

Linalg-Zero: Distilling Neurosymbolic Reasoning for Linear Algebra in Small Language Models

Razvan Florian Vasile¹, Matteo Ferrara²

¹Computer Science, University of Bologna, ²Supervisor, University of Bologna



Introduction

We train the **Qwen2.5-3B** base model to solve linear algebra tasks by leveraging **tool calls** instead of free-form arithmetic. This combines symbolic methods (precise computation) with neural control (planning and tool selection) to achieve reliable tool use on a compact model. Training is kept tractable by utilising off-the-shelf cloud GPUs [1] together with **LoRA** for efficient **SFT+GSPO**.

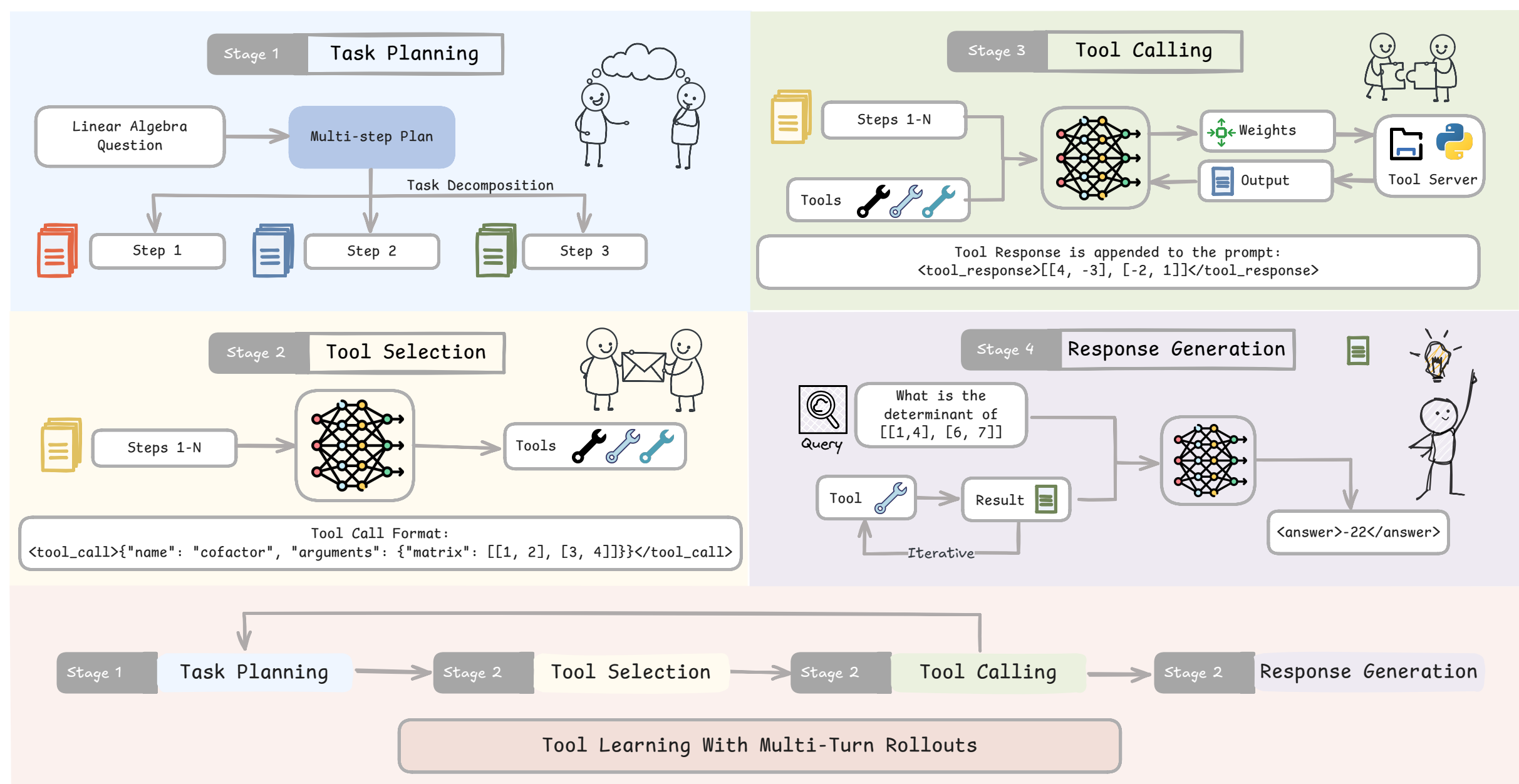


Figure 1. The tool-calling format contract illustrating the end-to-end process.

