
Iterate on an LLM, Serve on an SLM

Keegan Wang

Carnegie Mellon University · Phoebe
keeganwa@andrew.cmu.edu

Accepted to the NeurIPS 2026 2nd Workshop on Agentic AI Benchmarks and Applications for Enterprise Tasks (AABA4ET).

Abstract

As frontier LLMs gain more capabilities, the cost to run them also goes up. At the same time, small language models (SLMs) also gain more capabilities but have largely been able to keep their cost stable, offering $5\text{-}40\times$ cheaper inference for the narrow, repetitive subtasks that agents decompose work into. However, the standing limiting factor is getting the correct behavior out of a small model in a fast, efficient, and economical way. Unfortunately, in most cases, zero-shot small models fail narrow production tasks. Furthermore, retraining one every time the behavior changes is too slow to iterate on. We propose doing both in sequence. First, develop the desired behavior in prompt space, on a frontier model, with immediate feedback from live traffic. Then, once the behavior converges, distill the recorded production behavior (not synthetic imitations of it) into an SLM through supervised fine-tuning (SFT). We show a proper use case of this method through a scheduling agent for home-care agencies that fans out a single yes/no safety decision per caregiver-shift pair on GPT-5.4-nano. A LoRA-tuned Qwen3-8B trained on 11.6k recorded production decisions reaches 0.913 agreement, while the teacher (GPT-5.4-nano), re-run on its own recorded prompts, agrees with itself only 72.7% of the time, pointing to the fact that the SLM may also be more consistent. Furthermore, when the two disagreed, a blinded professional scheduler sided with the student 48-29. Finally, we find that the SLM has $\sim 2.5\times$ lower median latency and is a fraction of the cost. We believe this lifecycle of quick building, live feedback, and cheap inference is a blueprint for others building similar systems.

1 Introduction

Many position papers argue that small language models (SLMs) are the future for agentic systems because agents can decompose work into narrow, repetitive subtasks or function calls [4, 16] and call cheaper and faster purpose-built models rather than one larger LLM. However, those papers leave an important question unanswered. How do you get the correct behavior out of a small model? While a task could be narrow, it doesn't mean that the task is necessarily easy to develop on. In our production system, zero-shot models failed our specific task outright (five off-the-shelf SLMs failed to meet the quality of larger models, and base Qwen3-8B produced malformed tool calls on 23% of cases). Furthermore, training a custom small model for a behavior that you are in the process of designing requires paying the engineering and training cost again and again whenever the core behavior changes. If the desired behavior changes mid-training, the process to train an SLM must be restarted. That expense is what makes most engineers default to a bigger model that just works with minimal prompting. Yet the savings of SLMs are too large to ignore. With $5\text{-}40\times$ lower cost per call, when any one task or function needs to be done thousands of times a day, the total cost saved is almost a no-brainer. So how do we do both?

Our answer is a two-step method to cheaply do both. In phase one, we develop the behavior where iteration is cheapest: the prompt, skills, and tools. While our task’s desired decision policy was still changing, every prompt change gave us immediate feedback that we could see live and users could give feedback on. One of the biggest instances was with an early version of our agent in which it approved only 2 of 49 candidates because soft profile tags were acting as vetoes. That same pool moved to 40 after a one-line prompt change. If we were re-fine-tuning a model for every change we wanted to make, it would take days or weeks, not minutes or seconds, to iterate. Then, in phase two, the economics invert. The frontier model is now an overly large and expensive model, and the small model is the right-sized, cheaper model to run. At the same time, we have already recorded tens of thousands of traces and customer behaviors in which we see approvals and disapprovals in how the agent has acted on its given task. Importantly, these traces also include many edge cases that would be impossible to have thought about if we were to use only synthetic training data. Now we can distill the behavior of the LLM into an SLM with data from production rather than synthetic data.

We show this method on a production workload. For each shift a home-care agency is attempting to fill, an agent makes a yes/no decision for each caregiver, up to 1,000 decisions per outreach. After we reach our desired behavior, we saved 11.6k training examples from over 500 recorded production batches by replaying each prompt, and LoRA-tuned Qwen3 models on the results. Our fine-tuned Qwen3-8B hit 0.913 agreement on held-out cases. Interestingly, when we re-ran a random sample of these traces on GPT-5.4-nano’s original prompts, it gave the same answer only 72.7% of the time.

Thus, our three contributions are as follows. (1) A two-phase method where we first develop our desired behavior in prompt space on a frontier model, then distill the production behavior, rather than synthetic data, into cases to use for fine-tuning. (2) We re-run the teacher on its own recorded prompts and use its self-agreement as the noise floor to decide if the SLM is good enough to use in production. (3) A deployed-workload case study with a blinded evaluation by a professional scheduler that separates quality errors from benign errors.

