q > 0$. □

With no ties, if every pairwise order is preserved, the Spearman correlation with a fixed accuracy vector is unchanged. Pearson correlation need not be unchanged, since it depends on numerical distances. For accuracy rankings, Equation (16) gives analogous model-specific errors; a pairwise accuracy gap exceeding their sum preserves its ordering.

A Certifiable Interval. When only proxy values and error bounds are available, and $\widetilde{L} > \delta_L$, an interval containing the exact ratio is

$$\left[\frac{\max(0, \widetilde{H} - \delta_H)}{\widetilde{L} + \delta_L}, \frac{\widetilde{H} + \delta_H}{\widetilde{L} - \delta_L}\right]. \tag{19}$$

Disjoint intervals certify the pairwise ordering. A small audit set supplies evidence only on that set unless an additional generalization or concentration assumption is justified.

Counterexample: The Same Grouping Can Reverse Rankings. Let $N = \{1, 2, 3\}$, $B_1 = \{1, 2\}$, and $B_2 = \{3\}$. Define a valid score game by

$$v^{a,h}(x_T) = \mathbf{1}[3 \in T] + a\mathbf{1}[\{1, 3\} \subseteq T] + h\mathbf{1}[N \subseteq T], \qquad a, h \ge 0. \tag{20}$$

The only nonzero word interactions are $I(\{3\}) = 1$, $I(\{1, 3\}) = a$, and $I(N) = h$. The order ranges in Equation (5) give low order 1 and high order 3, so $L = 1/3$, $H = h$, and $R = 3h$. Grouping yields $J(\{1\}) = 0$, $J(\{2\}) = 1$, and $J(\{1, 2\}) = a + h$. Classifying these proxies by covered-word count gives a low-band mean of 1 and a high-band mean of $a + h$, hence $\widetilde{R} = a + h$. Model M with $(a, h) = (0, 2)$ has $R_M = 6$ and $\widetilde{R}_M = 2$; model Q with $(a, h) = (4, 1)$ has $R_Q = 3$ and $\widetilde{R}_Q = 5$. The ranking reverses despite identical grouping, nonnegative effects, and exact reconstruction of both full scores. Therefore a shared grouping protocol alone cannot prove accuracy of word-order statistics or fairness of their rankings.

A.5 Training Approximation Requires Additional Assumptions

An equality for a fixed checkpoint does not imply equality between parameters obtained by optimizing two different objectives. A sufficient stability result can be derived for gradient descent. Let F be the exact objective and $\widetilde{F}$ its proxy, with iterates $\theta_{t+1} = \theta_t - \alpha_t \nabla F(\theta_t)$ and $\widetilde{\theta}_{t+1} = \widetilde{\theta}_t - \alpha_t \nabla \widetilde{F}(\widetilde{\theta}_t)$ from the same initialization. Suppose ∇F is L_F-Lipschitz on the relevant region and $\|\nabla \widetilde{F}(z) - \nabla F(z)\| \le d_t$ at the proxy iterate. Then

$$\|\widetilde{\theta}_T - \theta_T\| \le B_T := \sum_{t=0}^{T-1} \alpha_t d_t \prod_{u=t+1}^{T-1} (1 + \alpha_u L_F). \tag{21}$$

Indeed, subtracting the two updates and applying the triangle inequality gives $e_{t+1} \le (1+\alpha_t L_F)e_t + \alpha_t d_t$; induction with $e_0 = 0$ proves the bound. If every decision score is K_f-Lipschitz in parameters, its change is bounded by $K_f B_T$, and margins exceeding $2K_f B_T$ preserve the corresponding decisions by the earlier theorem.