Problem Statement

- **Identify the Problem:** The literature is often fragmented where few studies demonstrate an end-to-end workflow to training a base model to reliably use tools.
- **Put it into context:** Prior work studies use tools in isolated stages (e.g. tool selection vs tool calling) as a way to differentiate failure points and therefore manage complexity.
- **Find the root cause:** No shared definitions, datasets, or evaluation protocol, so methods diverge and approaches are fragmented [2].
- **Ideal outcome:** Demonstrate an end-to-end workflow, including every protocol stage.
- **Propose a solution:** Build an end-to-end pipeline for linear-algebra tool-use, covering synthetic dataset generation, distillation, and post-training (SFT+GSPO).

Results: Generated Responses with Tool-Use

Example prompt. Step 1: what is the transpose of $A = \begin{bmatrix} 2 & 4 & -2 \\ -3 & 2 & 4 \\ -4 & 2 & -3 \end{bmatrix}$?
Step 2: given matrix B = the result from step 1, find $\det(B)$.

Thought: <think>Okay, let's see. The user has a two-step problem. First, they need the transpose of matrix A, and then they need the determinant of that transposed matrix. So step 1 is to find the transpose of A. Matrix A is a 3x3 matrix. The transpose is formed by swapping rows and columns. The matrix_transpose tool should handle this. I'll call that tool with the given matrix A. Once I get the transposed matrix B, step 2 is to find its determinant. The determinant requires a square matrix, and transposing preserves squareness. The determinant tool can compute that. So after getting B from step 1, I'll use the determinant tool on B to get the final answer.</think>

Tool Call: {"name": "matrix_transpose", "arguments": {"matrix": [[2, 4, -2], [-3, 2, 4], [-4, 2, -3]]}}

Tool Result: [[2, -3, -4], [4, 2, 2], [-2, 4, -3]]

Thought: <think>Okay, I need to find the determinant of the transposed matrix B, which is [[2, -3, -4], [4, 2, 2], [-2, 4, -3]]. The determinant tool can compute this. Since B is a 3x3 square matrix, the determinant function is applicable. I'll call the determinant tool with matrix B as the argument.</think>

Tool Call: {"name": "determinant", "arguments": {"matrix": [[2, -3, -4], [4, 2, 2], [-2, 4, -3]]}}

Tool Result: -132.0

Thought: <think>Okay, so the user asked for two steps. First, transpose matrix A, then find the determinant of that transposed matrix. I used the matrix_transpose tool on A and got matrix B. Then, I applied the determinant tool on B and got -132.0. The determinant of a transpose equals the determinant of the original matrix, so this is consistent. No more steps needed, so I'll present that as the final answer.</think>

Answer: <answer>-132.0</answer>

Table 1. A distilled example solving a linear algebra problem with two tool calls illustrating planning and tool-use.

Dataset Distribution: Monte Carlo value-range search

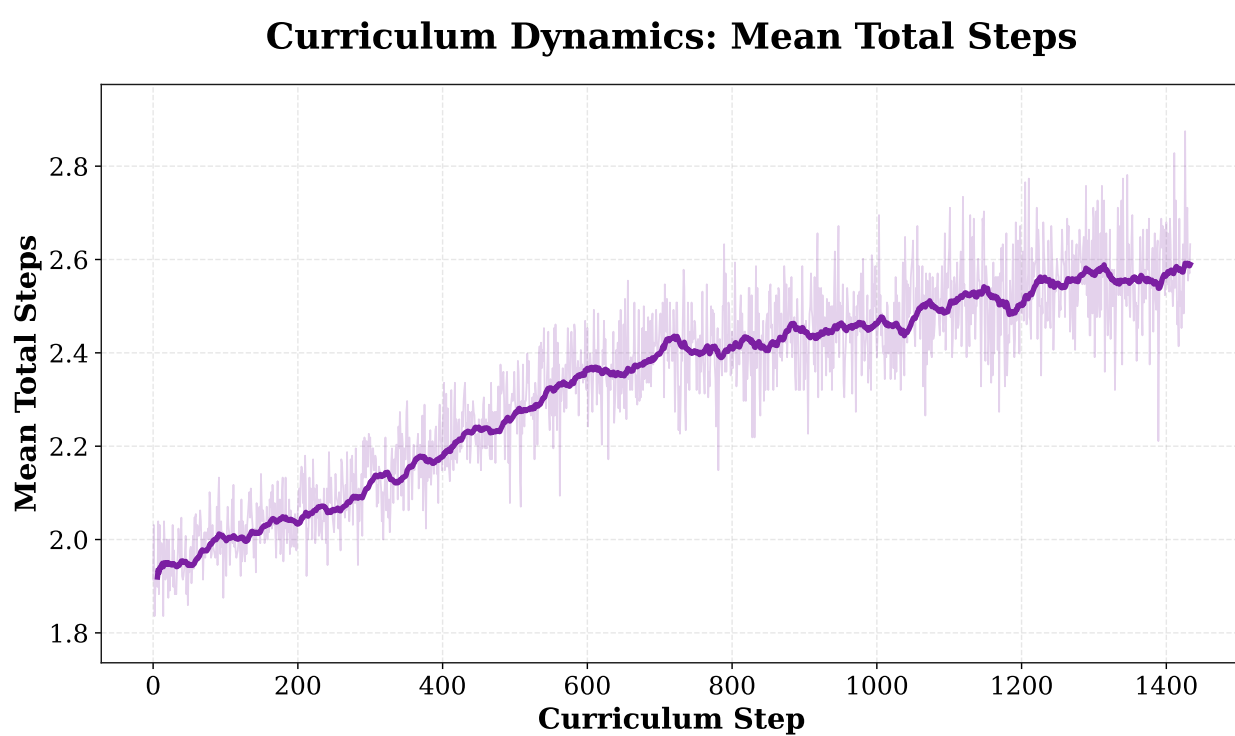
- **Grid Search.** We use grid search over the dataset-generation parameters to keep sampled numeric values within fixed bounds and avoid extreme magnitudes.

