Meta-Reasoning Improves Tool Use in Large Language Models

Lisa Alazraki
Imperial College London
lisa.alazraki20@imperial.ac.uk

Marek Rei
Imperial College London
marek.rei@imperial.ac.uk

Abstract

External tools help large language models succeed at tasks where they would otherwise typically fail. In existing frameworks, choosing tools at test time relies on naive greedy decoding, regardless of whether the model has been fine-tuned on tool-annotated data or prompted with in-context examples. In contrast, we find that gathering and choosing among a suitable set of candidate tools has greater potential to lead to an optimal selection. We present **T**ool sel**ECT**ion via meta-reas**ON**ing (TECTON), a two-phase system that first *reasons* over a task and outputs candidate tools using a custom fine-tuned language modelling head. Then, with the custom head disabled, it *meta-reasons* (i.e., it reasons over the previous reasoning process) to make a final choice. We show that TECTON results in substantial gains—both in-distribution and out-of-distribution—on a range of math reasoning datasets.

1 Introduction

Augmentation with external tools has proven effective at boosting the performance of large language models (LLMs) in knowledge-intensive tasks such as QA and math problem-solving (Hao et al., 2023; Paranjape et al., 2023; Parisi et al., 2022; Schick et al., 2023). Tools are self-contained programs or APIs which the model can execute with chosen arguments as needed. To teach an LLM how to use tools, previous work adopts one of three main strategies: (1) tool demonstrations via in-context learning (ICL) (Gao et al., 2023; Gupta and Kembhavi, 2023; Hsieh et al., 2023; Surís et al., 2023), (2) full model fine-tuning on a dataset where text samples are interleaved with tool annotations (Parisi et al., 2022; Schick et al., 2023; Tang et al., 2023; Patil et al., 2023), or (3) parameter-efficient fine-tuning (PEFT) on tool annotated data (Hao et al., 2023; Qiao et al., 2024; Wang et al., 2024). Similarly to full fine-tuning, PEFT methods can teach LLMs a very large number of tools, while ICL is limited by the fixed size of the context window (Hao et al., 2023; Patil et al., 2023). Although the fine-tuning paradigm in general binds the model to the set of tools learned during training, parameter-efficient learning reduces the cost of tuning, thus facilitating potential future extensions of the tool set. In contrast, adding further tools via a new round of full-model tuning incurs a significant computational cost (Hao et al., 2023). We note that a further advantage of PEFT is that it tunes a handful of additional task-specific parameters, which can be selectively disabled to reinstate the frozen model and its original capabilities (Ding et al., 2023; Han et al., 2024).

In existing literature, inference-time tool selection is made by greedily decoding the most likely tool (Gao et al., 2023; Hao et al., 2023; Schick et al., 2023; Wang et al., 2024), regardless of whether the model has been fully fine-tuned, PEFT-tuned, or prompted in-context. In this work, we revisit the paradigm that selects tools solely based on their probability at decoding time. We propose an alternative, novel framework for **T**ool sel**ECT**ion via meta-reas**ON**ing (TECTON) that gathers and chooses among a suitable set of candidate tools. We train a parameter-efficient language modelling head on tool-annotated data, similar to Hao et al. (2023), which can be switched on or off as needed. TECTON thus comprises two distinct phases: in the *reasoning phase*, it investigates a task and outputs candidate tools with the aid of the tuned LM head. Then, in the *meta-reasoning phase*, it uses the frozen LLM to re-examine the candidates and make a final decision. Fig. 1 illustrates the framework. We train and evaluate TECTON on math reasoning datasets, following established work on LLM tool calling (Chen et al., 2024; Das et al., 2024; Gou et al., 2024). Among tasks that benefit from tools, math reasoning is particularly challeng-

Findings of the Association for Computational Linguistics: NAACL 2025, pages 7900–7912
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