B Masking Input Words with a Learned [MASK] Embedding

Removal-based attribution and interaction analyses quantify the contribution of an input variable by the change in the model output when this variable is removed from the input (Lundberg & Lee, 2017;

Ancona et al., 2019; Fong et al., 2019). Every way of removing a variable has side effects, so the choice matters. Filling the removed variable with a constant, such as zero or its mean over the data, which appears as a black or grey region in an image (*e.g.*, the black-image baseline of Sundararajan et al. (2017) and Ancona et al. (2019)), creates inputs that lie outside the data distribution (Sturmfels et al., 2020) and may give the model new evidence to react to (Dabkowski & Gal, 2017). Smoothing the removed region with a Gaussian filter (Fong & Vedaldi, 2017; Fong et al., 2019) erases fine details but keeps the coarse, low-frequency content, so the variable is only partly removed (Covert et al., 2021; Sturmfels et al., 2020).

For text, we follow Wang et al. (2026) and overwrite each removed word with a special [MASK] token in the embedding space. For instance, removing "*flights*" from the request "*Find flights from Shanghai to Beijing*" turns it into "*Find [MASK] from Shanghai to Beijing*." The word thus loses its meaning, while the length of the input and the positions of all other tokens stay the same. Only the words in N are masked; the rest of the prompt, including the system prompt, the chat template, and the tool descriptions outside N, is identical in every masked input.

The [MASK] Embedding for Each LLM. Because the vocabularies of most decoder-only LLMs do not contain a native [MASK] token, we follow Ren et al. (2023) and learn a dedicated [MASK] embedding $\mathbf{b} \in \mathbb{R}^d$ for each LLM, where d is the dimension of the input embeddings. Concretely, the input embedding of every token of a masked word is replaced with $\mathbf{b}$ before the first Transformer layer, while the position and the attention mask of the token are kept unchanged. The vector $\mathbf{b}$ is initialized with the input embedding of the padding token of the tokenizer and optimized on a set of 720 question–answer samples. For each sample, we randomly draw three disjoint word subsets $S_1, S_2, S_3 \subseteq N$ and minimize

$$\mathcal{L}(\mathbf{b}) = \mathbb{E}_{S_1,S_2,S_3}\Big[\big|v(x_{S_1\cup S_2\cup S_3}) - v(x_{S_1\cup S_3}) - v(x_{S_2\cup S_3}) + v(x_{S_3})\big|\Big], \tag{22}$$

which measures the interaction between the word groups S_1 and S_2 in the context of S_3. A [MASK] embedding that minimizes the loss in Equation (22) therefore removes the masked words while introducing as little spurious interaction as possible. The embedding is optimized with SGD for 100 epochs, then frozen and shared across all experiments using the corresponding LLM, so that the masked inputs x_T in Equation (2) differ from x only in the words in $N \setminus T$.

C Implementation and Experimental Protocol

Data and Splits. For the Standard SFT baseline in Section 3, we fine-tune the model on 20,000 single-turn examples drawn from xLAM function-calling 60k (Liu et al., 2024), all of which require a call. The data-amount comparison of Section 3 and the ICR experiments in Section 4 use the controlled SFT set instead. The controlled SFT set contains 20,000 training examples of three kinds: 16,000 single-turn examples from xLAM function-calling 60k; 2,000 multi-turn agent–user trajectories from APIGen-MT-5k (Prabhakar et al., 2025), in which the agent asks the user for missing information, reads tool observations, and follows a domain policy; and 2,000 no-call examples (10%), built as in Hammer (Lin et al., 2025) by removing the correct function from the candidate list of an xLAM example and replacing the target with an empty call list. After auditing overlap with the evaluation data at the task, API, and near-duplicate level, we randomly sample the three subsets. The 500-example validation set follows the same proportions. APIGen-MT-5k is generated in the τ-bench retail and airline environments, so the Retail and Airline results of τ^2-bench are in-domain for the multi-turn part, whereas Telecom is not. No-call examples have no gold tool and receive only the SFT loss; in a multi-turn trajectory, every assistant turn that issues a call is an eligible example for both stages, with the preceding dialogue as its context. Separate discovery and post-training confirmation pools each contain 200 eligible examples per prediction stage. The training set, validation set, discovery pool, and confirmation pool are disjoint at the source-example level. All BFCL results use the complete evaluation categories of the stated benchmark version.

