FROM REASONING TO AGENTS: BUILDING THE NEXT GENERATION OF ON-DEVICE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in agentic models have enabled a growing range of applications, driving increasing demand for training models specialized for downstream agentic tasks. Existing work, however, either develops proprietary construction recipes for large-scale models or optimizes post-training in isolation, leaving the end-to-end training of small agentic models poorly understood. We show that downstream agentic performance is largely determined by the capabilities and behavioral support established during pre-training; post-training sharpens what the base model can already express, but does not reliably recover what it cannot. We therefore ask which pre-training signals predict post-training gains. We find that *sampling lift*, the improvement obtainable by sampling multiple rollouts over greedy decoding, serves as a good indicator, suggesting that pre-training should preserve diverse behavioral support rather than prematurely concentrating on narrow solution modes. Since pre-training metrics differ substantially in predictive power, we weight them by their measured correlation with post-training outcomes, providing a principled basis for checkpoint selection and data-mixture allocation, and allowing pre-training to be concentrated on the capabilities most predictive of downstream performance. Building on these principles, we study how capabilities equipped in general pre-training are converted into reliable agentic behavior. We find that long-horizon reasoning does not transfer directly to multi-turn tool use, while separating the decoding pattern and injecting trajectory-level supervision enables more effective learning from the student's evolving state. Together, these results form an end-to-end recipe for small agentic models. Trained from scratch with only 1.2T tokens, our 2.5B model matches or surpasses the 7.3B-parameter OLMo-3 model trained with 5$\times$ more tokens, achieving the best overall results on AIME24, GSM8K, HumanEval, and SWE-Bench.

1 INTRODUCTION

Recent progress in agentic AI, where models that plan, call tools, and act autonomously (Yao et al., 2022; Schick et al., 2023; Gao et al., 2023), has been driven almost entirely by frontier-scale models trained by a small number of well-resourced labs (e.g., Claude (Anthropic, 2024) and Kimi K2 (Team et al., 2025)). This concentration means that architectural and training innovations for agentic capability rarely reach the broader community: most teams that need an agentic model for a specific, in-domain task are limited to prompting or lightly fine-tuning an existing frontier or open-weight model, rather than participating in how such capability is built from the ground up. Yet a growing number of production agentic systems are shifting toward small, specialized models (e.g., Octopus v2 (Chen & Li, 2024), TinyAgent (Erdogan et al., 2024), Hammer (Lin et al., 2024)) precisely because narrow, tool-heavy workloads do not require frontier-scale general intelligence, they require reliability, low latency, and low cost on a bounded task distribution. This shift raises a practical question that has received little systematic study: *how should a team with limited compute train a small model, from scratch, to be a strong agent in their domain of interest*?

Existing recipes for small agentic models address this question almost exclusively at the post-training stage: they curate supervised fine-tuning or distillation data over tool-calling trajectories while treating pretraining as a fixed, off-the-shelf ingredient (see, e.g., ToolLLM (Qin et al., 2024), APIGen (Liu et al., 2024b), ToolACE (Liu et al., 2025), APIGen-MT (Prabhakar et al., 2026)). This practice implicitly assumes that any reasonably capable pretrained model can reach a similar agentic endpoint

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.

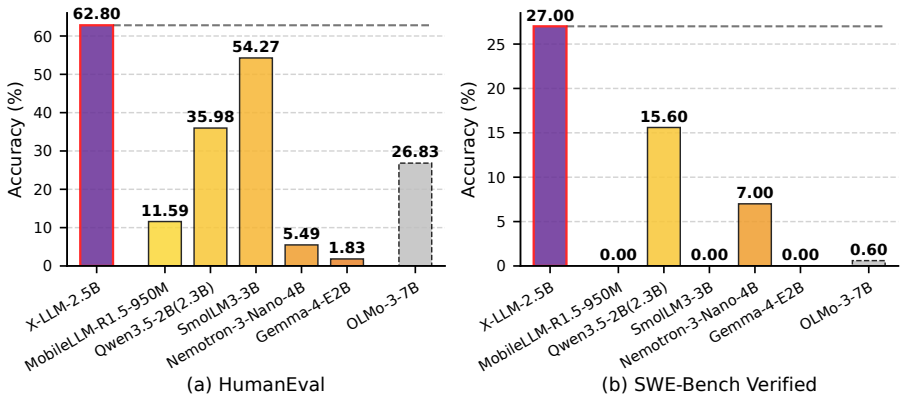

Figure 1: Performance comparison on HumanEval and SWE-bench. Full results are reported in Table 5.

when paired with a sufficiently strong post-training recipe. We show that this assumption does not hold. Through controlled interventions across the training pipeline, we find that downstream agentic trainability is strongly path-dependent: the composition and serialization of pretraining data substantially affect both the data efficiency and the performance frontier of subsequent agentic post-training. Introducing the same structure only during post-training does not close this gap within the data regime we study. More broadly, pretraining must preserve broad support over candidate behaviors rather than prematurely concentrating probability mass on a narrow solution mode. We quantify this support using the sampling lift, $\Delta_k = \text{pass}\,@k - \text{pass}\,@1$ (Brown et al., 2024; Chen et al., 2021). Together, these results suggest that post-training is most effective at eliciting and sharpening capabilities already supported by the pretrained model, rather than creating them from scratch.

We validate this claim end-to-end across both agentic and general reasoning capabilities. We use agentic coding as a primary testbed under a well-established evaluation suite, SWE-Bench, because it requires the full stack of agentic behavior, from multi-turn planning, tool invocation, error recovery to long-horizon state tracking. Using 1.2T pre-training tokens, an order of magnitude below typical frontier-model budgets, we train a small model that matches or exceeds state-of-the-art Qwen3.5-2B (Team, 2026) on agentic coding (SWE-Bench), as well as on long-horizon math benchmarks (AIME, MATH500, GSM8K) and coding benchmarks (HumanEval, MBPP, LiveCodeBench). We additionally evaluate out-of-domain generalization on agentic benchmarks (e.g., Terminal-Bench, Aider Polyglot) and general-purpose tool-calling benchmarks (BFCL), where our model is competitive.