Task Type	Steps	Optimized Entropy	Target Range	Observed Min	Observed Max	Samples
Determinant	1	0.9	$[-500, 500]$	-417.00	410.00	8000
Matrix Trace	1	2.1	$[-200, 200]$	-164.00	166.00	8000
Frobenius Norm	1	2.6	$[0, 600]$	18.57	460.79	8000
Matrix Rank	1	2.5	$[1, 3]$	1.00	3.00	8000
Transpose	1	3.2	$[-800, 800]$	-790.00	787.00	8000
Cofactor	1	1.6	$[-800, 800]$	-735.00	779.00	8000
Transpose $\rightarrow$ Det.	2	$[0.7, 0.0]$	$[-400, 400]$	-216.00	224.00	8000
Cofactor $\rightarrow$ Trace	2	$[1.4, 0.0]$	$[-800, 800]$	-469.00	604.00	8000
Cofactor $\rightarrow$ Rank	2	$[1.5, 0.0]$	$[-800, 800]$	-480.00	480.00	8000
Transpose $\rightarrow$ Frob.	2	$[2.8, 0.0]$	$[-800, 800]$	-315.00	768.88	8000
Transp. $\rightarrow$ Cof. $\rightarrow$ Rank	3	$[3.2, 0.0, 0.0]$	$[-800, 800]$	-790.00	791.00	8000
Cof. $\rightarrow$ Transp. $\rightarrow$ Trace	3	$[2.9, 0.0, 0.0]$	$[-800, 800]$	-765.00	774.00	8000
Transp. $\rightarrow$ Cof. $\rightarrow$ Frob.	3	$[2.9, 0.0, 0.0]$	$[-800, 800]$	-395.00	699.58	8000

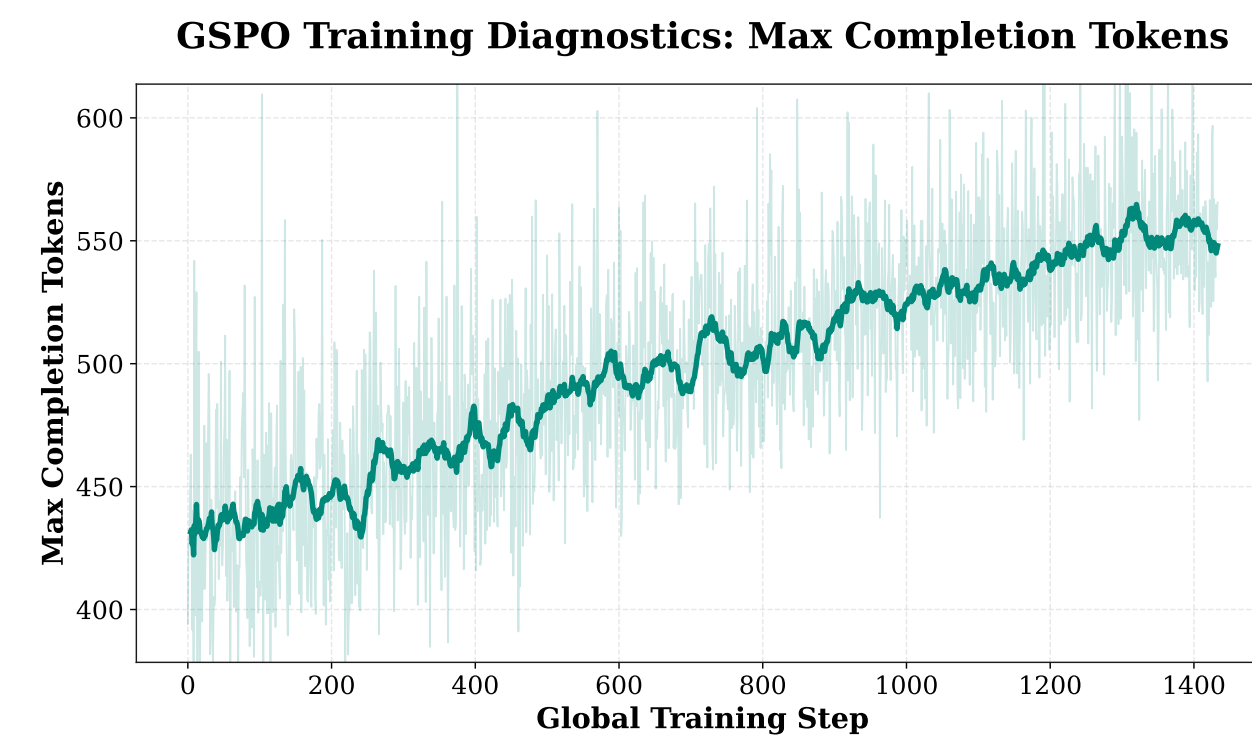
Table 2. Dataset Value Distribution. Calibrated entropy (8,000 samples/task) and observed output ranges.