Model and Method Comparisons. The scale comparison of Section 3 uses the Qwen3 models from 0.6B to 32B, and Table 1 uses Qwen3-4B/8B/14B, all from one release series with thinking disabled. The data-amount comparison of Section 3 fine-tunes Qwen3-8B with the controlled recipe on nested subsets of 10%, 20%, 30%, 50%, 70%, and 100% of the controlled SFT set, each with the same proportions of the three kinds of examples (2,000 to 20,000 examples) under a fixed optimizer-step budget, so that smaller subsets are revisited; the 100% checkpoint is the SFT model of

Table 1. Qwen3-8B, Llama-3.1-8B, Mistral-7B-v0.3, OLMo-2-7B, and DeepSeek-LLM-7B serve as backbones for the method comparison and the ICR experiments. We call the released instruction-tuned model the *initial checkpoint*. The scale comparisons for Llama 2, Ministral 3, OLMo 2, and DeepSeek-R1-Distill use released checkpoints without further training.

Standard SFT, Hammer, ToolGen, and CoALM are compared on Qwen3-8B, with the same comparisons extended to the four other backbones in Figure 7. All four methods are analyzed with shared evaluation prompts; Standard SFT is trained on the single-turn xLAM examples above, and Hammer, ToolGen, and CoALM on their native training data. A comparison controlling the source data (Table 3) is distinguished from reproduction using each method's original data. Before/after comparisons use the same evaluation interface for the corresponding method.

Training Configuration. The controlled recipe uses BF16 LoRA with rank 16, scaling 32, dropout 0, and the attention and MLP projections $q, k, v, o, \mathrm{gate}, \mathrm{up}, \mathrm{down}$ as targets, with architecture-specific names recorded. We use AdamW with a learning rate of 10^{-4}, zero weight decay, and a cosine decay schedule with 3% warmup. We train for one epoch with a global batch size of 48 and gradient clipping at 1.0. The maximum sequence length is 3,072 tokens for single-turn and no-call examples and 16,384 tokens for multi-turn trajectories. Micro-batch changes are offset by accumulation to preserve the example batch size; token counts are reported separately because tokenizers differ across families. Within each SFT/ICR pair, examples, order, initialization, optimizer, and supervised budget are matched.

Interaction Scores and Input Selection. Tool-selection scores use the target tool-identifier positions in Equation (1); argument scores use target argument positions under the gold tool and the same teacher-forced prefixes. Input words are chosen by fixed semantic rules independently of model attention and gradients. Tool selection considers intent, objects, constraints, tool descriptions, and schema cues. Argument generation considers parameter keys, input value spans, types, and cross-field constraints. Supervised labels may identify relevant input spans, but the target text and future observations are never inserted into the prompt. Words are aligned to subtokens, and all subtokens of a selected word are masked together with the [MASK] embedding described in Appendix B. During the masking process, masking is applied exclusively to the words in N, while all other "background" tokens of the prompt are kept unchanged and treated as fixed, unmasked background context within each comparison.

Efficient Training Implementation. At most 24 words are selected for each stage. To reduce the 2^n evaluations needed for exhaustive word-level decomposition, we group tool-selection words into at most six semantic roles and argument-generation words into at most four slot-aligned groups. The grouped coefficient is J_{tool} or J_{arg}, as defined in Appendix A.2. Band membership uses the covered-word count $m_{\mathcal{B}}(K) = |U(K)|$, not the number of groups. If either band is empty, the example retains SFT supervision and contributes no auxiliary term. This is a proxy, not an unbiased estimator of the full word-range loss. All grouped scores are evaluated for each selected example, requiring at most 64 tool-selection and 16 argument-generation scores.