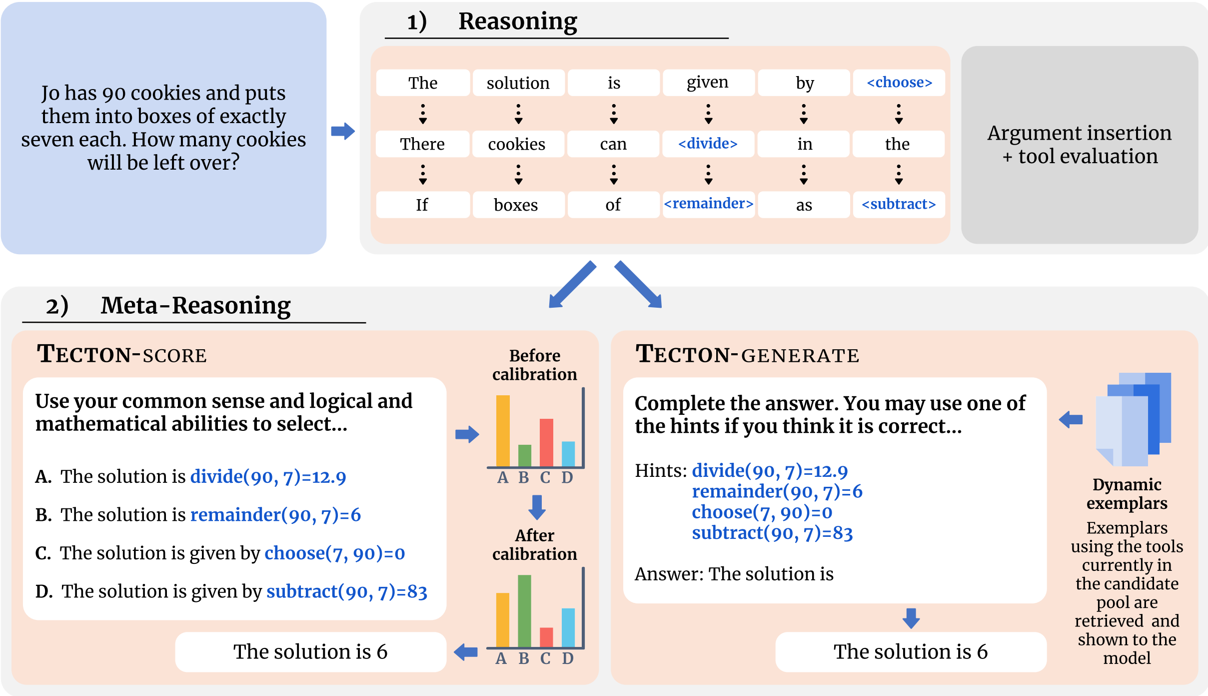

Figure 1: An overview of TECTON. In the reasoning phase, the system inspects the task and decodes a set of candidate tools, followed by argument insertion and evaluation of the tools via the Python interpreter. In the meta-reasoning phase, the model is asked to select the most useful tool, either by scoring multiple options (TECTON-SCORE) or by continuing the generation given the decoded tools as hints (TECTON-GENERATE).

ing, since it requires chains of multiple tool calls with errors that compound. This is evidenced by existing tool-augmented LLMs, which achieve the lowest performance on math reasoning when evaluated on multiple tasks (Hao et al., 2023; Schick et al., 2023). In summary, our main contributions are:

- We introduce TECTON, a novel two-phase framework that combines a custom fine-tuned head with a frozen LLM to improve tool use (Section 2).
- We show that TECTON outperforms strong baselines in math reasoning tasks, both on in-distribution data and on unseen benchmarks (Section 3).
- We enhance three popular math reasoning datasets to make them more challenging for current LLMs. We share our data and code at `https://github.com/lisaalaz/tecton`.

2 Method

2.1 Preliminaries

We augment the language modelling head of a base model with additional token embeddings T to represent math operations, and train them via a standard language modelling objective. Once trained, T comprises the tools available to the LLM for solving math problems. Prior work has shown the effectiveness of tuning additional tokens for both math tasks (Hao et al., 2023; Wang et al., 2024) and general reasoning (Goyal et al., 2024; Herel and Mikolov, 2024).

Our preliminary experiments show that in cases where the LLM has failed to generate the correct tool by greedy sampling, this can usually be found among tokens that have only slightly lower probabilities (Figure 3). Over-sampling an appropriate set of candidate tools may thus be a better strategy than greedily decoding the most likely tool, provided this is combined with a reliable method for choosing among the candidates. To this end, we design a two-phase framework that leverages both the specialised, augmented LM head (*reasoning phase*) and the underlying generalist LLM (*meta-reasoning phase*). The rest of the section describes this in detail.

2.2 Reasoning Phase

Given a math problem where individual reasoning steps are separated by newline tokens, we ask the LLM to solve it line by line, and collect a set of candidate tools for each line using the augmented LM head. We experiment with both temperature sampling and greedy decoding of multiple tokens (see Appendix A). We find that looking at the top k most likely tokens at every position in the sequence is the most promising approach to gather a diverse yet relevant pool of candidates. Hence, we generate an intermediate solution to each line of the problem, gathering tools from the top k tokens at each decoding step. For each line, the multiset C of candidate tools is given by $C = \{\{W_{ij} \in T, 1 \leq i \leq l, 1 \leq j \leq k\}\}$, where T is the set of available tool tokens, and W is the matrix resulting from decoding k top-probability tokens at each of the l token positions in the line of text. We set $k = 5$ as a trade-off between search space size and computation cost. We then prompt the LLM to produce arguments for each candidate tool given the previous context. Identical tools with the same arguments are dropped from the pool. Finally, we pass each candidate tool and arguments into the Python interpreter, and keep those that are successfully evaluated. Once a line of text toward the solution has been processed in this way, we move onto the meta-reasoning phase.