Our contributions are as follows:

- We observe that preserving generation diversity during pretraining is associated with stronger downstream performance, and propose *sampling lift* as a simple diagnostic of capability collapse and pretraining health.
- We find that downstream trainability is concentrated in a small subset of pretrained capabilities. By quantifying their predictive value, we derive a simple weighting signal that turns inexpensive base-model evaluations into actionable guidance for checkpoint selection and data allocation.
- Contrary to the common belief that more reasoning benefits agentic tasks, we find long-horizon reasoning can interfere with agentic multi-turn tool use, reflecting fundamentally different reasoning patterns. Separating high- and low-effort reasoning allows the two behaviors to coexist, improving agentic performance without sacrificing long-horizon reasoning.
- While on-policy distillation (Agarwal et al., 2024) benefits multi-turn agents, token-level on-policy supervision can become unreliable after early tool-selection errors, whereas supervising the teacher's remaining trajectory from the student's current state provides substantially stronger learning signals.
- With only 1.2T pretraining tokens, roughly 30× fewer than Qwen3-1.7B (Yang et al., 2025), our 2.5B model reaches 27.0% on SWE-Bench Verified, substantially outperforming Qwen3.5-2B (15.6%) (Team, 2026), as shown in Figure 5 and matching or exceeding larger models like OLMo-3 7B on software engineering while also achieving strong long-horizon reasoning.

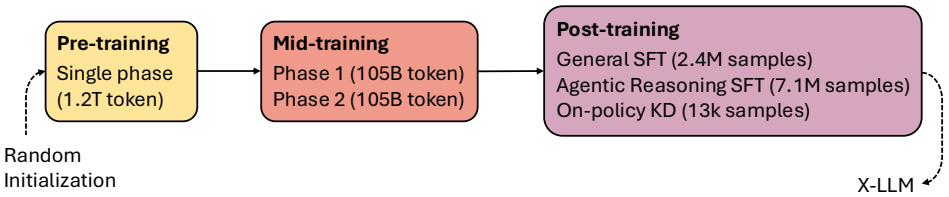

Figure 2: Overall training pipeline of X-LLM.

2 Related Work

Small language models have improved through advances in architecture design, data curation, knowledge distillation, and multi-stage training. MobileLLM established strong sub-billion-parameter architectures and demonstrated that compact models can support both conversational and API-calling workloads (Liu et al., 2024a). Subsequent models, including SmolLM3, Qwen3 have improved small-model capabilities through curated and synthetic data, extended pretraining, distillation, and increasingly sophisticated post-training pipelines (Allal et al., 2025; Yang et al., 2025). However, these approaches often rely on large training budgets: SmolLM3 and Qwen3 use approximately 11.2T, 36T pretraining tokens, respectively.

MobileLLM-R1 is particularly close to our setting, providing an open pretraining-to-post-training recipe for a sub-billion-parameter model using 4T pre-training tokens, but focuses on mathematical and coding reasoning rather than agentic tool use (Zhao et al., 2026). Qwen3.5-2B/4B provides a strong small-model agentic baseline, but does not release a fully reproducible end-to-end recipe, including key details of the training data composition and token budgets (Team, 2026). OLMo3 is the closest precedent in terms of openness: it releases the complete model development flow and incorporates function calling and multi-step tool-use trajectories, including interactions with real MCP environments (Olmo et al., 2025). Its smallest model, however, is 7B parameters, with approximately 5.9T pretraining tokens. In comparison, 2B-scale models are better suited to on-device deployment under tight memory, compute, and latency constraints (Yan & Ding, 2025). Yet, open end-to-end recipes at this scale remain relatively unexplored.

We address this gap by developing an on-device agentic model from scratch, with a complete training pipeline spanning pretraining, mid-training, and post-training. This end-to-end setup allows us to study how decisions across training stages jointly shape reasoning and multi-turn tool use. With only 1.2T pretraining tokens, we show that targeted, high-quality data can rapidly equip a small model with reliable reasoning and agentic behavior.

3 Pre-training and Mid-training Dynamics

3.1 Capability-Aware Data Design

Our data design is guided by a simple objective: to develop a small model that supports both *agentic reasoning* and *long-horizon deliberative reasoning* under limited training-token budget. We therefore optimize the data mixture against both capabilities rather than treating them independently. Following the mixing procedure of MobileLLM-R1 (Zhao et al., 2026), we calculate the influence score of each candidate dataset on the target capabilities, including tool calling, instruction following, mathematics, and coding, and use these influence scores to determine the mixing ratios. The resulting pretraining mixture is shown in Figure 3. It consists primarily of publicly released datasets, including web text from filtered Common Crawl derivatives (Li et al., 2024; Penedo et al., 2024; Su et al., 2025), code from permissively licensed repositories (Lozhkov et al., 2024), mathematics from web-mined and curated math corpora (Paster et al., 2024; Allal et al., 2025), and additional multi-source collections (Soldaini et al., 2024). Overall, more than half (i.e. 59.4%) of the mixture is generic web data and remaining comes from capability-targeted sources spanning knowledge, commonsense, reasoning, science, mathematics, and code. We apply the same influence-based mixing procedure when constructing the mid-training data mixture, allowing both training stages to explicitly account for the two target capability families.