Why Does Curriculum-Based Learning Help?

- **Curriculum & Budget.** Tasks start tractable and grow harder as performance improves; later phases increase the completion budget which enables longer planning and reasoning.



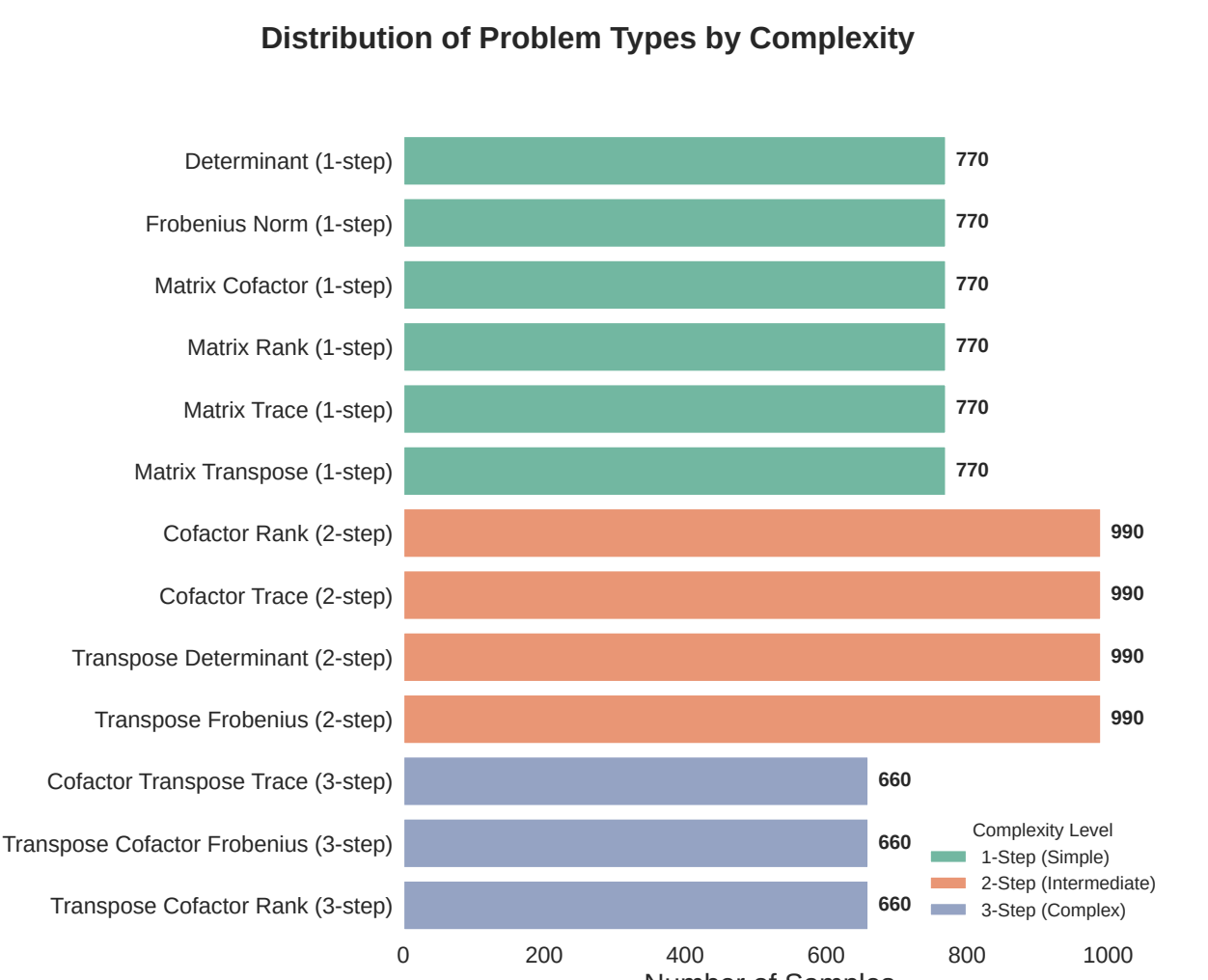
(a) Mean total steps as the curriculum progresses.