2.3 Meta-Reasoning Phase

In this phase, we disable the custom-tuned head and let the underlying LLM analyse its previous reasoning process to choose among the candidate tools. Frozen LLMs have been used to self-evaluate and meta-reason over previous answers in existing literature (Alazraki et al., 2023; Shinn et al., 2023; Yao et al., 2023a; Zeng et al., 2024b). We experiment with two ways of eliciting meta-reasoning: TECTON-SCORE and TECTON-GENERATE.

TECTON-SCORE. We join each candidate tool with the previous context, and present these as options for the LLM to *score*. We prefix each option with an uppercase letter label and select as the answer continuation the option whose label is assigned highest probability by the model, i.e., arg $\max_{t_i \in \mathcal{V}_{\text{sub}}} p(t_i \mid t_{<i}$ with $t_{<i} \in \mathcal{V})$, where $\mathcal{V}_{\text{sub}}$ denotes a subset of the vocabulary containing only the uppercase letter tokens that are in the label set. In this setup, we limit the number of candidates to a maximum of four.

TECTON-GENERATE. We pass the candidate tools as hints and ask the model to *generate* an appropriate continuation of the answer. Here, the hints serve as mere guidance for the LLM (i.e., the model could choose to ignore all candidates and generate something different). In this version of the system, the model benefits from dynamically retrieved few-shot exemplars demonstrating the tools in the candidate set.

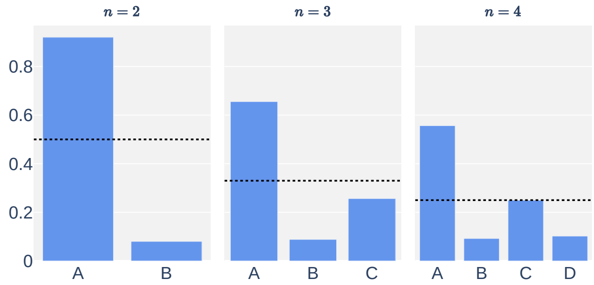

Figure 2: Averaged biased probability distributions over n labels (for $n = \{1, 2, 3\}$), obtained on GSM8K-XL's validation set. The dotted lines indicate the uniform averaged distribution that would be given by an unbiased model. Note that the label distribution is similarly skewed for FuncQA.

2.4 Bias Calibration

Upon running TECTON-SCORE without recalibration of the label probabilities, we find that the validation results are poor. Visual inspection of the samples reveals that the model assigns highest likelihood to the same label in most instances, as shown in Fig. 2. This is consistent with Zheng et al. (2024)'s finding that LLMs are prone to selection bias in multiple-choice tasks. To solve a similar problem, Duarte et al. (2024) measure their model's bias over the labels A, B, C, D using a set of neutral samples where a uniform distribution would be expected, and subtract that bias from each label's likelihood at inference time. Our math reasoning task does not lend itself to finding neutral samples, so we adopt a different strategy. Having created data samples of questions and options (by running the reasoning phase of TECTON on a validation set), we compute $n!$ permutations of the n options while keeping the letter labels in the same position. We have the model score the labels for each permutation, and average over all permutations and all data samples to obtain an averaged biased distribution $B^{(n)}$. Since the number of options in our task is variable, we run this process independently for data samples with $n = 2$, $n = 3$ and $n = 4$ labels. At inference time, given a set of labels $L^{(n)}$ of size n, we retrieve the corresponding $B^{(n)}$ and compute the calibrated probability $\hat{p}_i$ of each label l_i in the set as

$$\hat{p}_i = p_i + \frac{1}{n} - {B^{(n)}}_i$$

where p_i is the probability assigned by the model to label l_i for the current sample, ${B^{(n)}}_i$ is the pre-computed biased probability of label l_i, and n is the total number of labels.