Auxiliary updates occur every fourth optimizer step using one uniformly sampled eligible example per active stage from the global batch. Uniform example sampling estimates the empirical proxy objective; it does not remove the word-to-group discrepancy. Coefficients are calibrated on validation data to a target auxiliary-gradient norm of 5% of the SFT-gradient norm, accounting for update frequency, then selected using validation multipliers $\{0.25, 0.5, 1, 2, 4\}$. Paired ablations reuse the selected coefficients and samples. Alternating sums use FP32 accumulation with a higher-precision spot check.

Evaluation and Uncertainty. We report BFCL AST and multi-turn scores separately. StableToolBench uses a fixed tool environment, and τ^2-bench reports each domain separately. Figure 4 and Table 2 use the 330 solvable I1/I2/I3-Instruction tasks of StableToolBench and Table 8 its I1-Tool, I1-Category, and I2-Category groups, all with the gpt-4-turbo-2024-04-09 judge; the controlled comparisons provide the ground-truth tools of each query, whereas the method-specific comparisons use each method's own retrieval, and SoWR is the solvable win rate against the GPT-3.5-Turbo-0613 chain-of-thought reference. τ^2-bench (v1.0.1, base task split) reports per-domain pass[1] over four trials per task, with GPT-4.1 as the user simulator. Fixed decoder settings, model budgets, evaluator versions, environment snapshots, and retry policies are shared within comparisons. Direct generation and scoring use no grouped interaction proxy.

D Additional Results and Controls

Interaction Curves for the Remaining Backbones and Model Series. Figure 7 shows the SFT-method comparison of Figure 2(a, d) on the other four backbones, and Figure 8 shows the scale comparison of Figure 2(b, e) for the other four model series; the data-amount comparison of Figure 2(c, f) is conducted on Qwen3-8B only.

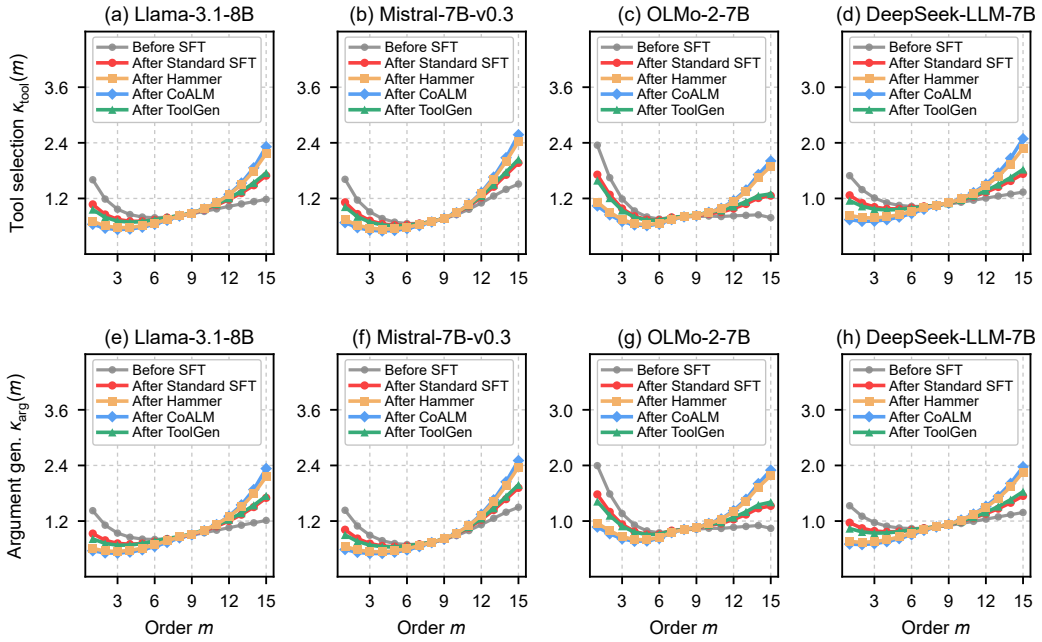


Figure 7: **SFT-based methods on the other four backbones.** Same layout as Figure 2(a, d): grey curves show the initial checkpoint and colored curves the model after SFT with each method; the top row is tool selection and the bottom row is argument generation.