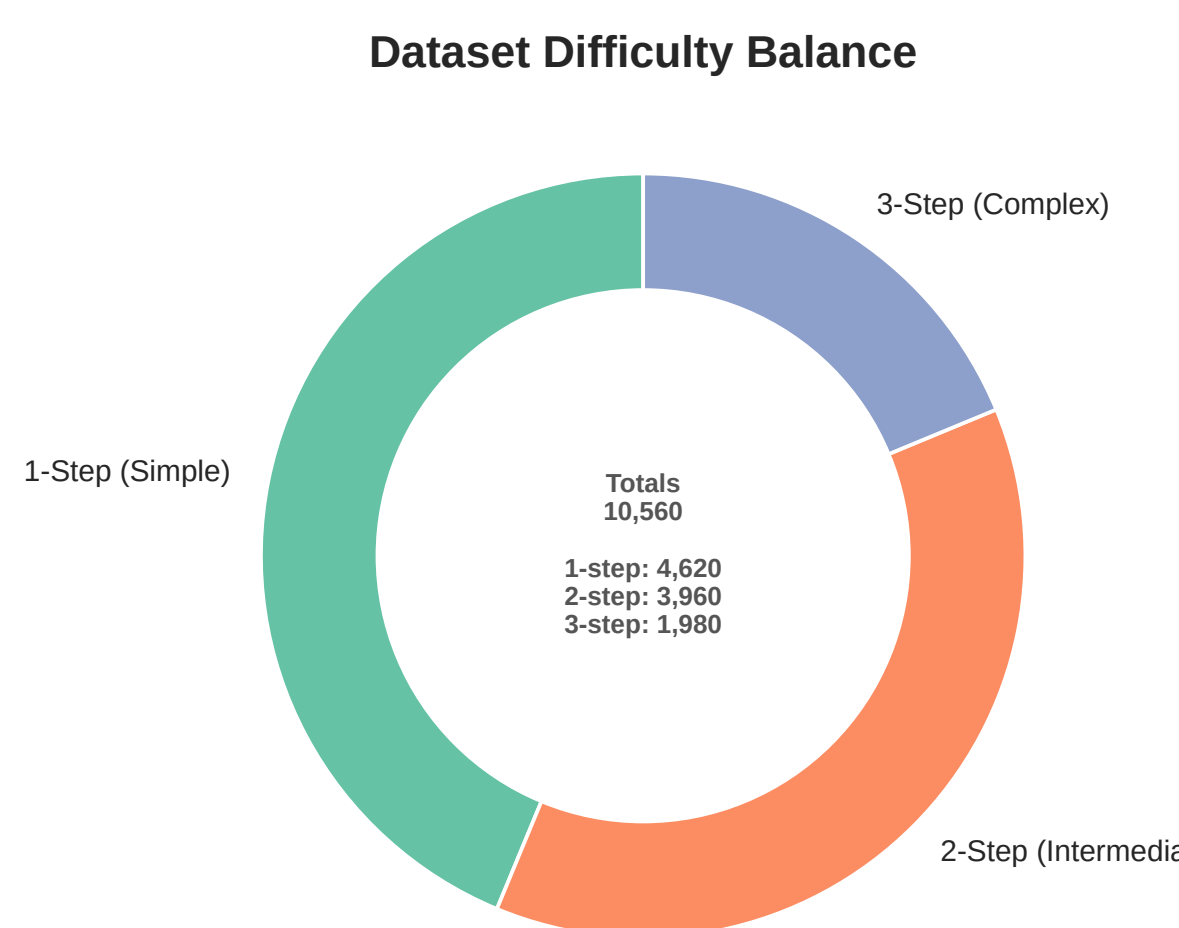
(b) Completion tokens increase as training progresses.

Analysis: Dataset Task Composition

- **Composition.** The dataset has 13 problem types; an arbitrary number of samples per type.

