CURRICULUM LEARNING TEACHES LLMS LONG-HORIZON IMPLICIT REASONING, UP TO A LIMIT

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs "reason" by externalizing intermediate steps as tokens through chain-of-thought (CoT), yet even frontier reasoning models hit abrupt performance cliffs on long-horizon tasks. CoT commits to one step at a time, and the model's *implicit reasoning*—reasoning within a single forward pass—sets how far the model can search ahead to decide the correct next move. Models with weak implicit reasoning are more likely to commit to wrong paths and must attempt to recover via backtracking, while models with strong implicit reasoning identify the correct proof strategy directly, shortening the CoT. However, prior work shows LLMs struggle to reason implicitly beyond shallow depths, with no relief from scale. We investigate whether curriculum learning can extend this depth by fine-tuning on gradually increasing vocabulary size and *lookahead*—the number of steps a model must search ahead to identify the correct next move. At matched compute, curriculum-trained models achieve lookaheads where no-curriculum training fails—`Qwen3-0.6B` reaches 98% accuracy at lookaheads up to 256 with curriculum, compared to 45% without. However, progress slows with training compute, following a saturating exponential that suggests a *maximum intrinsic lookahead* to which models can learn to reason implicitly. Curriculum step sizes that are too large stall progression, larger models reach higher fitted ceilings, and pretrained initialization far outpaces training from scratch. Finally, the capability acquired on synthetic data transfers to natural language reasoning, with curriculum-trained models outperforming models trained without curriculum at matched compute on 11 natural language reasoning benchmarks.

1 INTRODUCTION

Large language models (LLMs) are increasingly applied in reasoning-intensive settings such as coding (Jimenez et al., 2024), deductive proof generation (Polu & Sutskever, 2020), planning (Valmeekam et al., 2023), multi-hop question answering (Yang et al., 2018), and game AI (Silver et al., 2018). Reasoning in LLMs is typically performed by externalizing intermediate tokens through chain-of-thought (CoT; Wei et al., 2022), and reinforcement learning has been used to further improve reasoning performance (Jaech et al., 2024; Guo et al., 2025). However, even state-of-the-art reasoning models struggle on long-horizon tasks, hitting abrupt performance cliffs at modest depths, and these failures persist even when models are given their full reasoning-token budgets (Rameshkumar et al., 2025). Errors early in a reasoning chain can compound over the remaining steps (Dziri et al., 2023), forcing models to spend substantial tokens backtracking in order to recover. Even when models avoid such mistakes, reasoning chains can consume a considerable amount of tokens when each reasoning step makes only a small advance towards the solution.

Implicit reasoning occurs within a single forward pass and determines how far a model can "look ahead" over potential later steps before committing to the next CoT step (Saparov et al., 2025). This capability becomes increasingly important in long-horizon reasoning tasks, which require the model to look ahead to greater depths. Consider the example in Figure 1, where a model with weak implicit reasoning commits to a wrong path and must attempt to recover via backtracking, while a model with strong implicit reasoning looks far enough ahead to identify the correct next step directly, shortening the CoT. Despite these benefits, recent work has shown that transformers struggle to learn implicit search beyond shallow depths, with no relief from increased model scale (Saparov et al., 2025). This

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.

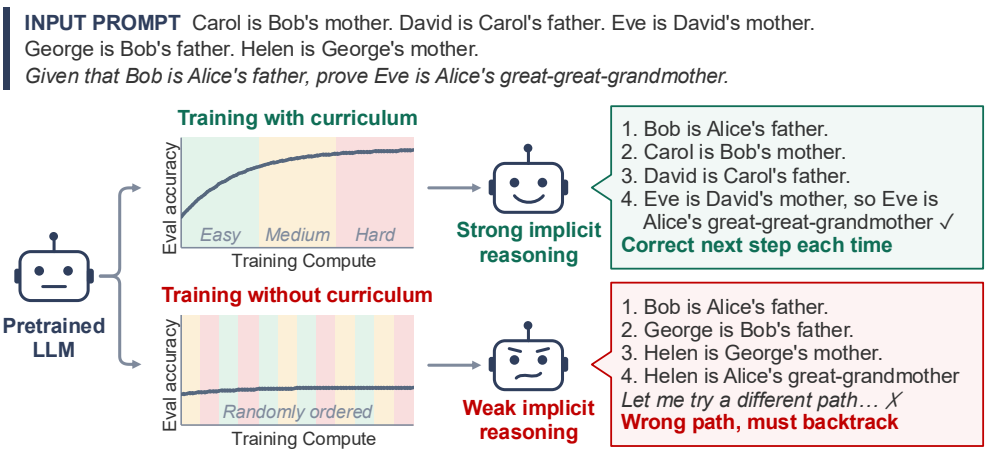

FIGURE 1: Curriculum versus non-curriculum fine-tuning of the same pretrained LLM. The curriculum orders training problems by increasing difficulty, whereas the baseline draws from the full range of difficulties from the start. On the example proof, the curriculum-trained model takes the correct next step each time, while the baseline commits to a wrong branch and must backtrack. Section 4.1 reports the measured comparison.

raises the question: can LLMs be trained to perform long-horizon implicit reasoning, and if so, is there a limit to the depth they can reach?