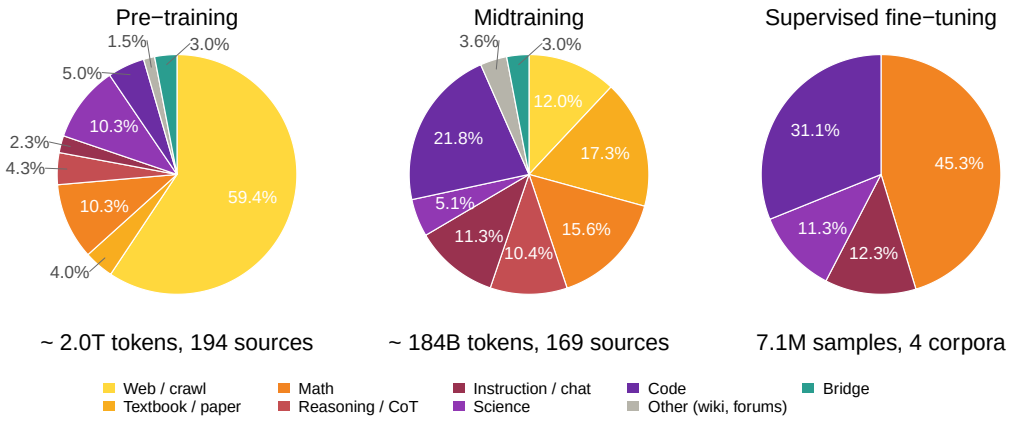

Figure 3: Composition of the training data across a shared capability taxonomy. The pretraining mixture provides broad capability coverage, augmented with synthetic *bridge* data to connect complementary capability domains. Mid-training reallocates additional token budget toward mathematics, coding, and reasoning to progressively concentrate capability-specific learning. Supervised fine-tuning denotes the Agentic Reasoning SFT stage, which jointly integrates agentic, mathematical, and coding data. Detailed post-training dataset composition is provided in Section A.

Synthetic cross-domain *bridge* data. While capability-aware mixing provides a principled way to allocate a limited data budget, different capability domains do not combine additively. Adding data for one capability can alter performance on others (Xie et al., 2023; Aryabumi et al., 2025; Dong et al., 2024), leading to trade-offs as the composition and amount of training data change. This tension is particularly important for our setting, where the model must simultaneously support agentic multi-turn tool use and long-horizon single-turn reasoning. Inspired by this observation, we propose to incorporate the bridging mechanism directly into pretraining through synthetic cross-domain *bridge* data. *Bridge* data connect otherwise competing domains by combining their content in a natural intermediate form. For instance, natural-language explanations of code, mathematical problems presented as factual or scientific text, and reasoning over encyclopedic content. We generate these documents with Llama-3 8B (Grattafiori et al., 2024) and allocate 3% of the training mixture to *bridge* data, as shown in pie charts in Figure 3. Rather than introducing a new capability, cross-domain *bridge* data reduces the distributional gap between existing capabilities, helping the model acquire them jointly under a limited token budget, and were shown to yield an average of 3-5% improvements across pretraining zero-shot benchmarks.

3.2 From Pre-training Signals to Post-training Potential

As large language models are developed through a sequence of training stages, pre-training lays the foundation for subsequent post-training stages. Yet, the quality of a pre-trained model is ultimately judged by its downstream post-training performance, while most signals available during pre-training only measure the model in its current state. This creates a fundamental gap: how can we tell, before investing in expensive post-training, whether a pre-trained model is on track to become a strong final model? In practice, this makes pre-training decisions difficult to close the loop: changes in data, training strategy, or token allocation are typically evaluated only after completing the entire multi-stage pipeline. We therefore ask a simple but fundamental question: *can signals observed during pre-training predict how well the resulting model will perform after post-training?* We systematically study this question and identify pre-training signals that provide early evidence of downstream post-training performance, enabling pre-training to be evaluated not only by its immediate gains, but also by the quality of the model it enables at the end of the training pipeline.

Sampling lift as a signal of retained capability diversity. A pretrained model can improve its most likely solution without expanding the set of solutions it can generate. To measure this distinction, we track

$$\Delta_{k,t} = \text{pass}\,@k_t - \text{pass}\,@1_t, \tag{1}$$

where $\Delta_{k,t}$ measures the additional success obtained by sampling k solutions from checkpoint t.

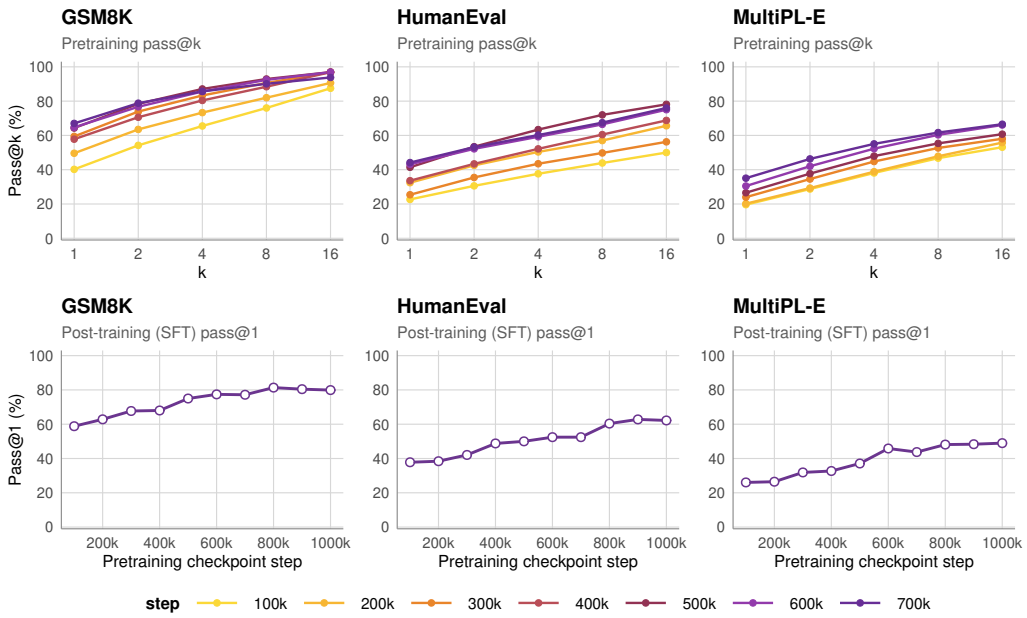

Figure 4: *Sampling lift* provides an early signal of downstream post-training potential. Later in pretraining, pass @1 improves while pass @16 saturates or declines, shrinking the sampling lift $\Delta_{16} = \text{pass @16} - \text{pass @1}$ and indicating reduced exploration. We find that this coincides with diminishing post-training gains: checkpoints with a plateauing Δ_{16} yield little additional improvement after post-training.