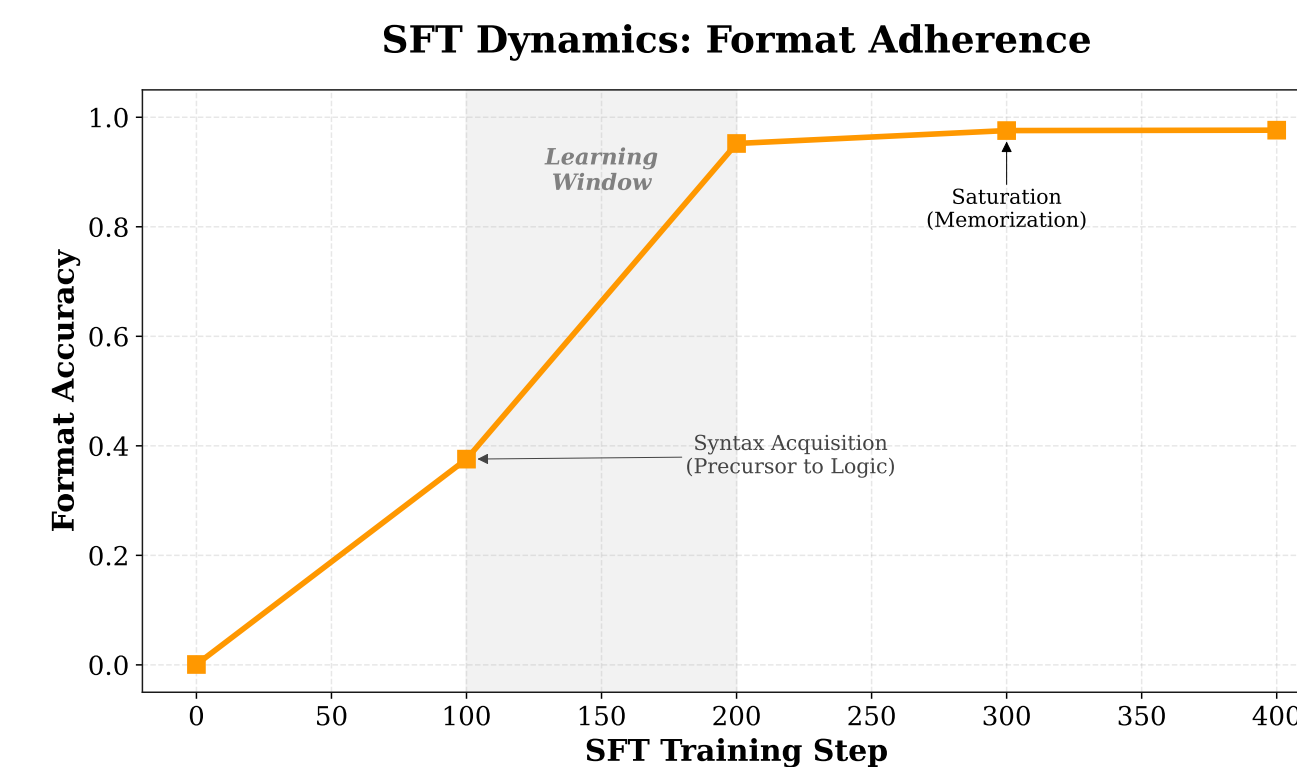


(a) Operation mix by tier.