Raw Interaction Strength. Figure 9 shows the raw strength $\mathbb{E}_x\mathbb{E}_{S\subseteq N,\,|S|=m}[|I(S)|]$ of the checkpoints in Figure 5, *i.e.*, the numerator of Equation (4). Because $\kappa(m)$ divides each curve by its own order-averaged strength Z, a curve that merely becomes steeper appears to lose low-order and gain high-order strength after normalization, even if its absolute strength rises or falls at every order. Only when the SFT+ICR curve lies below the SFT curve at low orders and above it at high orders in raw strength do the two directions of Equation (9) occur simultaneously. This holds for both tool selection and argument generation, so the change in Figure 5 is not an artifact of normalization.

Other Qwen3 Scales. Table 2 reports the BFCL and StableToolBench results of Qwen3-4B and Qwen3-14B in the controlled setting of Table 1.

Table 2: BFCL v4 AST and M-T and StableToolBench SoPR of Qwen3-4B and Qwen3-14B in the controlled setting of Table 1, in percent. Qwen3-8B is shown in the SFT row of Figure 4.

Backbone	Training	AST	M-T	SoPR
Qwen3-4B	Vanilla	80.84	7.63	42.07
	SFT	80.91	10.46	44.02
	SFT + ICR (Ours)	82.20	10.96	46.33
Qwen3-14B	Vanilla	83.38	18.13	54.29
	SFT	84.37	20.25	55.61
	SFT + ICR (Ours)	85.20	20.88	57.51

Controlled Masking and Cross-Backbone Replication. Table 3 shows a common-checkpoint recipe comparison, and Table 4 reports cross-backbone ablations.

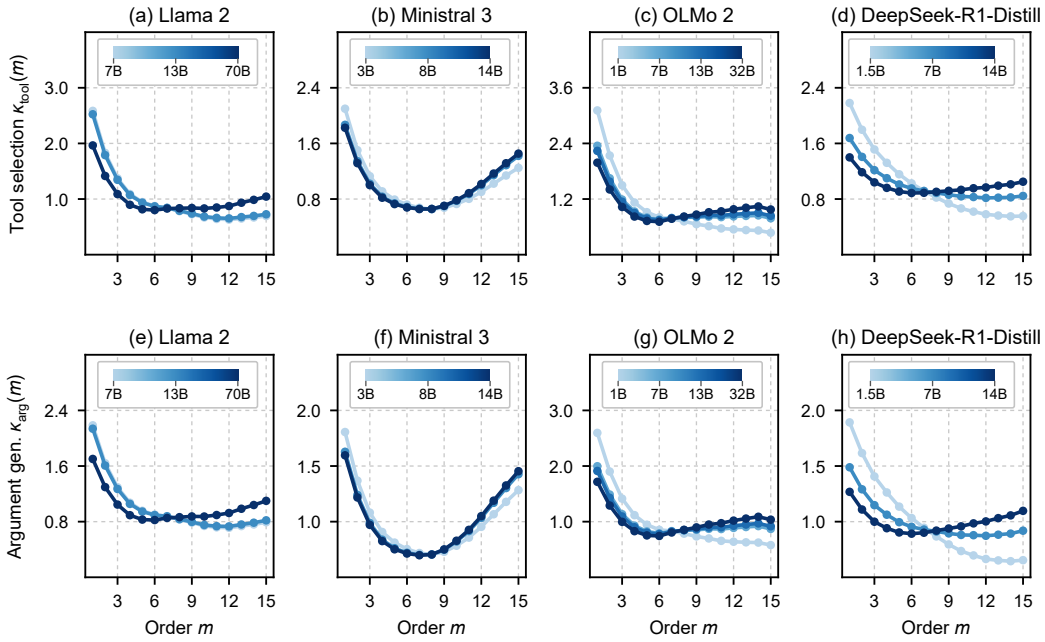


Figure 8: **Model scale in the other four series.** Same layout as Figure 2(b, e); darker curves denote larger models.