Early in pretraining, pass @1 and pass @k improve together, indicating simultaneous growth in both the most likely solution and the broader solution space. Later in training, however, pass @1 can continue to improve while pass @k saturates, causing $\Delta_{k,t}$ to shrink. This pattern indicates increasing concentration of probability mass on a narrower set of solutions: the checkpoint becomes better at reproducing a dominant trajectory, but retains less exploratory diversity. Importantly, the emergence of this regime is also associated with diminishing gains in post-training accuracy (Figure 4). This observation suggests that base-model quality cannot be characterized by a single point estimate such as pass @1. The sampling lift provides a complementary view of *how much useful capability remains distributed across alternative solutions*, which can affect the amount of capability that post-training is able to unlock.

Zero-shot scores as predictors of downstream post-training performance. Sampling lift provides a useful signal of how much capability diversity a checkpoint retains. Beyond this global measure, we ask whether the individual zero-shot capabilities routinely measured during pretraining can predict performance on downstream post-training objectives. This distinction is important because we find that zero-shot benchmarks can differ substantially in their predictive power: average zero-shot accuracy may improve without indicating which capabilities are actually relevant to subsequent post-training. We therefore measure how predictive each capability is of downstream post-training accuracy. This allows us to identify the pretrained capabilities that provide the strongest signals for the eventual post-training objective.

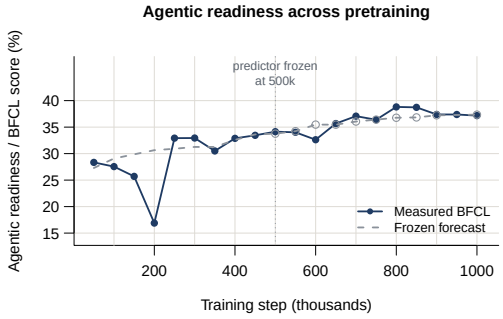

Figure 5: Zero-shot capability composition predicts post-training performance. The predictor is fitted exclusively on early pretrained checkpoints (<500k steps) using their zero-shot benchmark scores and sampling lift, then frozen and applied to predict post-training accuracy at later checkpoints (>500k steps).

Let $\mathbf{x}_t \in \mathbb{R}^{20}$ denote the zero-shot scores across 20 benchmarks spanning six capability domains: knowledge (MMLU, MMLU-Redux, MMLU-Pro, BoolQ), science (ARC-Challenge, ARC-

Easy, GPQA, SuperGPQA), commonsense (HellaSwag, PIQA, SIQA, WinoGrande), mathematics (GSM8K, MATH), reasoning (BBH, BBEH), and code (HumanEval+, MBPP, CRUXEval, MultiPL-E). For each checkpoint, sampled at 50k-step intervals, we apply the same fixed post-training recipe and evaluate BFCL performance, providing a direct measure of the checkpoint's downstream tool-use capability. We then fit a predictor of the form

$$\widehat{B}_t = f(\mathbf{x}_t, \Delta_{k,t}), \tag{2}$$

where B_t denotes the post-training BFCL score. The predictor is trained only on early pretrained checkpoints and evaluated on later checkpoints, using no post-training information. We fit a multivariate predictor to benchmark trajectories from checkpoints before 500k iterations, estimating each benchmark's predictive contribution and using the resulting coefficients to construct a capability-weighted predictor of downstream performance.

Figure 5 shows that the resulting prediction closely tracks the measured post-training performance on later checkpoints (>500k steps) that are never used for fitting. In contrast, the uniform aggregate can continue to improve even when the downstream BFCL score changes little.

Which pretrained capabilities matter? The learned predictor reveals a strong alignment between pretraining capabilities and downstream tool-use readiness. We therefore examine the normalized contribution of each benchmark using an interpretation-only sensitivity analysis over the 20 zero-shot scores and Δ_{16} (Figure 6). The resulting coefficients can be interpreted as predictive associations.

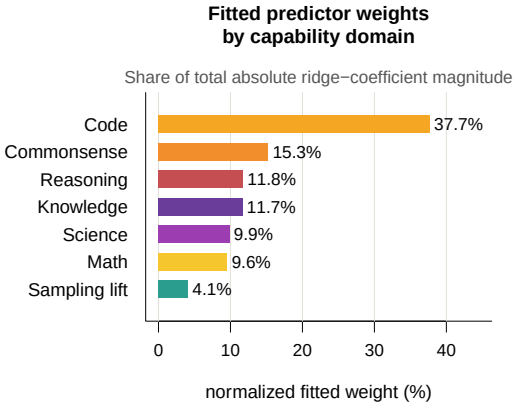

Figure 6: Fitted predictor weights across capability domains, quantified as normalized ridge coefficients for predicting post-training performance. Higher coefficient magnitudes reflect stronger predictive association.

This analysis yields two complementary insights. First, the predictive value of a benchmark depends on its alignment with the downstream post-training objective. Code provides the strongest predictive signal (37.7%), plausibly because its compositional, constraint-driven, and executable structure closely aligns with tool-use behavior. Other capabilities provide complementary but more indirect signals, while highly correlated benchmarks provide limited additional signal beyond one another. Second, sampling lift contributes information that is not captured by zero-shot accuracy alone: it measures the diversity of alternative solutions available to the base model rather than only the performance of its most likely output. Together, these signals characterize both *what the model already knows* and *how much useful capability remains distributed across alternative behaviors*.

3.3 Guiding Pre-training and Mid-training with Downstream Signals

Overall, these results turn inexpensive base-model measurements into actionable signals for pretraining design. Sampling lift reveals whether further training is preserving a sufficiently diverse capability space, while capability-specific zero-shot scores identify which pretrained competencies are predictive of the downstream objective. These signals can therefore guide training process by rather than preparing data mixtures according to averaged zero-shot accuracy alone, we can favor configurations that maximize the capabilities most predictive of post-training performance while preserving sufficient solution diversity.