2.5 Retrieval of Tool Demonstrations

To aid answer generation in TECTON-GENERATE, we dynamically retrieve and add to the context few-shot exemplars demonstrating the candidate tools. We create the retrieval pool by extracting training samples and collecting candidate tools for each, by running the reasoning phase of TECTON. We construct each exemplar to simulate the inference task, as follows: (1) we append to the sample its set of candidate tools, and (2) we append to it the golden answer demonstrating how the correct tool is used to obtain the final solution. At inference time, we retrieve only the exemplars whose golden answers contain the tools currently in the candidate set. It is worth noting that dynamic retrieval was not included in TECTON-SCORE as it did not improve validation performance.

3 Experiments

3.1 Experimental Setup

Our system is model-agnostic and can be applied to any open-weights LLM. Here, we use Llama 3 8B Instruct (Dubey et al., 2024) (henceforth referred to as Llama 3) as the base model in all experiments. Implementation details and hyperparameters are given in Appendix B.

3.2 Datasets

We train and evaluate TECTON on GSM8K-XL (Cobbe et al., 2021; Hao et al., 2023) and FuncQA (Hao et al., 2023). The test set of the latter is comprised of two distinct subsets: a 'one-hop' corpus containing problems solvable with one single operation (FuncQA-OH), and a 'multi-hop' one requiring multiple operations (FuncQA-MH). For GSM8K-XL we tune four additional tokens corresponding to the four basic operations, and extend these to 13 in the case of FuncQA. The complete set of tools for each dataset is shown in Appendix B. Additionally, we evaluate on out-of-distribution datasets that were not observed during fine-tuning. For this purpose we choose a range of math reasoning datasets: ASDiv (Miao et al., 2020), MAWPS (Koncel-Kedziorski et al., 2016) and SVAMP (Patel et al., 2021). These were found by Ott et al. (2023) to be OOD with respect to GSM8K: not only do they display minimal n-gram overlap, but they also require reasoning chains of different average lengths from GSM8K to be solved. These datasets have been used to evaluate LLMs in previous works, both with and without the aid of tools (Kojima et al., 2022; Schick et al., 2023). Following a strategy similar to the one used by Hao et al. (2023) for constructing GSM8K-XL, we magnify the numbers in these datasets to make them challenging for current LLMs (the enhancement process is described in Appendix D). We thus obtain ASDiv-XL, MAWPS-XL, and SVAMP-XL. We use these datasets to test models tuned on GSM8K-XL.

3.3 Baselines

We implement recent tool-augmented models as baselines: TRICE (Qiao et al., 2024) and ToolkenGPT (Hao et al., 2023). These share a parameter-efficient approach with TECTON (see Appendix C for details). Despite their relatively low computational cost, they have been shown to outperform strong systems: TRICE paired with Alpaca (Taori et al., 2023), ChatGLM (Zeng et al., 2024a) and Vicuna (Zheng et al., 2023) surpasses the much larger GPT-3.5 as well as tool learning via supervised fine-tuning. In addition to outperforming GPT-3.5, ToolkenGPT paired with LLaMA (Touvron et al., 2023) is more accurate than ReAct (Yao et al., 2023b). It should also be noted that LLMs are increasingly able to solve math problems and perform difficult arithmetic without tools, as shown by the high accuracy (79.6%) achieved by Llama 3 on the non-enhanced version of GSM8K (Dubey et al., 2024). Hence we additionally compare against a vanilla version of Llama 3 as well as Chain-of-Thought (CoT) prompting (Wei et al., 2022) with exemplars extracted from the train set.

3.4 Results

Table 1 shows TECTON's gains on math reasoning. Both versions of the system achieve scores above baselines for in-distribution and out-of-distribution-data, with the exception of TECTON-GENERATE on GSM8K-XL, whose performance is slightly below that of ToolkenGPT.

In-distribution performance. On GSM8K-XL, our best implementation scores 7.2 percentage points above TRICE and 2.3 above ToolkenGPT. When evaluated on in-distribution data, TECTON's most significant gains are on FuncQA: the GENERATE setting doubles the performance of

	In-distribution			Out-of-distribution		
	FuncQA-OH	FuncQA-MH	GSM8K-XL	ASDiv-XL	MAWPS-XL	SVAMP-XL
Llama 3	10.0	2.9	13.0	25.8	26.2	27.8
+ CoT	25.0	5.9	37.3	40.6	58.7	51.9
+ TRICE	-	-	43.5	52.2	71.2	49.6
+ ToolkenGPT	65.0	10.3	48.4	45.3	68.3	60.4
+ TECTON-SCORE	66.7	17.6	**50.7**	53.6	76.9	62.2
+ TECTON-GENERATE	**70.0**	**20.6**	45.8	**55.3**	**77.4**	**66.7**

Table 1: Accuracies on math reasoning datasets measured via exact match of the result rounded to 2 decimal places. We do not tune TRICE on FuncQA as this dataset lacks the necessary annotations for RLEF training. Note that the separation between in-distribution and out-of-distribution data does not apply to the vanilla version of Llama 3.