2 The task

Currently, when a home-care agency wants to fill a client’s shift, a human scheduler must reach out to N caregivers, where N can be in the hundreds, to ask if they want to take the shift. Rather than have one model reason over hundreds of candidates with hundreds of lines of information in a single context window, the production system forks one evaluation model per candidate. Each subagent sees the shift’s context and request, then has a single tool-call output deciding to add or reject the caregiver to the shift outreach along with the reasoning.

What makes this decision hard are the unstructured data, conflicting or missing data, and differing definitions of “conflict” between clients, caregivers, organizations, and schedulers. Thus we choose to use an LLM rather than a rule-based decision system.

Our production runs this system on GPT-5.4-nano at high reasoning effort, and each outreach can scale up to 1,000 concurrent decisions at a time.

3 Training data by byte-exact replay

To ensure that our training data directly matched production systems, every production agent decision is durably recorded. Our extractor then replays each recorded batch through the same production pipeline, using the same prompt builder and serializer that the live evaluator uses, which produces the identical chat context that would be served to a deployed model. It then pairs it with the recorded tool call as the target. This differs from the dominant synthetic-trajectory approach [25, 17, 13]. Instead, the training distribution is the serving distribution.

Contract filtering. Recorded rejects in which the reasoning lacks a hard-conflict justification are purposely dropped to ensure that training does not teach the violation. We also dedupe decisions that overlap, keep the most recent decisions, and hold a train/validation split by source batch. Our training corpus ended up being 549 production batches with 11,621 train and 1,604 validation examples across 17 different organizations.

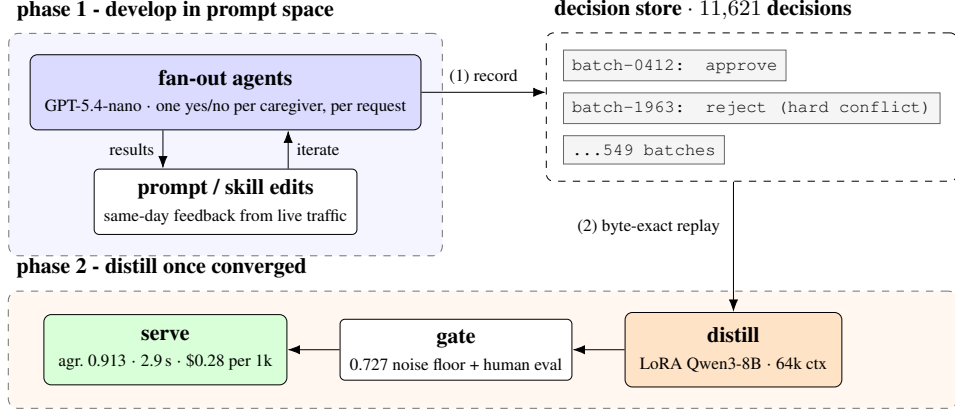


Figure 1: The two-phase lifecycle. Phase 1: the deployed agent runs the task on a frontier model while the behavior is iterated in prompt space, and every trace is recorded. Phase 2: once the behavior converges, the stored traces are used as training data, and an SLM is distilled from them, gated on the teacher’s self-agreement noise floor plus a blinded human evaluation.

Table 1: Evaluation against recorded production decisions (the 880 of the 1,604 validation cases whose prompts fit the serving context, with the teacher noise floor on a seed-fixed random 150-case subsample of the same set). *Base-model agreement covers only its 676 scored cases. The 23% it failed to answer are excluded, so it reads selection-biased high.

	Teacher floor (GPT-5.4-nano)	Base 8B (zero-shot)	Tuned 4B	Tuned 8B
Agreement w/ recorded decisions	0.727	0.774*	0.863	0.913
Malformed outputs	0	203 (23%)	106 (12%)	0
False-approve rate (rec. rejects)	0.218	0.208	0.231	0.232
False-reject rate (rec. approves)	0.305	0.236	0.091	0.011
Hard-conflict prefix compliance	1.00	1.00	1.00	1.00
Latency p50/p95 (s)	7.1 / 13.1	14.9 / 36.6	2.6 / 3.7	2.9 / 4.3

Fine-tuning. We use LoRA [12], at the training provider’s default adapter rank, to fine-tune Qwen3-4B and Qwen3-8B [27] on this corpus: constant LR 10^{-4} , 1,023 optimizer steps $\approx$ 1B training tokens $\approx$ 3 passes, at 65,536 max context. Final eval loss: 0.534 (4B), 0.254 (8B).