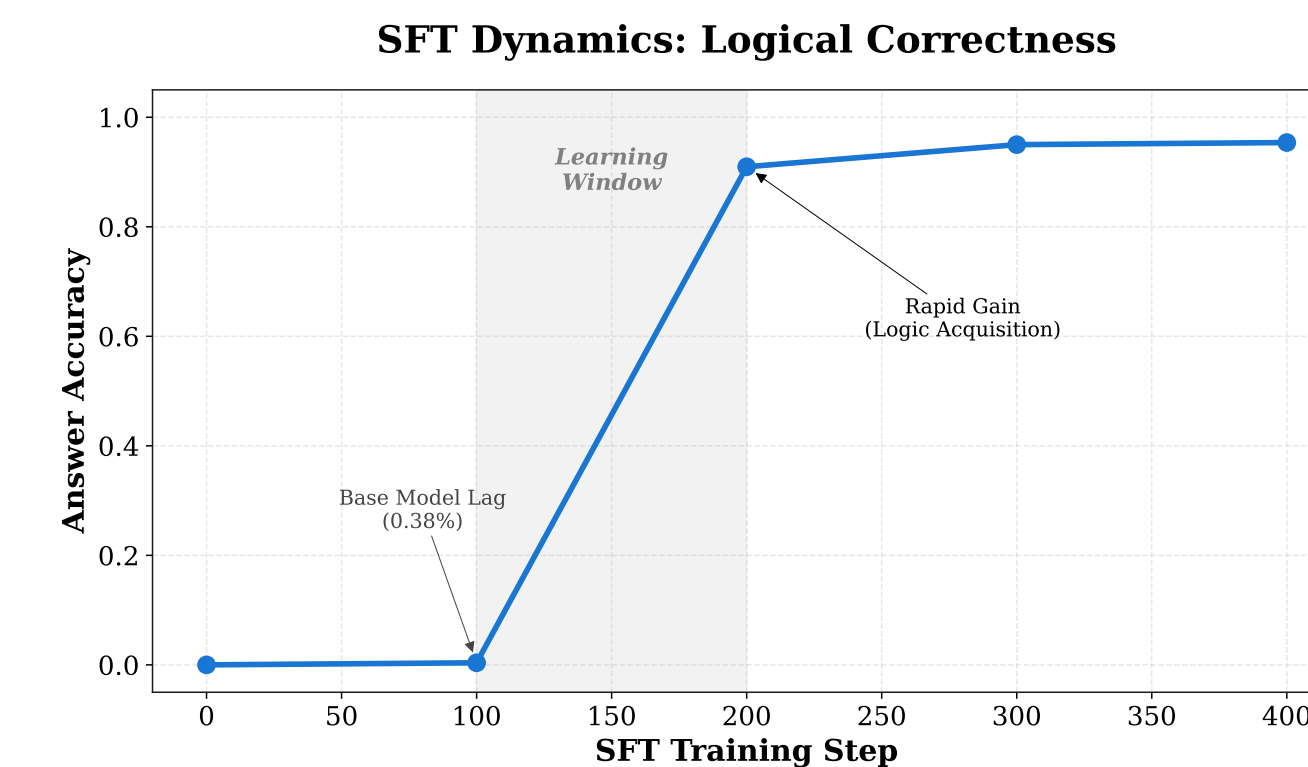


(b) Tier composition (1/2/3-step).