	In-distribution			Out-of-distribution		
	FuncQA-OH	FuncQA-MH	GSM8K-XL	ASDiv-XL	MAWPS-XL	SVAMP-XL
TECTON-SCORE **– bias calibration**	58.3	8.8	49.3	52.2	75.0	61.9
TECTON-GENERATE **– dynamic exemplar retrieval**	58.3	13.2	43.5	50.8	71.2	62.9

Table 2: Results of ablating components of TECTON-SCORE and TECTON-GENERATE, on in-distribution and out-of-distribution data.

ToolkenGPT in the challenging multi-hop task, reaching 20.6% accuracy. For context, Llama 3 can only solve 2.9% of the dataset and CoT only raises performance to 5.9%.

Out-of-distribution performance. On OOD datasets, TECTON's advantage is substantial: on average, TECTON-GENERATE gains 8.5 percentage points over ToolkenGPT and 8.8 over TRICE. TECTON-SCORE achieves 6.2 and 6.6 above the two baselines, respectively. This highlights the adaptability of the method to unseen data.

3.5 Ablations

To gain more insight into these results, we perform an ablation study on the meta-reasoning phase of TECTON, shown in Table 2. We ablate bias calibration from TECTON-SCORE and dynamic exemplar retrieval from TECTON-GENERATE. On average, TECTON-SCORE's ablated accuracy is 6.2 percentage points lower than the non-ablated version on in-distribution data, and 1.2 on unseen datasets. Additionally, TECTON-GENERATE's average performance drops by 7.2 on in-distribution data and 4.9 on OOD data. The most significant performance loss is on FuncQA: the ablated versions of TECTON-SCORE and TECTON-GENERATE see a decrease of 8.4 and 11.7 percentage points, respectively, on the one-hop test set. They also drop by 8.8 and 7.4, respectively, on the multi-hop split. On GSM8K, the decline is less severe: -1.4 for TECTON-SCORE and -2.3 for TECTON-GENERATE. TECTON-SCORE's decrease after ablation is similarly modest on OOD datasets, while TECTON-GENERATE's is more significant: its ablated performance drops by 6.2 points on MAWPS, 4.5 on ASDiv and 3.8 on SVAMP. This highlights the benefit of dynamic exemplar retrieval during the meta-reasoning phase of TECTON-GENERATE. Despite the performance decline, we find that both ablated versions of TECTON outperform CoT on all datasets. They also consistently surpass ToolkenGPT's accuracy on OOD datasets, and either match or outperform TRICE, with the sole exception of TECTON-GENERATE on ASDiv-XL.

4 Conclusion

We introduce TECTON, a novel two-phase framework that first samples a set of candidate tools and then selects the optimal candidate via meta-reasoning. We implement two versions of the system and find that both achieve superior performance on math reasoning datasets, surpassing our strongest baseline by $\sim 4\%$ on in-distribution data and $\sim 9\%$ on unseen benchmarks, on average. These results confirm our hypothesis that a specialized, custom-tuned framework and a generalist pre-trained model can work together to improve tool use in challenging tasks.

Limitations

This paper solely focuses on math reasoning tasks. While this is consistent with established literature, there are other domains (e.g., knowledge-intensive QA, virtual environment navigation) that can benefit from the use of tools. Future work can investigate a wider range of tasks.

Ethical Considerations

We have verified that all datasets and software utilized in this paper allow for their use, distribution and modification. Our non-commercial purpose is consistent with all licenses. The distribution of our code and data is accompanied by the licenses and credits to the original authors.

Acknowledgments

The authors would like to thank Joe Stacey for his insightful comments on the first draft of this paper.

References

Lisa Alazraki, Lluis Castrejon, Mostafa Dehghani, Fantine Huot, Jasper Uijlings, and Thomas Mensink. 2023. How (not) to ensemble LVLMs for VQA. In *Proceedings on "I Can't Believe It's Not Better: Failure Modes in the Age of Foundation Models" at NeurIPS 2023 Workshops*, volume 239 of *Proceedings of Machine Learning Research*, pages 1–20. PMLR.

Nuo Chen, Hongguang Li, Baoyuan Wang, and Jia Li. 2024. From good to great: Improving math reasoning with tool-augmented interleaf prompting. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pages 64–79, Bangkok, Thailand. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Debrup Das, Debopriyo Banerjee, Somak Aditya, and Ashish Kulkarni. 2024. MATHSENSEI: A tool-augmented large language model for mathematical reasoning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 942–966, Mexico City, Mexico. Association for Computational Linguistics.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Yang Zonghan, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5:1–16.