To address this question, we examine whether the training procedure itself can be shaped to unlock deeper implicit reasoning. Incrementally increasing the depth a model must look ahead during training might offer a natural path to do so, and we therefore investigate whether curriculum learning (Bengio et al., 2009) can extend this depth. We generate training data using a modified version of `DeepRD` (Rameshkumar et al., 2025), where each example provides a set of premises in natural language over entities and attributes (e.g., "cat" in "Alex is a cat") and asks the model to predict the next step of a multi-hop proof. We deliberately keep the curriculum simple, as our goal is not to find the optimal design but to characterize the behavior of curriculum training itself. Our curriculum gradually increases the *lookahead*—the number of steps a model must search ahead to identify the correct next step—jointly with the attribute vocabulary, which we find outperforms a curriculum over lookahead alone (Section 4.1). As an illustration, the lookahead in Figure 1 is 3, since the model must search 3 steps ahead to conclude that "Carol is Bob's mother," not "George is Bob's father," is the correct next step. By training only on the output of a single forward pass, we isolate implicit reasoning as a pure computational capability, separating it from what can be achieved by externalizing intermediate steps. With this setup, we pose two primary research questions. **RQ1**: Can LLMs learn to reason implicitly to arbitrary depths, and what training conditions—including curriculum, model size, and initialization—govern this capability? **RQ2**: Does training a model to perform long-horizon implicit reasoning improve its performance on downstream natural-language reasoning tasks?

We find that curriculum learning can extend the depth at which LLMs perform implicit reasoning. At matched compute, curriculum-trained models achieve lookaheads where no-curriculum training fails—`Qwen3-0.6B` reaches 98% accuracy in one setting with curriculum vs. 45% without (Section 4.1). Achieved lookahead grows predictably with training compute following a saturating exponential function, suggesting a *maximum intrinsic lookahead* to which each model can learn to reason implicitly (Section 4.2). The step size of the curriculum, the increase in lookahead between successive stages, must be small enough for training to progress—with `Qwen3-0.6B`, step size 64 stalls at lookahead 128, while step size 8 reaches 256 (Section 4.3). Larger models progress through the curriculum faster and appear to reach a higher maximum intrinsic lookahead, with fitted values of 51, 82, and 235 for `Pythia-410M`, `1.4B`, and `2.8B`, respectively (Section 4.4). At matched compute, it is much easier to train a model initialized from pretrained weights than one initialized randomly—at a budget of 1.5M PFLOPs, `Pythia-1.4B` completes lookahead 63 from pretrained initialization vs. 23 from random initialization (Section 4.5). Finally, the implicit reasoning capability acquired on synthetic data transfers to natural language reasoning, with curriculum-trained models outperforming models trained without curriculum at matched compute on 11 benchmarks, with the largest gains on multi-step deductive reasoning—`Qwen3-1.7B` gains 14.3 points on `ProofWriter OWA` at lookahead 48 (Section 4.6). In summary, we make the following core contributions:

C1 We introduce a curriculum over lookahead and attribute vocabulary and show that it extends the depth to which LLMs can reason implicitly (Section 4.1), with achieved lookahead following a saturating exponential in training compute, suggesting a *maximum intrinsic lookahead* to which a model can learn to reason implicitly (Section 4.2).

C2 We show that curriculum step size must be sufficiently small for training to progress (Section 4.3), depth reached scales with model size (Section 4.4), and pretrained initialization allows a model to reason implicitly to far greater depths than training from scratch (Section 4.5).

C3 We show that training with curriculum on synthetic data transfers better to downstream natural-language reasoning than training without curriculum at matched compute (Section 4.6).

2 Related Work

Reasoning and search in transformers. A growing line of work characterizes how LLMs struggle on long-horizon reasoning. Dziri et al. (2023) argue that transformers reduce multi-step reasoning to pattern matching, so errors compound as tasks grow more complex. Formal analyses of deductive reasoning show that LLMs reason greedily at branch points (Saparov & He, 2023) and that their accuracy drops as the correct proof grows longer (Saparov et al., 2023). Saparov et al. (2025) formalize task complexity via a *lookahead* metric and show that while small transformers can learn to search under a carefully balanced training distribution, learning degrades as graphs grow, with no relief from model scale. Rameshkumar et al. (2025) show that even frontier reasoning models hit a performance cliff at modest lookaheads on graph connectivity and proof planning, without exhausting their token budgets. Heyman & Zylberberg (2025) report similar failures on graph-coloring tasks.

Explicit and implicit reasoning. Explicit reasoning externalizes tokens via CoT prompting (Wei et al., 2022) and its derivatives (Wang et al., 2023b; Yao et al., 2023), extended further by trace-supervised search (Lehnert et al., 2024; Gandhi et al., 2024) and reinforcement learning (Lambert et al., 2024; Jaech et al., 2024; Guo et al., 2025). Implicit approaches enhance reasoning without verbalizing intermediate steps via learnable auxiliary tokens (Goyal et al., 2024), latent vectors in place of discrete tokens (Hao et al., 2025; Shen et al., 2025; Wei et al., 2026), or architectural recurrence (Dehghani et al., 2019; Geiping et al., 2025). Mechanistic studies show transformers can reason implicitly within a forward pass (Brinkmann et al., 2024; Jenner et al., 2024), and it can emerge through grokking, but only with significant over-training (Wang et al., 2024). However, implicit reasoning can reflect shortcut learning rather than generalizable reasoning (Lin et al., 2025b), and its generalization depends on which compositional structures appear in training (Ye et al., 2025).

Curriculum learning for reasoning. In early curriculum learning work, Bengio et al. (2009) train language models on a restricted vocabulary that is progressively expanded; our attribute-vocabulary curriculum applies the same idea to the attributes of synthetic proofs. Curriculum learning has since been used to improve performance on reasoning tasks, for example by gradually removing CoT tokens or replacing them with hidden states while the problems stay fixed (Deng et al., 2024; Hao et al., 2025). Our curriculum instead never uses CoT and increases the lookahead of the problems. Recent theoretical results show that staged curricula can avoid exponential sample-complexity bottlenecks (Wang et al., 2025; Bu et al., 2026; Huang et al., 2026) and can let agents trained with reinforcement learning search trees deeper than those seen in training (Yang et al., 2026). Empirically, the picture is mixed: curriculum learning improves learning efficiency (Abbe et al., 2024) and mitigates, but does not eliminate, the exponential data cost of implicit k-hop reasoning in small from-scratch models (Yao et al., 2025), while difficulty-ordering a fixed dataset gives no robust advantage for explicit CoT post-training (Mordig et al., 2026). Lee et al. (2025) report that pretrained initialization accelerates curriculum progression. Whether a curriculum can unlock implicit reasoning in pretrained LLMs at depths where direct training fails has remained open; our work answers this question.