This view also motivates our mid-training strategy. We treat mid-training as a controlled concentration of the broad pretraining distribution toward capabilities with high downstream predictive value. In particular, allocating additional training tokens to code, mathematics, instruction following, and tool use strengthens capabilities that can be effectively leveraged during post-training. The resulting pretraining and mid-training recipe therefore aims to retain broad general capabilities while selectively allocating additional training budget to competencies that support both *agentic* and *long-horizon reasoning* post-training. Together, these findings provide a principled way to design pretraining

Table 1: Performance comparison under different training data mixtures.

Training data	**MATH500**	**AIME'25**	**LCB**	**SWE-Bench**
Long-horizon reasoning data only	90.8%	38.3%	31.4%	0.0%
Agentic multi-turn data only	30.0%	2.6%	12.0%	25.4%
Joint reasoning + agentic training	91.2%	37.1%	29.3%	27.0%

Table 2: Output length across thinking-effort levels. For the agentic task (SWE-bench), output length is measured per turn.

	Long-horizon Single-turn Reasoning			**Agentic Multi-turn Tool Use**
Metric	MATH500	AIME'25	LCB	SWE-Bench
Output length (p50)	1724	13157	16718	210
Output length (p90)	8331	21527	29999	629

and mid-training not simply to produce a stronger base model, but to construct a *post-training-ready substrate* whose capabilities are broad, balanced, and well aligned with the behaviors that post-training can effectively unlock.

4 POST-TRAINING

4.1 TASK-CONDITIONED REASONING EFFORT

The development of chain-of-thought (CoT) reasoning has established inference-time computation as an important axis for improving language model capabilities: explicitly eliciting intermediate reasoning and allocating additional computation can substantially improve performance on complex mathematical and coding problems (Wei et al., 2022; Lightman et al., 2024; Snell et al., 2025). Recent reasoning models have extended this idea by supporting different levels of reasoning effort, from direct responses to extended deliberation, as exemplified by Qwen3's thinking and non-thinking modes and the configurable reasoning effort of frontier models (Yang et al., 2025; Anthropic, 2025). This shift raises a fundamental training question: *how should a model be trained to support different reasoning behaviors without causing interference between them?*

We argue that different tasks require fundamentally different reasoning patterns, not simply a decoding-time length knob. Long-horizon mathematical and coding tasks benefit from extended internal deliberation before producing an answer, whereas agentic tool use requires a distinct pattern centered on retrieving relevant information, tracking state across turns, and selecting the next action. Thus, agentic reasoning is not simply a different amount of reasoning, but a different form of reasoning that must be learned alongside long-horizon deliberation. To explicitly separate these behaviors during training, we associate each training example with an effort token $e \in \{\text{HIGH}, \text{LOW}\}$, assigning long-horizon reasoning trajectories to high effort and agentic tool-use trajectories to low effort. This conditioning provides a unified way to jointly train heterogeneous reasoning behaviors while explicitly indicating the desired reasoning pattern.

Table 1 reveals a complementary relationship between the two training objectives: joint training improves both long-horizon reasoning and agentic performance over training on either data type alone. Rather than competing for a shared capacity, the two forms of supervision can reinforce the model's overall capability when their target behaviors are explicitly differentiated.

We further examine whether the model actually learns the intended distinction between the two modes. As shown in Table 2, the model produces substantially longer reasoning traces for long-horizon single-turn tasks, while generating shorter reasoning before interacting with the environment in multi-turn agentic tasks. Importantly, these differences emerge under the corresponding effort conditions while maintaining strong task performance, indicating that the model learns distinct task-dependent reasoning patterns rather than simply applying a uniform reasoning-length strategy.

4.2 ON-POLICY DISTILLATION FOR AGENTIC BEHAVIOR

Supervised fine-tuning learns from expert trajectories, whereas deployment exposes the model to its own prefixes, where an early tool-selection error can induce an unsupported state and compound throughout the episode. On-policy knowledge distillation (OPD) (Agarwal et al., 2024) addresses

Table 3: BFCL V4 multi-turn accuracy comparison on on-policy knowledge distillation (OPD) design choices.

Model / recipe	**MT Avg.**	**Δ**	**Base**	**Miss Func.**	**Miss Param.**	**Long Ctx.**
Full SFT initialization†	14.25	0.00	17.00	16.00	16.50	7.50
Exp. 1: Naive OPD-KD	19.00	+4.75	33.50	11.50	19.50	11.50
Exp. 2: One-Turn OPD	24.88	+10.63	38.50	**19.50**	25.50	16.00
Exp. 3: Till-End OPD	**30.75**	**+16.50**	**50.50**	17.00	**34.50**	**21.00**
Exp. 4: Verified Till-End OPD	16.50	+2.25	29.00	6.00	19.00	12.00

Table 4: On-policy KD Corpus statistics. Candidate episodes counts sampled rollout episodes before row construction and filtering. Training rows counts serialized examples consumed by the optimizer after expansion and filtering. Supervised turns counts assistant messages whose token spans carry nonzero loss. Input tokens counts all serialized tokens processed by the optimizer, including masked context, whereas loss tokens counts only tokens with nonzero loss. Loss/input is their ratio.

Recipe	Candidate eps.	Training rows	Sup. turns	Input tok.	Loss tok.	Loss/input
Exp. 1: Naive OPD-KD	12,180	12,180	67,265	70.58M	35.04M	49.65%
Exp. 2: One-Turn OPD	12,180	59,377	59,377	300.91M	19.60M	6.51%
Exp. 3: Till-End OPD	12,180	10,967	70,431	53.08M	14.77M	27.83%
Exp. 4: Verified Till-End	12,180	4,082	26,170	15.60M	3.91M	25.08%

this mismatch by supervising states reached by the student itself, but tool use introduces a causal asymmetry: once an action produces an invalid observation, subsequent token-level guidance cannot recover the counterfactual trajectory. This motivates controlling *where* the teacher intervenes and *how long* it remains responsible for the correction.