André V. Duarte, Xuandong Zhao, Arlindo L. Oliveira, and Lei Li. 2024. DE-COP: Detecting copyrighted content in language models training data. *Preprint*, arXiv:2402.09910.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *Proceedings of the Twelfth International Conference on Learning Representations*, ICLR'24.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *Proceedings of the Twelfth International Conference on Learning Representations*, ICLR'24.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Preprint*, arXiv:2403.14608.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, pages 45870–45894. Curran Associates, Inc.

David Herel and Tomas Mikolov. 2024. Thinking tokens for language modeling. *Preprint*, arXiv:2405.08644.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *Preprint*, arXiv:2308.00675.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022. Lila: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Simon Ott, Konstantin Hebenstreit, Valentin Liévin, Christoffer Egeberg Hother, Milad Moradi, Maximilian Mayrhauser, Robert Praas, Ole Winther, and Matthias Samwald. 2023. Thoughtsource: A central hub for large language model reasoning data. *Scientific Data*, 10:528.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. ART: Automatic multi-step reasoning and tool-use for large language models. *Preprint*, arXiv:2303.09014.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: Tool augmented language models. *Preprint*, arXiv:2205.12255.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive APIs. *Preprint*, arXiv:2305.15334.

Shuofei Qiao, Honghao Gui, Qianghuai Jia, Huajun Chen, and Ningyu Zhang. 2024. Making language models better tool learners with execution feedback. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual inference via python execution for reasoning. *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *Preprint*, arXiv:2306.05301.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMa: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Xinyi Wang, Lucas Caccia, O. Ostapenko, Xingdi Yuan, and Alessandro Sordoni. 2024. Guiding language model reasoning with planning tokens. *Preprint*, arXiv:2305.15334.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the Eleventh International Conference on Learning Representations*, ICLR'23.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024a. ChatGLM: A family of large language models from GLM-130b to GLM-4 All Tools. *Preprint*, arXiv:2406.12793.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2024b. MR-GSM8K: A meta-reasoning benchmark for large language model evaluation. *Preprint*, arXiv:2312.17080.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Hao Zhang, Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Siyuan Zhuang, Yonghao Zhuang, Gonzalez Joseph E, Ion Stoica, and Eric P Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality.

A Preliminary Experiments

A.1 Top-*k* Decoding

We run ToolkenGPT (in its original implementation based on the first version of LLaMA) on the validation set of FuncQA. We find that in samples where the system has generated an incorrect tool by greedy sampling, the correct one—complete with correct arguments—is among the top five most likely tokens in over 60% of cases. These include samples where the correct tool can be found by searching the top tokens at a *different* position from the one that has produced the incorrect tool. Overall, the system decodes the correct tool (regardless of the arguments it generates for it) in 76.9% of samples; this rises by over 10% to 87.2% when we consider the top $k = 5$ tokens at each decoding step (Fig. 3). Further increasing k to 10 incurs a higher computational cost without raising the proportion of decoded golden tools.

A.2 Temperature Sampling

We experiment with temperature sampling and find that it is not an optimal strategy to gather candidate tools, as lower temperatures do not lead to enough diversity while higher ones generate irrelevant tools.

A.3 Tool Selection by Self-Consistency

We select tools by self-consistency on FuncQA's validation set. We try choosing the tool that is most represented in the candidate set both before and after argument insertion. In both cases, We find that tool choice by self-consistency performs poorly (over 6% below ToolkenGPT in the best case).

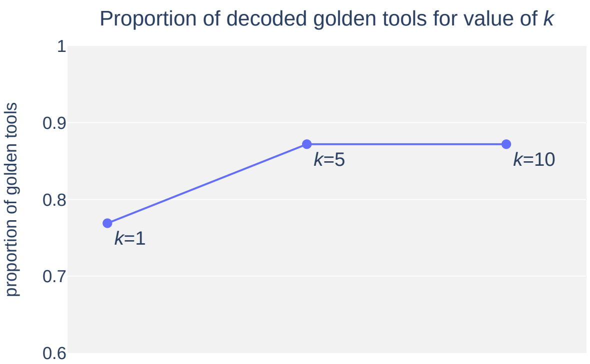

Figure 3: Proportion of golden tools across the FuncQA validation set, for *k*=1, *k*=5 and *k*=10, where *k* is the number of top-probability tokens at each decoding step.

B Model Implementation and Training

We do all training and inference on a single NVIDIA Tesla V100 GPU.