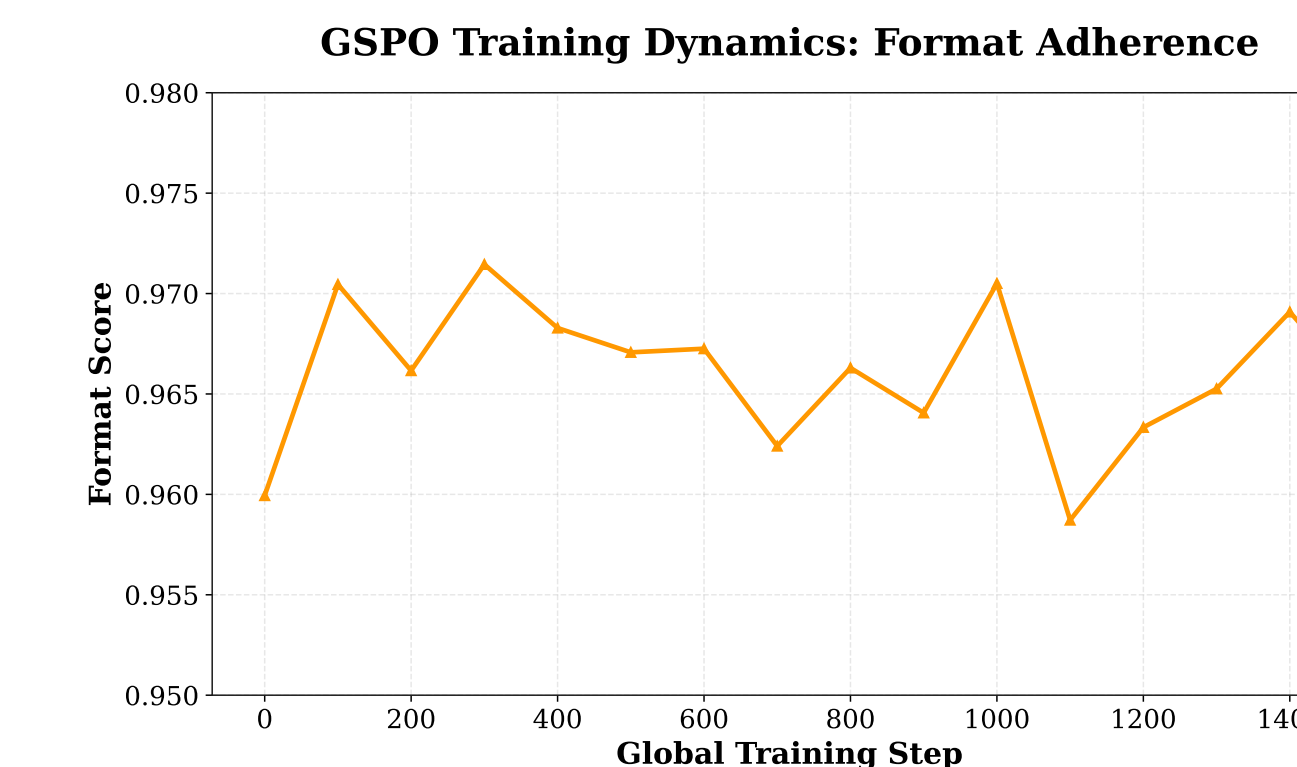
Post-Training: SFT and RL with GSPO



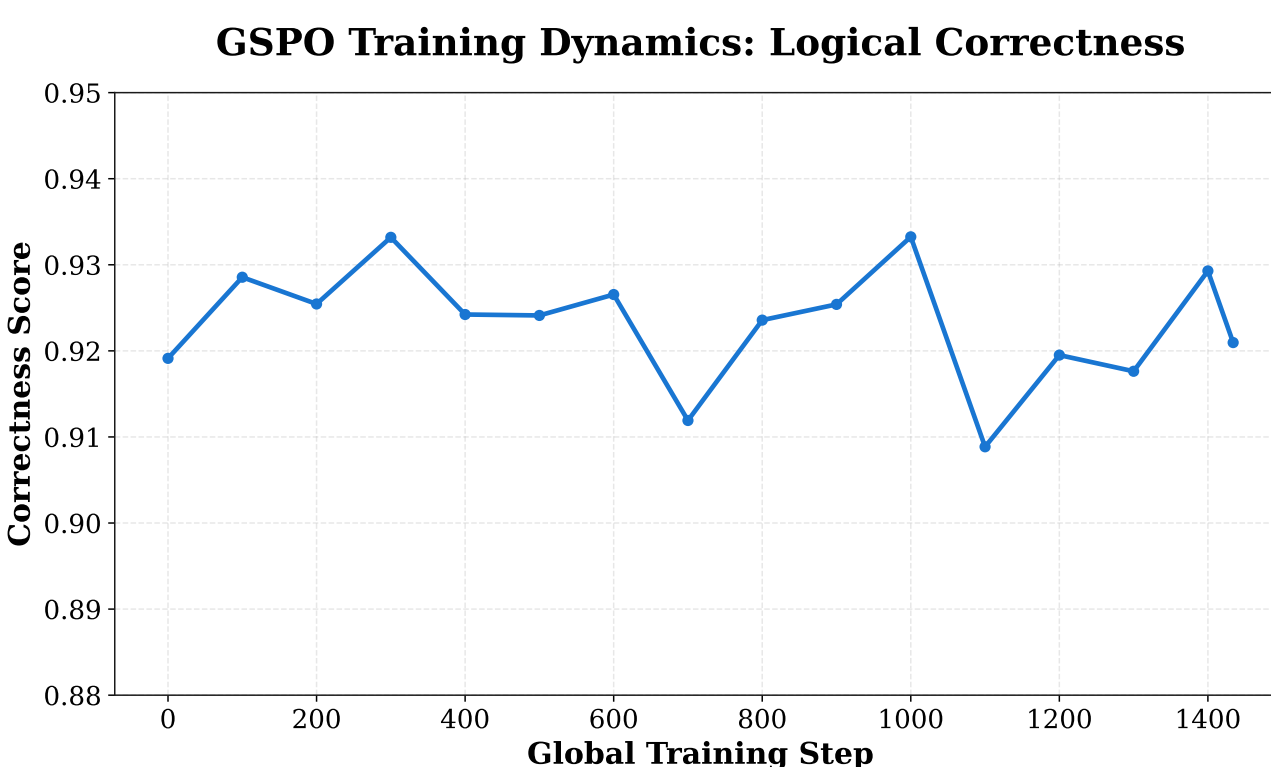
(a) Format Adherence (Syntax)



(b) Logical Correctness (Reasoning)



(c) Format Adherence



(d) Logical Correctness

Figure 4. The top row reports SFT metrics, while the bottom row reports GSPO results; all curves are evaluated with greedy decoding on a held-out 520-samples validation set.

- **Post-Training (SFT).** Starting from a foundation model that does next-token prediction and cannot reliably follow instructions, SFT teaches the format contract for reliable tool-use.
- **Post-Training (GSPO).** GSPO stress-tests the format contract under multi-turn rollouts and a curriculum, marginally improving overall accuracy but hitting a frozen-head format ceiling.

Conclusion

- **Take-Away.** Qwen2.5-3B can be fine-tuned into a reliable tool-use agent that follows instructions and plans multi-step trajectories, using an end-to-end workflow.
- **Test-Set Results (520 ex).** Optimal 90.26%, Correctness 92.63%, Format 96.66%, Tool 100%. Averaged over 3 runs. Metric definitions provided in final report.

Future Work

- **Preference alignment.** Collect chosen vs rejected tool-use traces (multiple rollouts per prompt) and align behavior with DPO as a preference-based alternative to GSPO.
- **Scale and complexity.** Test broader math domains, longer tool chains, and more branching dependencies between intermediate results.

References

- [1] RunPod: Serverless GPU and Pod based GPU. <https://www.runpod.io/>, 2024.
- [2] Zhiruo Wang, Zhoujun Cheng, Hao Zhu, Daniel Fried, and Graham Neubig. What are tools anyway? a survey from the language model perspective, 2024.