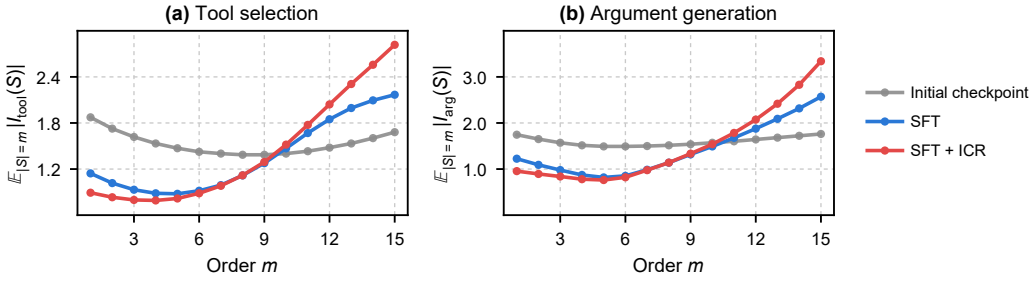


Figure 9: **Raw strength of the held-out interactions.** Raw mean absolute strength $\mathbb{E}_{|S|=m}|I(S)|$ versus the order m for tool selection (a) and gold-tool-conditioned argument generation (b) of Qwen3-8B, for the initial checkpoint (grey), SFT (blue), and SFT+ICR (red), on the same examples and words as Figure 5. In both stages, SFT+ICR lies below SFT at low orders and above it at high orders.

Table 3: Common-backbone method comparison on fixed ten-word tool-selection inputs. Raw low averages orders 1–4; raw high averages orders 8–10. Function masking and multitask training are controlled adaptations, not the native results in Figure 4. Multiple is the BFCL v4 Multiple score, i.e., the mean of the Non-Live and Live Multiple AST scores, and AST is the BFCL v4 AST as in Figure 4, both in percent.

Controlled recipe	Multiple	AST	Raw low	Raw high
SFT	86.02	83.66	1.02	1.87
Function-masking SFT	85.85	83.47	1.05	1.84
Multitask SFT	85.40	82.03	1.07	1.80
SFT + ICR	87.14	84.74	0.86	2.26

Table 4: Replication of the decisive ablations on Qwen3-8B and Llama-3.1-8B. Values are BFCL v4 AST scores in percent, as in Figure 4.

Condition	Qwen3-8B	Llama-3.1-8B
SFT	83.66	80.52
Tool ICR only	84.09	81.12
Argument ICR only	84.32	81.80
Suppress low only	84.14	81.26
Enhance high-order only	84.28	81.55
Reversed ICR	82.97	79.27
Equal GPU-hour SFT	83.90	80.91
KL-to-initial	83.89	80.86
SFT on masked inputs x_T	84.04	81.16
Full ICR	84.74	82.36

Error Types of the Stage Ablations. Table 5 splits the failed Multiple entries of Figure 6(b) by the error type that the official BFCL AST checker assigns. Tool-only ICR mainly removes calls to a wrong function (from 4.48% to 3.93%) and hardly changes wrong arguments (from 8.33% to 8.28%), whereas argument-only ICR mainly removes wrong arguments (from 8.33% to 7.84%) and hardly changes wrong functions (from 4.48% to 4.45%). Therefore, each term mainly improves the stage to which it is applied, and joint ICR reduces both error types.

Table 5: Error types of the failed BFCL v4 Multiple entries in the stage ablations of Figure 6(b) on Qwen3-8B, in percent. The official BFCL AST checker labels each failed entry: *wrong function* is a call to a function other than the correct one (`wrong_func_name`); *wrong arguments* are missing or unexpected parameters, incorrect parameter types, or incorrect parameter values of the correct function; *other* covers a wrong number of calls and outputs that cannot be decoded. As for the Multiple score, each rate is the mean over the Non-Live and Live Multiple categories, so the three rates add up to the failure rate, *i.e.*, 100 minus the Multiple score. Parentheses give the change relative to SFT.