B.1 Tools

GSM8K-XL and FuncQA (Hao et al., 2023) are annotated with four and 13 tools respectively, each representing a math operation. We illustrate these in Table 3. Tools are trained as additional tokens added to the standard language modelling head of Llama 3, which comprises 128,256 token representations. Therefore, we extend these representations to 128,269 in the case of FuncQA and 128,260 for GSM8K-XL.

B.2 Details of the Training Process

The LM head of TECTON consists of the standard head of Llama 3 8B Instruct (Dubey et al., 2024) concatenated with an additional linear layer of size $embedding\_dimension \times number\_of\_tools$. The tool token embeddings are randomly initialized and trained on math reasoning QA pairs from GSM8K-XL and FuncQA, following a standard language modelling objective. Both datasets are annotated with the token positions at which each tool should be generated. Note that the original datasets made available by Hao et al. (2023) are annotated according to SentencePiece tokenization (Kudo and Richardson, 2018). We edit the annotations to be compatible with Llama 3's Byte-Pair Encoding tokenizer (Sennrich et al., 2016). We tune two distinct sets of special tokens, one on each dataset, as each requires a different set of tools as shown in Table 3. We train at half precision (FP16) for ten epochs, using learning rates $\{1e{-}4,\ 1e{-}3\}$, batch size 1, and saving checkpoints at each epoch. We select the best checkpoint by measuring performance on a validation set.

Table 4 gives an overview of our training, validation, and testing data. It should be noted that GSM8K-XL's training and validation sets coincide with those of GSM8K, as only the test portion of the dataset was enhanced by Hao et al. (2023). Table 5 reports the hyperparameter combinations of our chosen checkpoints.

B.3 Inference-time Hyperparameters

At inference, we decode with temperature $t = 0$, $p = 0.95$, $k = 5$. In the reasoning phase, we apply logit bias to the tool tokens to promote their generation. We use logit bias 3.0 for GSM8K-XL and 4.0 for FuncQA. Since our generation strategy is deterministic, all our accuracies are reported on a single run, rounded to 1 decimal place.

Task	Tools
GSM8K-XL	<add> <subtract> <multiply> <divide>
FuncQA	<add> <subtract> <multiply> <divide> <power> <sqrt> <log> <ln> <lcm> <gcd> <remainder> <choose> <permutate>

Table 3: Math reasoning datasets and their corresponding tools.

Dataset	Train	Validation	Test
GSM8K-XL	5054	1000	568
FuncQA	611	39	128
ASDiv-XL	N/A	N/A	360
MAWPS-XL	N/A	N/A	416
SVAMP-XL	N/A	N/A	270

Table 4: Sizes of our training and testing datasets. Note that we use ASDiv-XL, MAWPS-XL and SVAMP-XL only for testing. For FuncQA the reported test data size includes both splits—FuncQA-OH and FuncQA-MH.

Dataset	Epoch	Learning Rate
GSM8K-XL	9	1e−3
FuncQA	4	1e−4

Table 5: Hyperparameter combinations of the chosen model checkpoints for GSM8K-XL and FuncQA.

C Baselines Implementation

All baselines are built upon Llama 3 8B Instruct.

Llama 3 8B Instruct. We measure our base model's performance in the zero-shot setting. We experiment with CoT zero-shot prompting (prepending to the question the instruction `Let's think step by step`) but find that just using the raw question as input results in better performance.

Chain-of-Thought (CoT). We extract from the training set six pairs of questions and answers demonstrating maths operations and their use. For comparability, we use the same exemplars as in the implementation of ToolkenGPT and TECTON. Note that Llama 3 8B Instruct achieves 79.6% accuracy (Dubey et al., 2024) on the non-enhanced version of GSM8K when prompted in few-shot CoT fashion. Our experiments show that it performs significantly worse (37.3%) on GSM8K-XL, highlighting the difficulty of the enhanced dataset.

TRICE. TRICE is a two-stage system where instruction tuning is followed by reinforcement learning from environmental feedback (RLEF), leveraging LoRA at both stages. We train TRICE using the same hyperparameters as in the original implementation, with the exception of the batch size in the first phase of training, reduced to 128 with 8 gradient accumulation steps due to the memory limitations of our hardware. Since we train on a single dataset (GSM8K-XL), our implementation corresponds to the setup referred to as TRICE-SPLIT in the original paper. Note that Qiao et al. (2024) implement TRICE on top of Alpaca (Taori et al., 2023), ChatGLM (Zeng et al., 2024a) and Vicuna (Zheng et al., 2023). For comparability with TECTON and the other baselines, here we use Llama 3 as the base model. We thus adjust TRICE's prompt templates and special tokens to be consistent with Llama 3's model card[1].