3 Methods

3.1 Task Formulation and Complexity

We study whether LLMs can learn to perform long-horizon reasoning implicitly, where the model emits the next proof step directly, without intermediate reasoning tokens. To do so, we focus on the task of predicting the next step in a proof: given a set of premises and a goal conclusion, the model must identify the correct next step in a valid proof of the goal from the premises. A proof is a

sequence of proof steps, where each proof step has some premises followed by a conclusion, and the conclusion of one proof step may serve as a premise for a later proof step. Solving the task requires the model to search through possible inference paths, starting from the premises, and find a proof from the premises to the goal.

Training Data. To generate examples of this task, we use a modified version of the `DeepRD` dataset. In `DeepRD`, each problem is generated by first generating a directed acyclic graph (DAG). Each graph is then translated into natural language by mapping every node into an atomic proposition (e.g., "Alice is a cat") and every edge into a rule (e.g., "If Alice is a cat, then Alice is a mammal"). In `DeepRD`, attribute names, such as "cat", are replaced with fictional names (e.g., "zorbi") to facilitate the generation of very large reasoning problems and to avoid data contamination issues during evaluation. Since `DeepRD` is synthetic, we can generate a limitless number of examples for large-scale training. Some examples are shown in Appendix H. A key reason we choose `DeepRD` is that our curriculum requires examples whose lookahead we can control and make arbitrarily large. Real-world reasoning data cannot meet this requirement, because its complexity is long-tailed (Rameshkumar et al., 2025) and examples with large lookahead are rare. The representation of each reasoning problem as a graph enables us to precisely control problem complexity: in the graph analogy, finding a proof from a premise to a goal is equivalent to finding a directed path from a start node to a goal node.

Lookahead. We measure task complexity using the *lookahead* ℓ which is the minimum number of breadth-first search iterations that must be performed from the start node to determine the correct next step in the path to the goal. Intuitively, ℓ captures how far the model must search to determine the correct path from incorrect alternatives.[1] This is distinct from the length of the path from the start node to the goal node. For example, if the start node has only one valid child node, no search is needed to determine the next correct step, regardless of the distance of the goal node from the start node. In this case, $\ell = 1$, and lookahead is a more faithful characterization of problem complexity.

3.2 Curriculum over Lookahead and Attribute Vocabulary

We aim to test whether curriculum learning can help transformers to learn to perform long-horizon implicit reasoning. With this in mind, we develop a curriculum learning procedure (Figure 8 and Algorithm 1, Appendix A). We train the model in stages: at stage k, the model is presented with examples whose lookahead is at most L_k, where $L_k = k \cdot s$ and $s \geq 1$ is the *curriculum step size*—i.e., the amount by which the lookahead is increased at each stage of the curriculum. For example, with $s = 8$, a 12-stage curriculum advances through $L_k = 8, 16, 24, \ldots, 96$. Figure 12 illustrates the first three stages of a curriculum with $s = 1$. At each stage, we perform supervised fine-tuning on examples from `DeepRD` where, for each example i, the lookahead of the example ℓ_i is sampled i.i.d. from a uniform distribution $\ell_i \sim \text{Uniform}\{1, \ldots, L_k\}$. Including easier examples at each stage helps prevent catastrophic forgetting of previously learned lookaheads. The training objective is cross-entropy on the answer tokens: the tokens identifying the correct next attribute in the proof chain. The model advances from stage k to stage $k + 1$ when its exact-match accuracy over the most recent W training examples reaches 98%.

Attribute-vocabulary curriculum. Besides lookahead, the number of distinct attribute labels that appear across training examples also affects the difficulty of the task. We therefore also grow the attribute vocabulary as part of the curriculum, following the vocabulary curriculum of Bengio et al. (2009). The attributes in each generated proof are given fictional labels (e.g., `zorbi` in "Alice is zorbi"); the *attribute vocabulary* $\mathcal{V}_k$ is the pool of these attribute labels available to the instances at stage k, and $|\mathcal{V}_k|$ denotes its size. We emphasize that $|\mathcal{V}_k|$ does *not* refer to the model's tokenizer vocabulary. The tokenizer and its token set are unchanged, and each attribute label is encoded as ordinary subword tokens. We grow $|\mathcal{V}_k|$ with lookahead L_k so that the vocabulary is always sufficient to provide distinct attributes with distinct labels. Specifically, an instance at lookahead L_k contains at most $2L_k + 1$ distinct attributes: the goal-directed chain of length L_k plus a distractor branch of length L_k that shares the initial attribute. We set $|\mathcal{V}_k| = 2(L_k + 1)$, one more than this maximum, so every attribute in an instance can receive a distinct label. A pool smaller than $2L_k + 1$ would force two or more attributes in the largest instances to share a label, which would make the problem ill-posed. We explore whether this *joint* curriculum over both lookahead and attribute vocabulary size performs better than a curriculum over lookahead alone (Section 4.1). In the latter setting, holding

[1] For a formal definition, see Rameshkumar et al. (2025); Saparov et al. (2025).

the pool fixed at its maximum size from the start leaves proof structure unchanged and only reduces cross-instance reuse of attribute labels.