Condition	Failed	Wrong function	Wrong arguments	Other
SFT	13.98	4.48	8.33	1.17
Tool-only ICR	13.38 (−0.60)	3.93 (−0.55)	8.28 (−0.05)	1.17 (0.00)
Argument-only ICR	13.46 (−0.52)	4.45 (−0.03)	7.84 (−0.49)	1.17 (0.00)
Joint ICR	12.86 (−1.12)	3.91 (−0.57)	7.78 (−0.55)	1.17 (0.00)

Alternative Explanations. Table 6 compares equal-compute SFT, SFT on masked inputs x_T, and generic regularization. Table 7 lists the accuracy of the checkpoints of Figure 2(c, f).

Table 6: Alternative explanations. Calling/task values are percentages; GPU-hours are relative to ordinary SFT on identical hardware. AST is the BFCL v4 AST score as in Figure 4, and Task is the mean pass^1 over the three τ^2-bench domains of Table 1, both on Qwen3-8B.

Training control	AST	Task	GPU-h ratio
SFT	83.66	27.37	1.0
SFT, equal GPU-hours	83.90	27.47	1.16
SFT + label smoothing	83.81	27.43	1.0
SFT + KL-to-initial	83.89	27.48	1.16
SFT on masked inputs x_T	84.04	27.52	1.16
SFT + mask consistency	84.01	27.51	1.16
SFT + ICR	84.74	27.84	1.16

Generalization and Independent Analysis. Table 8 reports results on official benchmark splits that differ from the training data, and Table 9 reports the word-level confirmation.

Table 7: Accuracy of the SFT checkpoints of Figure 2(c, f) on Qwen3-8B, in percent. The checkpoints are fine-tuned on nested subsets of the controlled SFT set under a fixed optimizer-step budget, so smaller subsets are revisited; the 100% checkpoint is the SFT model of Table 1.

Training variant	BFCL v3 overall	BFCL v4 AST
10 percent data	56.30	81.90
20 percent data	57.66	82.46
30 percent data	58.54	82.81
50 percent data	59.50	83.27
70 percent data	60.69	83.46
100 percent data	60.60	83.66

Table 8: Results of Qwen3-8B on official benchmark splits that differ from the training data, in percent. Live contains the user-contributed queries of BFCL v4; Miss Func, Miss Param, and Long Context are the BFCL v4 multi-turn categories with a missing function, a missing parameter, and a long context; I1-Tool, I1-Category, and I2-Category are the StableToolBench groups with unseen tools, unseen tool categories, and unseen categories of multi-tool instructions. Splits and denominators are separate, so rows must not be averaged.

Split	SFT	SFT + ICR	Metric
BFCL v4 Live	78.11	79.30	AST
BFCL v4 multi-turn Miss Func	12.33	12.83	Accuracy
BFCL v4 multi-turn Miss Param	16.50	17.17	Accuracy
BFCL v4 multi-turn Long Context	15.33	16.33	Accuracy
StableToolBench I1-Tool	47.26	50.11	SoPR
StableToolBench I1-Category	46.19	49.02	SoPR
StableToolBench I2-Category	41.53	44.89	SoPR

Table 9: Independent word-level confirmation on the fixed ten-word tool-selection set, after block-based training. Low and high denote unnormalized mean interaction strengths over the low- and high-order bands, respectively. Corresponding argument curves are included in Figure 9. AST is the BFCL v4 AST score as in Figure 4, in percent.

Training	Low	High	AST
SFT	1.02	1.87	83.66
Suppress low only	0.87	1.93	84.14
Enhance high-order only	1.00	2.23	84.28
SFT on masked inputs x_T	0.99	1.94	84.04
Reversed ICR	1.19	1.53	82.97
Full ICR	0.86	2.26	84.74