ToolkenGPT. Like TECTON, ToolkenGPT augments the output matrix of an LLM with additional special tokens, each representing a tool. Only the additional tokens are tuned while the rest of the weights remain frozen.We implement ToolkenGPT with Llama 3 and train it on the same annotated datasets as we train TECTON, using the same sets of tools and the same hyperparameter combinations for direct comparability.

[1]https://www.llama.com/docs/model-cards-and-prompt-formats/meta-llama-3

Test set	Original	XL version
ASDiv	618	360
MAWPS	505	416
SVAMP	299	270

Table 6: Test set sizes of our OOD math reasoning datasets, in their original version and in the XL version enhanced via an automated process.

D Out-of-distribution Datasets

We enhance the test sets of ASDiv, MAWPS and SVAMP and obtain 'XL' versions of each. Our goal is to replace the numbers in each test sample with larger-magnitude ones in an automated manner, yet ensuring that the new numbers in each question correctly map to a new numerical answer. To this end, we use datasets that have been annotated with the golden sequence of operations. Since the original versions of ASDiv and SVAMP do not contain such annotations, we use the versions that are part of the Lila benchmark (Mishra et al., 2022).

Firstly, we extract all the numbers from the chain of operations using regular expressions. Secondly, we search for these numbers in the question text (we use the `num2words` library[2] to include numbers expressed as words) and replace them with random numbers in the interval $[-10^5, 10^5]$. We replace the numbers in both the text and the annotated chain of operations. Note that we replace negative numbers with negative numbers and positive numbers with positive numbers. We also take care to substitute integers with integers and floats with floats. If we cannot match any of the numbers in the operations to the text, we discard that sample. Lastly, we evaluate the new chain of operations with the substituted numbers and store the result as the new golden answer for that data point.

This process results in datasets containing questions with the same structure and wording as the original ones, but with the numbers greatly magnified. This provides a real challenge for contemporary large language models, which can otherwise easily solve small-number arithmetic.

As our automated process discards any data samples that it is unable to process, our final test sets are typically smaller than their original counterparts. Table 6 reports the sizes of the original and modified test sets.

[2]https://github.com/savoirfairelinux/num2words

Prompt 1: TECTON-SCORE meta-reasoning prompt

```
In this task, you must select the correct and most plausible continuation for the answer in the text below. You should use your common sense and logical and mathematical abilities. You should directly answer by choosing the correct option. Output only the letter corresponding to the correct option.

[FIXED EXEMPLARS]

Text to continue:
Question: [QUESTION]
Answer: [PARTIAL ANSWER]

Possible answer continuations:
[OPTIONS]

The correct continuation is:
```

Prompt 2: TECTON-GENERATE meta-reasoning prompt

```
Complete the answers below. In square brackets you will find some hints showing the possible math operations. You may use one of these hints if you think it is correct.

[DYNAMIC EXEMPLARS]

Question: [QUESTION]
Answer: [HINTS] [PARTIAL ANSWER]
```

E Prompts

In the reasoning phase, we prompt TECTON to generate an answer to a math reasoning question step by step, aided by in-context exemplars. Prompts 1 and 2 illustrate how we elicit tool choice in the meta-reasoning phase, for TECTON-SCORE and TECTON-GENERATE respectively. In both cases, we show the model in-context exemplars, followed by the current question and the current partial answer. The latter consists of the lines of text generated during the previous iterations of TECTON, if any. In Prompt 1, the placeholder [FIXED EXEMPLARS] is replaced with six in-context exemplars extracted from the training set. The number of options in each exemplar matches the number of available options in the current sample. In Prompt 2, we have [DYNAMIC EXEMPLARS] that are retrieved to demonstrate the tools currently in the candidate set. Here, the partial answer also includes the tokens generated during the current iteration, up to the sequence position where the first tool is found among the k top probability tokens. The hints are prepended to the partial answer in this setting.

F Efficiency

We compare the efficiency of our method with the fine-tuned baselines—TRICE and ToolkenGPT.

F.1 Comparison with TRICE

TECTON is more efficient than TRICE, as it only requires tuning additional embeddings in the output layer, with minimal backpropagation. In contrast, TRICE performs two rounds of finetuning each time updating LoRA modules throughout the model.

F.2 Comparison with ToolkenGPT

We train TECTON in the same way as the ToolkenGPT baseline, by only updating the additional token embeddings in the output layer. At inference, TECTON requires generating additional text for each line in the problem. Note that for TECTON-SCORE this overhead is negligible, as we generate only one additional token per line.