4 Experiments and Results

Models. We use two model families. `Qwen3` (0.6B, 1.7B; Yang et al., 2025) gives capable models at low training cost across our many curriculum configurations, and it has fairly strong instruction following capabilities, which facilitates evaluation using downstream benchmarks that require specifically formatted outputs (Section 4.6). `Pythia` (160M, 410M, 1.4B, 2.8B; Biderman et al., 2023) provides sizes that share the same pretraining data and data order, together with base (non-instruction-tuned) checkpoints; our scaling and initialization comparisons (Sections 4.4 and 4.5) rely on both properties.

Setup. All training examples are generated with our modified `DeepRD` generator as described in Sections 3.1 and 3.2. Experiments perform full fine-tuning with AdamW, a constant learning rate, and bf16 precision; per-experiment hyperparameters are listed in Appendix B. Training compute is estimated as $C \approx 6ND$ (N parameters, D tokens; Kaplan et al., 2020).

Evaluation. The comparison with training without curriculum (Section 4.1) uses exact-match accuracy of the predicted next attribute on a held-out set shared by both settings, with lookaheads sampled uniformly from 1 to L (Appendix E).[2] All other synthetic-task experiments report *achieved lookahead* L, the last curriculum stage completed, i.e., the largest L at which the rolling exact-match training accuracy reaches 98% (Section 3.2).

Baselines. In both the next-step proof prediction comparisons (Section 4.1) and the downstream transfer experiments (Section 4.6), the baseline is the same model trained without curriculum, directly on data with a fixed maximum lookahead L from the start, compared against its curriculum-trained counterpart at matched training compute.

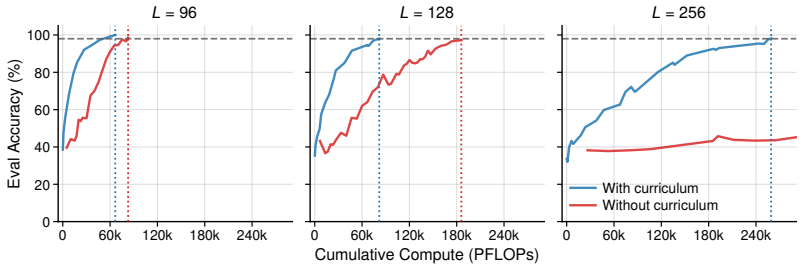

Figure 2: Eval accuracy of `Qwen3-0.6B` at maximum $L \in \{96, 128, 256\}$ vs. cumulative training compute (PFLOPs). The dashed line marks the 98% threshold; dotted lines mark where each setting crosses it (Table 1).

4.1 Curriculum Learning Enables Deeper Reasoning at Matched Compute

Curriculum versus no curriculum. To determine whether curriculum learning is needed to train models to perform long-horizon reasoning implicitly, we compare training `Qwen3-0.6B` with curriculum learning at $s = 8$ against training without curriculum learning, where the maximum lookahead is fixed from the start. We run this comparison for each maximum lookahead $L \in \{96, 104, 112, 120, 128, 256\}$. Figure 2 shows evaluation accuracy vs. cumulative training compute for $L \in \{96, 128, 256\}$, with the full sweep reported in Appendix C.1. Table 1 summarizes the compute required to reach 98% evaluation accuracy across all six values of L. For $L \leq 128$ both methods eventually converge, but training with curriculum is consistently 1.25–$2.27\times$ **more compute-efficient**. At maximum lookahead L=256 the gap becomes qualitative: training without curriculum plateaus at $\sim$45% accuracy and never converges within the compute budget, while training with curriculum continues climbing to 98%.

Table 1: Cumulative training compute (PFLOPs) to reach 98% accuracy at each target lookahead L. A dash marks a setting that never reaches 98% within budget.

L	Curriculum	No curriculum	Speedup
96	66.5K	82.8K	$1.25\times$
104	66.5K	106K	$1.59\times$
112	74.3K	140K	$1.89\times$
120	74.3K	106K	$1.42\times$
128	81.8K	186K	$2.27\times$
256	259K	–	–

[2] During the training curriculum, the maximum lookahead of samples reaches L only at the final stage.

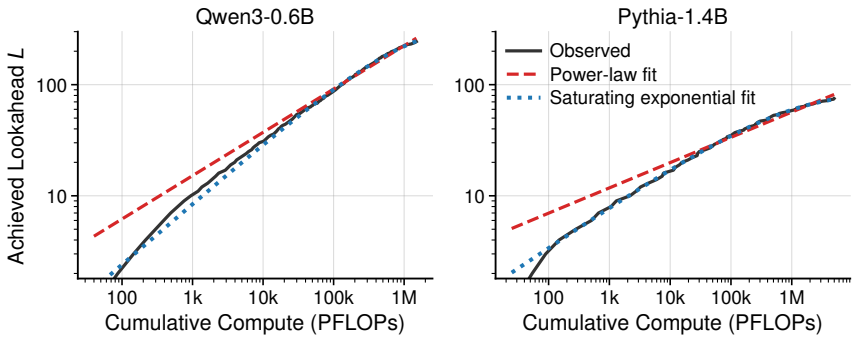

FIGURE 4: Achieved lookahead L vs. cumulative training compute (PFLOPs, log–log scale) for `Qwen3-0.6B` (*left*) and `Pythia-1.4B` (*right*) trained under the curriculum with s=1, with the power-law and saturating exponential fits overlaid on the observed trajectory.

Growing versus fixed attribute vocabulary. We next test the design choice introduced in Section 3.2: using a curriculum over both attribute vocabulary size and lookahead, with $|\mathcal{V}_k| = 2(L_k + 1)$, versus using a curriculum over lookahead alone, with the attribute vocabulary fixed at its maximum size throughout training. We train `Qwen3-0.6B` under both configurations, holding all other hyperparameters constant. Figure 3 and Table 9 show that the joint curriculum completes L=128 within 100K PFLOPs, where the fixed-vocabulary configuration has completed only L=96. One plausible explanation is the following: at a fixed (large) attribute vocabulary, the model sees fewer examples of each name in varied reasoning contexts, encouraging memorization rather than generalization, while the joint curriculum keeps the attribute vocabulary small in early stages and grows it progressively as the model masters larger lookaheads. We therefore conduct all subsequent experiments using the joint curriculum over lookahead and attribute vocabulary size.