We study a progression from token-level guidance to trajectory-level correction. *Naive OPD* (Agarwal et al., 2024) applies dense reverse-KL supervision to every student-sampled token, but continues training on potentially corrupted prefixes. *One-Turn OPD* instead uses a teacher-generated next-turn response as a counterfactual target at each student-visited state, without executing the teacher response or backpropagating loss through the preceding student action (Sun et al., 2026; Li et al., 2026a;b). *Till-End OPD* extends this correction by letting the teacher complete the trajectory from a sampled student state and supervising the entire suffix, providing recovery beyond a single turn (Lauffer et al., 2025). Finally, *Verified Till-End* retains only structurally complete and outcome-correct teacher suffixes, trading coverage for higher-confidence trajectory-level supervision.

We conduct controlled experiments to isolate the effect of intervention granularity, using the same 12,180 candidate episodes and rollout horizon. *One-Turn OPD* improves BFCL multi-turn accuracy from 19.00% with *Naive OPD* to 24.88%, indicating that localized counterfactual correction is more effective than dense token-level supervision on student-corrupted prefixes. *Till-End* further reaches 30.75% accuracy by extending supervision through the observation-conditioned recovery trajectory. Consistent with this, analysis of raw evaluation traces shows that tool-execution errors drop from 7.63% to 0.99% of tool-executing steps, while failed-action repetition drops from 74% to 8%. This improvement also comes with higher training efficiency: the *Loss/Input ratio* is only 6.51% for *One-Turn*, compared with 27.83% for *Till-End*. *Verified Till-End*, however, falls to 16.50% in multi-turn accuracy, mainly because only 29.9% of switch-reached suffixes pass verification, filtering out difficult recovery states and reducing coverage. Accordingly, it retains only 4,082 episodes from the same 12,180 candidates, reducing loss tokens from 14.77M to 3.91M.

5 Main Results

Following the training pipeline in Figure 2, we obtain our final 2.5B-parameter model, `X-LLM`. `X-LLM` uses 1.2T pre-training tokens and fewer than 2T tokens in total, yet supports both long-horizon reasoning and agentic tool-calling. This co-designed training recipe provides a token-efficient path to jointly acquiring these complementary capabilities in a compact model. We next evaluate `X-LLM` against state-of-the-art models across reasoning, coding, and agentic benchmarks, with results summarized in Table 5. Detailed model architecture and training hyperparameter settings are provided in Section A.

The results in Table 5 reveal a pronounced shift from *reasoning competence* to *effective agency*: `X-LLM` achieves 27.0 on SWE-Bench Verified, substantially exceeding the strongest prior result of

Table 5: Performance comparison of X−LLM with six state-of-the-art open-source language models ranging from sub-billion to 7B parameters: MobileLLM-R1.5 (Zhao et al., 2026), Qwen3.5-2B (Team, 2026), SmolLM3 (Bakouch et al., 2025), Nemotron 3 Nano 4B (Blakeman et al., 2025), Gemma 4 E2B (Team et al., 2026), and OLMo 3 (Olmo et al., 2025). We report benchmark accuracy in percent (%). Bold denotes results within 1.0 percentage point of the best model with fewer than 3.5B total parameters, while italics denote the best result overall.

		MobileLLM-R1.5 950M	Qwen3.5 2B	SmolLM3 3B	Nemotron 3 Nano 4B	Gemma 4 E2B	OLMo 3 7B	X−LLM
Category	Benchmark	0.95B	2.27B	3.08B	3.97B	2.00B / 5.12B	7.30B	2.50B
Math	AIME24 @16	39.8	13.3	46.7	35.0	32.5	44.2	***51.5***
	AIME25 @16	31.0	16.5	**37.7**	*78.5*	23.8	32.5	**37.1**
	MATH500	86.6	67.8	**91.8**	*95.4*	50.8	87.2	**91.2**
	GSM8K	83.7	76.2	88.0	87.9	80.6	85.7	***93.4***
Knowledge	MMLU	37.4	**65.9**	63.2	65.8	58.1	*69.1*	55.3
	GPQA	29.3	**51.5**	41.4	*61.1*	44.4	40.4	38.9
Coding	HumanEval	11.6	36.0	54.3	5.5	1.8	26.8	***62.8***
	MBPP	5.6	23.2	***50.0***	10.8	22.0	29.8	**49.6**
	LCB	**29.1**	12.6	17.3	43.8	*44.0*	21.4	**29.3**
Agentic	BFCL	16.7	**48.9**	35.0	*54.6*	44.2	22.4	36.7
	SWE-Bench Verified	0.0	15.6	0.0	7.0	0.0	0.6	***27.0***
OOD	Terminal-Bench	0.0	**4.1**	0.4	2.6	1.9	2.2	***4.9***
	Aider Polyglot	2.7	**3.7**	2.5	*5.0*	3.7	2.1	**3.6**
	BFCL-Parity	0.0	30.1	**70.2**	68.0	*84.0*	69.4	64.8

15.6 despite using a comparable 2B-scale parameter budget. This gap is particularly notable because SWE-Bench requires a model to coordinate code understanding, tool calls, iterative feedback, and long-horizon execution rather than solve a single isolated problem. The gains therefore suggest that compact models can acquire meaningful agentic behavior when tool use is treated as a first-class training objective rather than an inference-time scaffold.

Importantly, this agentic specialization does not come at the expense of long-horizon reasoning. X−LLM reaches the best overall results on AIME24 at 51.5 and GSM8K at 93.4, while matching the strongest sub-3.5B result on MATH500 and LCB. It also achieves the best HumanEval score at 62.8, indicating that the same training recipe transfers beyond mathematical reasoning to algorithmic code generation. Notably, X−LLM exceeds the 7.3B OLMo 3 model across all four mathematical benchmarks while using nearly $3\times$ fewer parameters.

The resulting capability profile is achieved with remarkably little data: X−LLM uses $9.3\times$ fewer pretraining tokens than SmolLM3, yet reaches comparable or stronger performance across reasoning and coding and a substantially higher level of software-engineering agency. The trend also extends beyond training targets: X−LLM obtains the best overall Terminal-Bench result at 4.9 and remains competitive on Aider Polyglot and BFCL-Parity. Taken together, these results show that a compact model trained under a jointly designed token budget can move beyond strong standalone reasoning toward effective long-horizon action, with SWE-Bench providing the clearest evidence of this transition.