4 Evaluations

To compare our fine-tuned models to the teacher, and decide if we could use the new model without regression, we replay held-out validation prompts with the decision tool forced at temperature 0 and score against the held-out production decisions. The live model itself, re-run on a random 150 of its own recorded prompts, only agreed with itself 72.7% of the time (95% CI 0.65-0.79) [2]. Using this, we can then interpret our decision agreement, false-approve rate on recorded rejects, false-reject rate on recorded approves, hard-conflict prefix compliance, malformed output rate, and latency compared to the teacher.

5 Results

We have four main results (Table 1). The teacher (the production GPT-5.4-nano agent) scores against its own recorded decisions as a self-agreement noise floor. The tuned 8B Qwen model scoring 0.913 (95% CI 0.89-0.93) is more consistent with the existing traces than the teacher itself. When measured against the teacher’s original decisions, the disagreement concentrates in the benign direction (false rejects) and the false-approve rate is flat across each arm. While the 4B model had decent decision making, its 12% malformed output rate makes it a bad candidate for production use. Therefore the capability floor likely sits between 4B and 8B parameters.

Table 2: Accuracy compared to human labels, stratified and weighted to the full 880-case validation set. Error directions also reported for the two arms that decide every case.

vs. human labels	Teacher (recorded)	Base 8B	Tuned 4B	Tuned 8B
Accuracy (reweighted)	0.866	0.705	0.797	0.888
False-approve (human rejects)	0.199	—	—	0.273
False-reject (human approves)	0.098	—	—	0.024

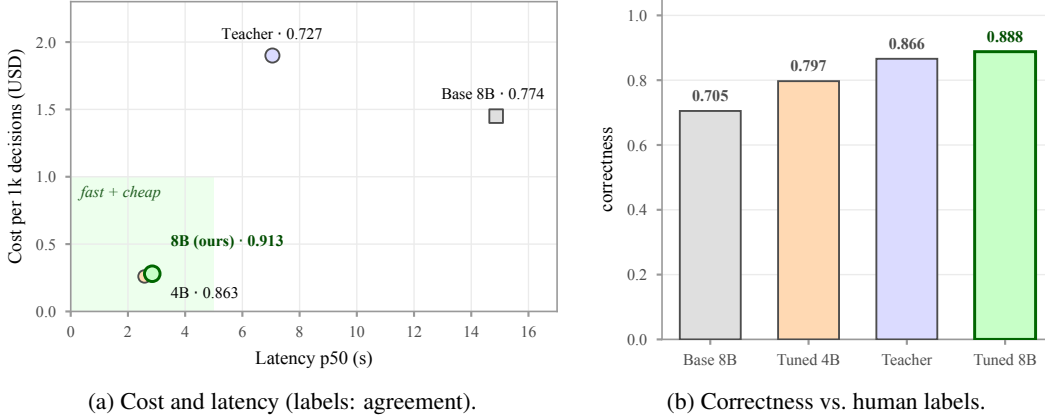


Figure 2: The four arms compared. (a) Only the tuned models land in the fast-and-cheap quadrant (shaded). The tuned 8B is $\sim 7\times$ cheaper and $2.5\times$ faster than the teacher. Teacher plotted at its cached-prompt cost (\$1.9 per 1k, \$7.7 uncached). SLM costs are price estimates at full utilization based on industry virtual GPU pricing. (b) Accuracy against the blinded human labels, post-stratified and reweighted to the full validation set (Table 2): the tuned 8B is the most correct model, and the zero-shot base is the least correct arm.

We also find that the zero-shot, un-fine-tuned Qwen3-8B fails to produce a usable tool call on 23% of cases and takes a median of 14.9 s before outputting. The tuned model eliminates the malformed outputs and answers $5\times$ faster, so using Qwen3-8B without the fine-tuning would not have been a viable alternative.

5.1 Human ground truth

Next, to test the agents’ correctness against the real world, we ran a blinded 192-case evaluation by a professional home-care scheduler. We included all 77 student-teacher disagreements as well as stratified samples of agreeing rejects (49) and approves (66) to ground the comparison in human judgement. Because the sample deliberately oversamples disagreements, we post-stratify by the student-vs-recorded outcome, whose population sizes over the full 880-case validation set are known (77 disagreements, 235 agreeing rejects, 568 agreeing approves).

Head-to-head disagreement. On the 77 disagreement cases the human sided with the tuned 8B 48 times vs. 29 for the production decision (62%-38%, 95% CI 0.51-0.72, exact binomial two-sided $p = 0.040$).

What human grounding reveals. Against the teacher’s own labels, the student’s false-approve rate looked flat. However, against human labels, an asymmetry appears (Table 2). The student approves somewhat more caregivers the human would reject (0.273 vs. 0.199) while cutting false rejects from 0.098 to 0.024. The net difference is more accurate and far less over-filtering, but slightly more permissive toward caregivers that a human would not add to an outreach. Note that we generally find that humans add fewer caregivers to outreaches than LLMs, even if all caregivers are good fits.