Vocabulary growth rate. Beyond whether to grow the attribute vocabulary at all, we also test how fast to grow it. We compare $|\mathcal{V}_k| = bL_k + 2$ for $b \in \{2, 3, 4\}$ on `Qwen3-0.6B` with $s = 8$ under an otherwise identical curriculum (Figure 3). At $b = 2$, this recovers our default schedule, $|\mathcal{V}_k| = 2L_k + 2 = 2(L_k + 1)$. For a given lookahead, a larger b "widens" the deepest instances from 2 branches to b, rather than simply enlarging the attribute vocabulary while holding proof structure fixed, since $|\mathcal{V}_k|$ is one more than the maximum number of attributes in an instance. This comparison therefore varies proof *breadth* alongside attribute vocabulary size. Within a budget of 100K PFLOPs, the three settings complete L=128, 72, and 64, respectively. Thus, the curriculum design that advances fastest is the one that grows the attribute vocabulary while holding the deepest instances at two branches, and we use this configuration in all subsequent experiments.

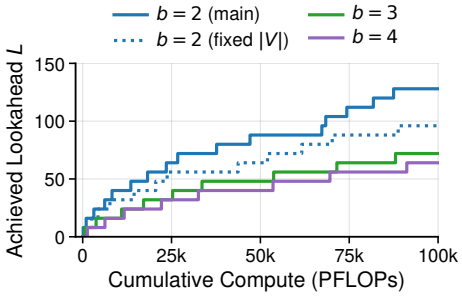

FIGURE 3: Achieved lookahead L vs. cumulative training compute (PFLOPs) for `Qwen3-0.6B` at s=8, with the attribute vocabulary grown as $|\mathcal{V}_k| = bL_k + 2$ or held fixed at its maximum size.

4.2 ACHIEVED LOOKAHEAD GROWS PREDICTABLY TOWARD AN APPARENT CEILING

Having shown that curriculum is instrumental in teaching models to perform long-horizon implicit reasoning, we examine how achieved lookahead grows with compute. Figure 4 plots achieved L against cumulative training compute (PFLOPs, log–log) for the trajectories of two models trained under our curriculum with $s = 1$: `Qwen3-0.6B` and `Pythia-1.4B`. In both cases achieved L grows smoothly with compute but decelerates, curving below a straight line on log–log axes. We fit two functional forms[3]: a power law $L = aC^p + L_0$, the simplest unbounded form, and a saturating exponential $L = L_{\max}\big(1 - \exp\big(-(C/C_0)^k\big)\big)$, the simplest bounded form, with a ceiling $L_{\max}$.

Under a $L_0 \geq 0$ constraint (to ensure positive lookahead) the fitted offset vanishes, leaving a pure power law. Table 2 gives the fitted forms and their in-sample RMSE.

[3]We also fit a shifted power law and a saturating power law, which reduce to the power law and to a logarithmic form, respectively. The logarithmic form fits worse than the saturating exponential on every trajectory.

TABLE 2: Fitted forms of achieved lookahead L against cumulative training compute C (PFLOPs), with in-sample RMSE.

Trajectory	Form	Fitted equation	RMSE
`Qwen3-0.6B`	Power law	$L \approx 1.03\, C^{0.390}$	6.10
	Saturating exponential	$L \approx 324\big(1 - \exp\big(-(C/7.7{\times}10^5)^{0.547}\big)\big)$	2.22
`Pythia-1.4B`	Power law	$L \approx 2.45\, C^{0.228}$	2.98
	Saturating exponential	$L \approx 82\big(1 - \exp\big(-(C/5.0{\times}10^5)^{0.371}\big)\big)$	0.53

The saturating exponential achieves lower in-sample RMSE than the power law by factors of 2.7 and 5.6 on `Qwen3-0.6B` and `Pythia-1.4B`, respectively.

We next test whether this in-sample advantage extends to prediction. We fit both forms on the first 70% of each trajectory's stage completions, ordered by cumulative compute, and evaluate on the remaining 30%. Held-out RMSE is 1.4× and 23.8× lower for the saturating exponential on `Qwen3-0.6B` and `Pythia-1.4B` (Table 3). A Diebold–Mariano test (Diebold & Mariano, 1995) is a standard comparison of held-out forecast errors. Because errors along a single trajectory are serially correlated, however, we instead apply the test to the number of training iterations required to complete each stage, whose residuals are approximately independent (Appendix D.1). On both trajectories, the saturating exponential predicts the held-out per-stage iteration counts significantly better than the power law ($p = 7{\times}10^{-10}$ for `Qwen3-0.6B` and $p = 1.5{\times}10^{-4}$ for `Pythia-1.4B`; Table 3). The advantage also increases with lookahead, as the gap between the two forms' held-out prediction errors widens at later stages. Appendix D reports the results of the same comparisons on `Pythia-410M` and `Pythia-2.8B`.

TABLE 3: Held-out RMSE for achieved lookahead L. p-value: significance of the saturating exponential's advantage under the per-stage iteration test, two-sided. Additional fittings on less-trained trajectories in Table 15.

	Held-out RMSE		
Trajectory	Power law	Sat. exp.	p-value
`Qwen3-0.6B`	23.88	17.08	$7{\times}10^{-10}$
`Pythia-1.4B`	16.14	0.68	$1.5{\times}10^{-4}$

Within our observed range both forms describe the data closely, but achieved L decelerates faster than a power law allows. The saturating fit is better, and it assigns each model a finite ceiling $L_{\max}$. The slowdown is particularly pronounced for `Pythia-1.4B`. Within a budget of 5M PFLOPs, it reaches L=75, 91% of the fitted asymptote $L_{\max} \approx 82$. Its last five completed stages, L=71–75, consume 2.10M PFLOPs, approximately 42% of the budget, and the per-stage cost is still rising at the end of training (Appendix D.2, Figure 11).