6 Conclusion

We show that building capable small agentic models requires more than effective post-training: agentic trainability is strongly shaped by the pretraining trajectory. Our findings highlight three key principles: pretraining should preserve diverse behavioral support, reasoning for agentic interaction differs fundamentally from conventional long-horizon reasoning, and agentic learning should account for the student's evolving trajectory rather than rely solely on local supervision. Together, these results suggest that pretraining and post-training should be co-designed as a unified process. With only 1.2T pretraining tokens, our 2.5B model achieves strong performance across reasoning, tool use, and software engineering, demonstrating that capable agentic models can be developed without frontier-scale training budgets. We release the complete training recipe and checkpoints to facilitate further research on efficient and reproducible agentic model training.

AI USE STATEMENT

We used AI-assisted tools during the preparation of this manuscript and the development and execution of experimental code. AI tools were used primarily for language polishing, code implementation, experiment management, and workflow automation. All AI-assisted outputs were subject to human review and verification. For experimental code, the authors specified the experimental settings, requirements, and evaluation procedures, and carefully reviewed the generated code and experimental results. The authors take full responsibility for the correctness and reproducibility of the code and experiments, and the code used in this work will be released to facilitate reproducibility.

ETHICS STATEMENT

This work does not involve human subjects, user studies, or the collection of new personally identifiable information. The experiments use publicly available or appropriately licensed datasets and standard machine learning evaluation benchmarks. We have reviewed the relevant data sources and experimental practices for potential ethical, privacy, and security concerns and do not identify any additional ethical considerations specific to this work. We adhere to the ICLR Code of Ethics and applicable research and data-use policies.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the training data composition, training stages, optimization settings, and post-training procedures in Sections 3-5. Further implementation details, including the model architecture, training hyperparameters, and post-training data sources, are provided in the Appendix. To support reproducibility and facilitate future research, we will release the training and evaluation code, model checkpoints, configurations, and associated scripts.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Aider. o1 tops aider's new polyglot leaderboard. `https://aider.chat/2024/12/21/polyglot.html`, December 2024. Aider blog post.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Anthropic. Claude 3.5 sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`, 2024.

Anthropic. The "think" tool: Enabling claude to stop and think in complex tool use situations. `https://www.anthropic.com/engineering/claude-think-tool`, March 2025.

Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. To code or not to code? exploring impact of code in pre-training. In *International Conference on Learning Representations*, volume 2025, pp. 79469–79495, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, et al. Smollm3: smol, multilingual, long-context reasoner. *Hugging Face Blog*, 2025.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wei Chen and Zhiyuan Li. Octopus v2: On-device language model for super agent. *arXiv preprint arXiv:2404.01744*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 177–198, 2024.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Richard Charles 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*, pp. 80–88, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Maxwell Jia. Aime 2024 dataset. Hugging Face, 2024. URL `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Niklas Lauffer, Xiang Deng, Srivatsa Kundurthy, Brad Kenstler, and Jeff Da. Imitation learning for multi-turn lm agents via on-policy expert corrections. *arXiv preprint arXiv:2512.14895*, 2025.

Changhao Li, Rushi Qiang, Jiawei Huang, Chenxiao Gao, Chao Zhang, Niao He, and Bo Dai. Revisiting dagger in the era of llm-agents. *arXiv preprint arXiv:2605.12913*, 2026a.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, et al. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *arXiv preprint arXiv:2606.15912*, 2026b.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*, 2024.

Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, et al. Toolace: Winning the points of llm function calling. In *International conference on learning representations*, volume 2025, pp. 41359–41381, 2025.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024a.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482, 2024b.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*, 2024.

MathArena. Aime 2025. Hugging Face Dataset, 2025. URL `https://huggingface.co/datasets/MathArena/aime_2025`.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Meta AI. Introducing muse glimmer: An open agentic model that runs on your device. `https://research.meta.ai/blog/introducing-muse-glimmer-open-agentic-model`, August 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text. In *International Conference on Learning Representations*, volume 2024, pp. 20357–20379, 2024.

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 *Forty-second International Conference on Machine Learning*, 2025.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *Advances in Neural Information Processing Systems*, 38, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Negin Raoof, Richard Zhuang, Marianna Nezhurina, Etash Guha, Atula Tejaswi, Ryan Marten, Charlie F Ruan, Tyler Griggs, Alexander Glenn Shaw, Hritik Bansal, et al. Openthoughts-agent: Data recipes for agentic models. *arXiv preprint arXiv:2606.24855*, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, 2024.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Changhui Sun, Lanbo Liu, Hang Lei, Tong Ling, Jiahang Xie, Zhiyong Zheng, Yujia Wang, Hao Liu, Feng Xiao, Lu Liu, et al. Step-level on-policy distillation: Interpolating between on-policy distillation and supervised fine-tuning. *arXiv preprint arXiv:2608.16333*, 2026.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Xiao Yan and Yi Ding. Are we there yet? a measurement study of efficiency for llm applications on mobile devices. In *Proceedings of the 2nd International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things*, pp. 19–24, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, 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. *arXiv preprint arXiv:2210.03629*, 2022.

Changsheng Zhao, Ernie Chang, Zechun Liu, Chia-Jung Chang, Wei Wen, Chen Lai, Sheng Cao, Yuandong Tian, Raghuraman Krishnamoorthi, Yangyang Shi, et al. Mobilellm-r1: Exploring the limits of sub-billion language model reasoners with open training recipes. In *International Conference on Learning Representations*, volume 2026, pp. 121936–121955, 2026.

A Experimental Setup & Evaluation Protocol

To facilitate reproducibility, we detail the model architecture and training recipe, including the data composition and training configurations across pre-training, mid-training, and post-training.

A.1 Model Architecture

Our architecture builds upon the designs of MobileLLM (Liu et al., 2024a) and Muse-Glimmer-30B (Meta AI, 2026). We adopt the Muse-Glimmer tokenizer, which has a 202,048-token subword vocabulary, together with SwiGLU (Shazeer, 2020) activations and RoPE (Su et al., 2024) with $\theta = 10^7$. We share the input and output embedding weights. The complete architectural specifications are summarized in Table 6.