5.2 Cost and latency

Figure 2a places the four arms on the cost-latency plane. At list prices and measured token distributions, the teacher costs $\sim \$1.9$ - $\$7.7$ per 1k decisions depending on prompt-cache state. A dedicated

single-GPU deployment of the tuned 8B at the observed evaluation throughput lands at $\sim$ \$0.19-\$0.38 per 1k at full utilization (A100- and H100-class GPU-hour prices). The tuned 8B is better than every other usable arm. It is simultaneously the cheapest of the arms that answer every case, the most consistent (0.913), and within a third of a second of the fastest arm (the 4B, ruled out by its malformed outputs), while the zero-shot base model is dominated on every axis.

6 Related work

SLMs for agents. Belcak et al. [4] argue agentic workloads are SLM-shaped. Surveys map the capability landscape [16]. We contribute a deployed instance with measured stakes.

Distillation from teacher outputs. Classic knowledge distillation [10] and its LLM lineage (rationale distillation [11], instruction imitation [25, 17], and agent-trajectory distillation [13, 28]) mostly train on synthetic teacher generations. Our targets are recorded production decisions replayed byte-exactly.

Tool-use tuning of small models [19, 21, 15, 14, 8] shows small models can match frontier function calling. TinyAgent [8] is the closest deployed example.

Students surpassing noisy teachers. Born-again networks [9], noisy student [26], and weak-to-strong generalization [5] all show that a student can end up better than the teacher it learned from. We see the same thing in a production setting. Since the teacher only agrees with itself 72.7% of the time [24, 29, 2], it acts less like ground truth and more like a noisy annotator [7, 23, 20].

Context distillation [1, 22] takes what a prompt does and trains it into the model’s weights. Our method does the same thing one level up. Instead of a single prompt, we distill a whole behavior that was shaped through prompt iteration.

Cost-motivated routing [6, 18] cuts costs by sending easier requests to cheaper models. We instead replace the entire subtask with one small model. Industry case studies on compressing models for production serving report the same kind of savings we find [3].

7 Limitations and conclusion

Limitations. As of writing this paper, the distilled model is not yet serving production traffic. Every number here is an offline replay, and the true quality test is an online A/B test in production, where prompt-cache dynamics, traffic mix, and downstream scheduler behavior all contribute to cost, correctness, and latency. Our ground-truth labels come from a single professional scheduler, and our statistics do not model correlation between cases from the same organization or batch. Finally, results cover one subtask, one domain, and one teacher. Thus, further work is needed to prove this method effective for a wide range of engineering tasks.

Conclusion. To make good SLM systems, one should first develop the behavior where iteration is cheap (in prompt space on a frontier model) and, once the behavior converges, compile it into a small model trained on the recorded behavior itself. On our production workload, this yielded an 8B SLM that executes the policy more reliably than the teacher that defined it, at a fraction of the cost and latency, and, by blinded expert judgement, more accurately as well. This protocol of quick building, live feedback, then cheap inference should be a blueprint for others building similar systems.

References

- [1] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment, 2021. arXiv:2112.00861.
- [2] Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. Non-determinism of “deterministic” LLM settings, 2024. arXiv:2408.04667.

- [3] Kayhan Behdin et al. Scaling down, serving fast: Compressing and deploying efficient LLMs for recommendation systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1687–1702, 2025. arXiv:2502.14305.
- [4] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic AI, 2025. arXiv:2506.02153.
- [5] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. arXiv:2312.09390.
- [6] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance, 2023. arXiv:2305.05176.
- [7] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1): 20–28, 1979.
- [8] Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. TinyAgent: Function calling at the edge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2024.
- [9] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- [10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014. arXiv:1503.02531.
- [11] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.
- [12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. arXiv:2106.09685.
- [13] Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling LLM agent into small models with retrieval and code tools. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.17612.
- [14] 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 Kam-Fai Wong. ToolACE: Winning the points of LLM function calling, 2024. arXiv:2409.00920.
- [15] 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, 2024. arXiv:2406.18518.
- [16] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights, 2024. arXiv:2409.15790.
- [17] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4, 2023. arXiv:2306.02707.

- [18] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data, 2024. arXiv:2406.18665.
- [19] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs, 2023. arXiv:2305.15334.
- [20] Barbara Plank. The “problem” of human label variation: On ground truth in data, modeling and evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.
- [21] 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, 2023. arXiv:2307.16789.
- [22] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. arXiv:2209.15189.
- [23] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [24] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.
- [25] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshnab, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.
- [26] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves ImageNet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [27] An Yang et al. Qwen3 technical report, 2025. arXiv:2505.09388.
- [28] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for LLMs, 2023. arXiv:2310.12823.
- [29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2023.