These trajectories suggest a model- and procedure-specific ceiling on achievable lookahead: for a given model, our curriculum appears to drive L upward with diminishing returns toward that ceiling.

4.3 SMALL STEP SIZES ARE REQUIRED FOR DEEP CURRICULUM PROGRESSION

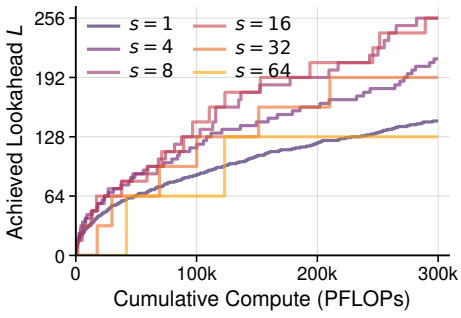

FIGURE 5: Achieved lookahead L vs. cumulative training compute for `Qwen3-0.6B` with step sizes $s \in \{1, 4, 8, 16, 32, 64\}$.

Our curriculum has one main hyperparameter, the step size s, which controls the increase in maximum lookahead L between consecutive stages. We examine how s affects the lookahead the curriculum reaches within a fixed compute budget. We tune `Qwen3-0.6B` using our curriculum at step sizes $s \in \{1, 4, 8, 16, 32, 64\}$, each with a target lookahead of L=256.

Figure 5 shows that the step size must be neither too large nor too small. A step size that is too large stalls curriculum advancement: s=64 stalls at L=128 and s=32 at L=192. A step size that is too small does not stall but advances slowly: within 300K PFLOPs, s=1 reaches only L=145 and s=4 reaches L=212, whereas s=8 and s=16 complete L=256. A plausible explanation for stalling is that larger s makes training more similar to the no-curriculum case: at s equal to the target lookahead the curriculum has a single stage, which is exactly training without curriculum, and Section 4.1 shows that this stalls at L=256. The higher the target, the smaller s must be to avoid stalling.

4.4 Maximum Intrinsic Lookahead Grows with Model Scale

Prior work finds that scaling model size alone does not improve implicit reasoning (Saparov et al., 2025). We test whether this conclusion holds with curriculum learning, and whether the curriculum's progression rate depends on model size. We train `Pythia` models of four different sizes (160M, 410M, 1.4B, 2.8B) using our curriculum with $s = 1$, with learning rate and batch size individually tuned per model size (Appendix B).

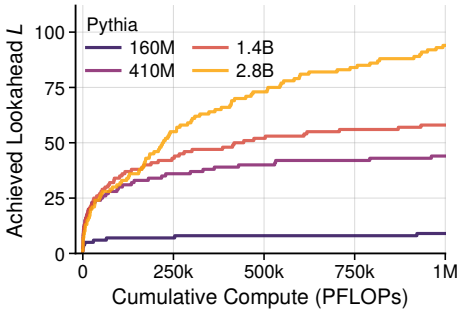

FIGURE 6: Achieved lookahead L vs. cumulative training compute for the four `Pythia` sizes at s=1. At 1M PFLOPs, the curves reach L=9, 44, 58, and 94, from smallest to largest.

At matched compute, larger models reach greater lookaheads once the training budget is sufficiently large (Figure 6). `Pythia-2.8B` achieves the deepest lookahead among the tested `Pythia` models (L=133). It completes L=130 using 2.40M PFLOPs, less than half the compute `Pythia-1.4B` uses to reach L=75 (4.97M PFLOPs). `Pythia-160M` plateaus at L=9 over the observed trajectory, suggesting that the model's capacity is insufficient for curriculum progression.

Scaling of the fitted asymptote with model size. The saturating exponential's asymptote $L_{\max}$ is itself model-dependent. We fit the saturating-exponential curve introduced in Section 4.2 separately to each `Pythia` trajectory except that of `Pythia-160M`, which plateaus at L=9. The fitted asymptote rises with model size: $L_{\max} \approx 51$, 82, and 235 for `Pythia-410M`, `-1.4B`, and `-2.8B` respectively (per-size fits in Appendix D, Figure 10; fitted parameters in Table 17). Parameter count is not the whole story, however: `Qwen3-0.6B` has a fitted asymptote of $L_{\max}\approx$324, well above the larger `Pythia-1.4B`'s ≈82, suggesting that factors beyond parameter count also affect a model's maximum intrinsic lookahead. Bootstrap intervals (Table 17) show that the ordering of these ceilings is robust.

4.5 Pretrained Initialization Reaches Greater Lookahead at Matched Compute

Lee et al. (2025) report that pretrained initialization accelerates curriculum progression in arithmetic and maze-solving tasks. We test whether this effect holds for our curriculum learning in reasoning and quantify the size of the gap. We train `Pythia-1.4B` and `Qwen3-0.6B`, each twice under our curriculum with s=1: once from the released pretrained checkpoint (full fine-tuning) and once from random weights (training from scratch). Learning rate and batch size are tuned separately for each condition, since random initialization makes little progress under the small learning rate suited to fine-tuning (Appendix B). Pretraining gives a large head start in both families (Figure 7). Within a budget of 1.5M PFLOPs, random initialization reaches L=23 for `Pythia-1.4B` and L=35 for `Qwen3-0.6B`. Pretrained initialization reaches the same lookaheads with ∼53× and ∼117× less compute, respectively (Table 14, Appendix C.4). That the stronger, more heavily pretrained model benefits more suggests the curriculum builds on capabilities already present in the pretrained weights rather than instilling them from scratch.