Table 6: Detailed architecture specifications of X−LLM. "Dim" denotes the embedding dimension and "Hidden Dim" represents the dimension inside the feed-forward network.

Model	Layer	Head	KV-Head	Dim	Hidden Dim	Params
X−LLM-2.5B	32	32	8	2048	8192	2.49B

A.2 Training Recipe

The complete set of training hyperparameters is detailed in Table 7, with stage-specific configurations as follows:

Pre-training. The model is optimized with AdamW, using $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and a weight decay coefficient of 0.1. We employ a cosine learning-rate schedule with $7{,}500$ warmup steps. Training further incorporates *online* knowledge distillation from Muse-Glimmer-30B (Meta AI, 2026), using forward KL divergence with a teacher-loss weight of 0.6.

Mid-training. Starting from the pretrained checkpoint, we perform two consecutive 50,000-step knowledge-distillation phases. We distill from Muse-Glimmer-30B using pure forward KL divergence, with no hard-label loss. The teacher uses its own tokenizer with a vocabulary size of 200K and applies output soft-capping. The learning rate starts at 3.6279×10^{-4} and decays linearly to zero without warmup. The two phases draw from the same 184B-token mid-training data pool shown in Figure 3 and differ only in their data mixtures. After normalization, the proportions of mathematics and code data increase from 15.1% and 27.6% in Phase 1 to 49.0% and 39.2% in Phase 2, respectively.

Post-training. Post-training consists of three sequential joint-SFT stages. 1st General SFT stage combines Nemotron-Agentic-v1 dataset with low thinking effort and full Dolci-Think-SFT-32B corpus with high thinking effort and trains for two epochs. 2nd Agentic Reasoning SFT stage balances low-effort and high-effort data by supervised-token count. Its low-effort half consists of seven passes over OpenToughts-Agent (Raoof et al., 2026) and 14k synthesized data, while the high-effort half contains OpenMathReasoning, OpenCodeReasoning-2, and OpenScienceReasoning-2. All sources are jointly mixed for SFT training. All SFT stages use a maximum sequence length of 32,768, RoPE $\theta = 10^7$, no sequence packing, and loss masks stored over assistant tokens. We use zero weight decay, and gradient clipping at 1.0. In the on-policy knowledge distillation stage, the student first generates a trajectory prefix, after which the teacher completes the remaining interaction, including tool execution. The teacher-generated responses are materialized as offline training examples, and the model is optimized using standard next-token cross-entropy on the teacher-generated assistant tokens, with no KL-divergence loss or online teacher used during optimization.

B Evaluation Protocol.

We evaluate all models under a unified inference and evaluation pipeline spanning mathematical reasoning, knowledge, coding, tool use, and agentic tasks. Unless otherwise specified, evaluations use zero-shot prompting with the model's native chat template and enable thinking for reasoning-oriented

Table 7: Training setup for X−LLM. Note that these values denote the actual number of training tokens consumed, whereas Figure 3 reports the sizes of the underlying training corpora. Training data are sampled from these corpora according to the specified composition proportions; consequently, the number of tokens consumed during training may differ from the corresponding corpus sizes.

Stage	Phase	Tokens / Samples	Batch Size	Seq. Len.	Updates / Epochs	Peak LR	#GPUs
Pre-training	Single phase	1.2T tokens	128	16k	600k	4e-3	16×8
Mid-training	Phase 1	105B tokens	128	16k	50k	3.63e-4	16×8
	Phase 2	105B tokens	128	16k	50k	3.63e-4	16×8
Post-training	General SFT	2.4M samples	1024	32k	2 epochs	8e-5	16×8
	Agentic Reasoning SFT	7.1M samples	1024	32k	1 epoch	4e-5	16×8
	On-policy KD	13k samples	1024	32k	1 epoch	4e-5	16×8

benchmarks. All generative evaluations use a standardized decoding configuration of temperature 0.6, top-p 0.95, and top-k 20, while benchmarks with deterministic evaluation use greedy decoding. We report the benchmark-native metric over the full evaluation set.

For mathematical reasoning, we evaluate on AIME'24 (Jia, 2024), AIME25 (MathArena, 2025), MATH500 (Hendrycks et al., 2021), and GSM8K (Cobbe et al., 2021). AIME is evaluated with 16 sampled generations per problem and reported using the corresponding average accuracy across samples, while MATH500 and GSM8K use single-sample pass@1. We use up to 30K generated tokens for long-horizon reasoning, subject to each model's supported context length. For knowledge evaluation, we use zero-shot MMLU (Hendrycks et al., 2020) with log-probability scoring and report the macro-average across its four subject groups. GPQA Diamond (Rein et al., 2023) is evaluated zero-shot with one sampled generation per question and reports pass@1.

For code generation, we evaluate HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), using pass@1, with 0-shot and 3-shot prompting, respectively. We additionally evaluate LiveCodeBench (v6) (Jain et al., 2025) using 16 samples per problem and report the mean pass rate over all graded samples. For agentic tool use, we evaluate BFCL v4 (Patil et al., 2025) using its full suite, and report the benchmark's overall accuracy. SWE-bench Verified (Jimenez et al., 2024) is evaluated through a standardized terminal-agent scaffold with 4 independent attempts per task, and we report task-level mean pass@1.

We further assess out-of-distribution agentic capabilities on Terminal-Bench 2 (Merrill et al., 2026), Aider Polyglot (Aider, 2024), and BFCL-Parity. Each task is evaluated with three independent attempts under the same terminal-agent scaffold, and we report the mean success rate over all graded task attempts. BFCL-Parity is evaluated as a separate 123-task agentic benchmark with a standardized Terminus-based scaffold and task construction distinct from native BFCL V4, and is therefore reported independently rather than as a BFCL V4 result.

Unless otherwise noted, agentic evaluations use a maximum of 4K generated tokens per turn and a 32K input-context budget, with automatic context summarization for long trajectories. All evaluations are conducted using the same inference stack and model-specific supported context limits, ensuring a consistent evaluation protocol across model families.