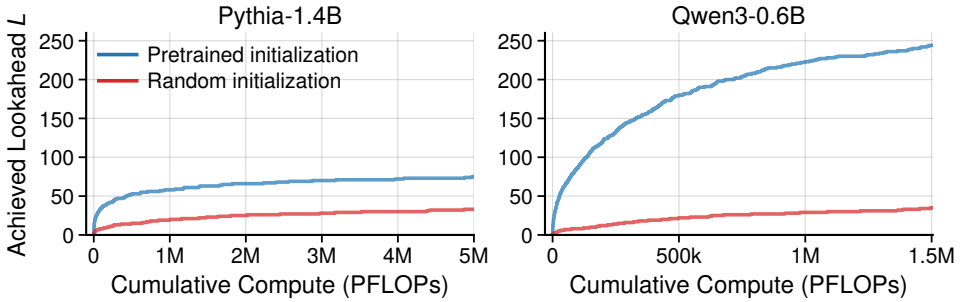

FIGURE 7: Achieved lookahead L vs. cumulative training compute (PFLOPs), starting from the pretrained checkpoint vs. random initialization. *Left:* `Pythia-1.4B`, *Right:* `Qwen3-0.6B`.

Pretrained models have a larger fitted asymptote. We train the randomly initialized `Qwen3-0.6B` long enough to fit the curve of Section 4.2, which gives $L_{\max} \approx 120$, against ≈ 324 from pretrained

initialization. This suggests that without pretrained knowledge, the model not only progresses more slowly but also reaches a lower ceiling.

4.6 Implicit Reasoning Learned on Synthetic Data Transfers Downstream

We now ask whether the implicit-reasoning capability acquired through curriculum learning on synthetic data transfers to natural language reasoning benchmarks. We fine-tune `Qwen3-1.7B` with our curriculum ($s = 1$). To avoid catastrophic forgetting of the instruction-following capability that downstream benchmarks require, we draw 6% of training examples, outside the curriculum, from the instruction-tuning corpus `Dolci-Instruct-SFT` (Olmo Team, 2025).

We evaluate at curriculum stages $L \in \{8, 16, 32, 48\}$ on 14 downstream reasoning tasks spanning logical, multi-hop, constraint-satisfaction, and graph reasoning (full list in Table 4; descriptions in Appendix G). Evaluating at multiple stages tracks how the transfer evolves as the curriculum deepens. We compare against the same model trained without curriculum, directly at L=48, compute-matched to the final curriculum checkpoint.

Table 4 reports the results, and Appendix F additionally reports the base model and an instruction-only control. At L=48, training with curriculum outperforms training without curriculum on 11 of the 14 benchmarks (citations in Appendix G), with statistically significant gains on both `ProofWriter` settings, `JustLogic`, `ZebraLogic-MC`, `PrOntoQA`, `ParaRules`, `FOLIO`, and `LogicBench-BQA`. On the three benchmarks where the model trained without curriculum scores higher, none of the differences is significant. These results suggest that training a model with curriculum learning to perform long-horizon implicit reasoning can improve its downstream reasoning. Optimizing the training procedure to maximize these gains is left to future work.

General model capability is preserved. On eight language-understanding benchmarks and `IFEval`, the curriculum-trained `Qwen3-1.7B` exceeds the non-finetuned model at every curriculum stage (eight-benchmark mean 63.4 to 64.2 vs. 61.7) and stays within one point of a compute-matched model fine-tuned only on instruction data (Table 22, Appendix F.1). Thus, we find no evidence that gains in implicit reasoning come at the expense of general capability.

Table 4: Downstream reasoning accuracy (%) for `Qwen3-1.7B` at curriculum stages L, vs. trained directly at L=48 without curriculum at matched compute. Δ: curriculum at L=48 minus no curriculum, computed before rounding. Best per row in bold. *: curriculum wins are statistically significant (two-sided exact McNemar test on paired items, $p < 0.05$); confidence intervals in Appendix F. †Extended by us to test for greater depths (Appendix G).

	Curriculum L				No curr.	Δ
Benchmark	**8**	**16**	**32**	**48**		
ProofWriter OWA	41.1	33.6	39.3	**54.9***	40.6	+14.3
ProofWriter CWA	66.8	57.2	67.6	**67.9***	55.2	+12.8
ParaRules	46.1	36.2	39.1	**55.1***	50.4	+4.7
ZebraLogic-MC	47.4	53.4	51.8	**53.5***	48.3	+5.2
PrOntoQA†	52.5	55.3	56.8	**57.1***	52.0	+5.1
FOLIO	45.8	**46.3**	45.8	45.8*	41.4	+4.4
JustLogic	51.1	52.6	45.5	**53.5***	46.1	+7.4
ProverQA	42.0	44.4	46.3	47.8	**49.2**	−1.4
CLUTRR	25.5	27.2	23.2	**28.6**	26.7	+1.9
LogicBench-BQA	**74.2**	72.5	74.1	73.6*	71.2	+2.4
RuleTaker	60.5	60.6	**63.5**	60.0	58.8	+1.2
Web of Lies†	55.5	56.5	57.2	**57.9**	57.8	+0.2
LogiQA	42.6	**44.9**	**44.9**	42.4	43.4	−0.9
NLGraph	41.5	43.5	44.3	43.7	**46.2**	−2.5

5 Conclusion

We investigate whether LLMs can be trained to perform implicit reasoning at greater depths using a curriculum over lookahead and attribute vocabulary size that gradually increases the maximum number of steps the model must search ahead to identify the next step in a proof. We find that curriculum learning enables models to reach greater depths than no-curriculum baselines. Achieved lookahead grows predictably with training compute but decelerates, and across both model families the trajectories are better described by a saturating exponential than by an unbounded power law, suggesting a maximum intrinsic lookahead that varies across models. Our fits suggest that model size and pretrained initialization raise the maximum intrinsic lookahead, while step sizes that are too large stall the curriculum. Finally, the capability acquired on synthetic data can transfer to downstream natural-language reasoning benchmarks, showing that curriculum-trained implicit reasoning can generalize beyond the synthetic training distribution.

Future work includes separating the effects of architecture and pretraining on the fitted asymptotes, applying the curriculum to semantically richer generators with a controllable difficulty parameter, and training curriculum-trained models with reinforcement learning to measure the effect on chain-of-thought length and backtracking.

Reproducibility Statement

Everything needed to reproduce our results is contained in the paper and its appendices. The task formulation and the curriculum over lookahead and attribute vocabulary are specified in Sections 3.1 and 3.2; the training procedure is given as pseudocode in Appendix A, and worked examples of generated training instances appear in Appendix H. Training data is produced with a modified version of the `DeepRD` generator, with our modifications described in Section 3.1. Model configurations, optimizer settings, batch sizes, and learning rates for every experiment are listed in Appendix B. The downstream evaluation protocol, including benchmarks, prompting format, and scoring, is given in Appendices E and G; all models are evaluated under an identical protocol. Confidence intervals and the significance tests behind our downstream comparisons are reported in Appendix F, and the curve-fitting procedure behind our $L(C)$ claims, including held-out validation, is detailed in Appendix D. We note that every downstream number comes from a single training seed, and that the intervals we report reflect evaluation sampling only. We will release our code upon acceptance.

AI Use Statement

We used generative AI tools to assist with writing and editing this manuscript, limited to copy-editing, tightening prose, and checking that the phrasing of claims matched the supporting results. We did not use generative AI tools for any of the other tasks for which ICLR requires disclosure. All AI-assisted text was reviewed and revised by the authors, and every claim it touches was checked against the corresponding table, figure, or experimental result. We take responsibility for the final content of this work.

References

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the globality barrier and inductive scratchpad. In *Advances in Neural Information Processing Systems*, volume 37, pp. 27850–27895, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/3107e4bdb658c79053d7ef59cbc804dd-Abstract-Conference.html`.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48. ACM, 2009. doi: 10.1145/1553374.1553380.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL `https://doi.org/10.1609/aaai.v34i05.6239`.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4082–4102, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.242. URL `https://aclanthology.org/2024.findings-acl.242`.

Dake Bu, Wei Huang, Andi Han, Atsushi Nitanda, Hau-San Wong, Qingfu Zhang, and Taiji Suzuki. Provable sample efficiency of curriculum post-training for transformer reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. doi: 10.48550/arXiv.2511.07372. URL `https://arxiv.org/abs/2511.07372`.

Michael K. Chen, Xikun Zhang, and Dacheng Tao. Justlogic: A comprehensive benchmark for evaluating deductive reasoning in large language models, 2025. URL `https://arxiv.org/abs/2501.14851`.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 2020. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3):253–263, 1995.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332, 2023.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (SoS): Learning to search in language. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=6LNTSrJjBe.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2026. URL https://zenodo.org/records/18636344.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ph04CRkPdC.

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

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Alex Heyman and Joel Zylberberg. Evaluating the systematic reasoning abilities of large language models through graph coloring. *arXiv preprint arXiv:2502.07087*, 2025.

Yixiao Huang, Hanlin Zhu, Zixuan Wang, Jiantao Jiao, Stuart Russell, Somayeh Sojoudi, and Song Mei. Transformers provably learn to internalize chain-of-thought. *arXiv preprint arXiv:2605.28600*, 2026. URL https://arxiv.org/abs/2605.28600.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Erik Jenner, Shreyas Kapur, Vasil Georgiev, Cameron Allen, Scott Emmons, and Stuart Russell. Evidence of learned look-ahead in a chess-playing neural network. In *Advances in Neural Information Processing Systems*, volume 37, pp. 31410–31437, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/37d9f19150fce07bced2a81fc87d47a6-Abstract-Conference.html.

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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2025.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul McVay, Michael Rabbat, and Yuandong Tian. Beyond A*: Better planning with transformers via search dynamics bootstrapping. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=SWYE2eGmYw.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37889–37905. PMLR, 2025a. URL https://proceedings.mlr.press/v267/lin25i.html.

Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. Implicit reasoning in transformers is reasoning through shortcuts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9470–9487, Vienna, Austria, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.493. URL https://aclanthology.org/2025.findings-acl.493/.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3622–3628, 2020.

Maximilian Mordig, Andreas Opedal, Weiyang Liu, and Bernhard Schölkopf. Rethinking easy-to-hard: Limits of curriculum learning in post-training for deductive reasoning. *arXiv preprint arXiv:2603.27226*, 2026. URL https://arxiv.org/abs/2603.27226.

Olmo Team. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reason-

ing evaluation. In *International Conference on Learning Representations*, pp. 54186–54205, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/862819c227b16f9af64dd6ad6cfdf275-Paper-Conference.pdf.

Revanth Rameshkumar, Jimson Huang, Yunxin Sun, Fei Xia, and Abulhair Saparov. Reasoning models reason well, until they don't. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 936–956, Mumbai, India, 2025. Asian Federation of Natural Language Processing and Association for Computational Linguistics. URL https://aclanthology.org/2025.ijcnlp-long.52/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using OOD examples. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3083–3105, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/09425891e393e64b0535194a81ba15b7-Abstract-Conference.html.

Abulhair Saparov, Srushti Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Seyed Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9cQB1Hwrtw.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. URL https://aclanthology.org/2023.findings-acl.824/.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, pp. 38975–38987, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7a92bcdede88c7afd108072faf5485c8-Abstract-Datasets_and_Benchmarks.html.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad217e0c7fecc71bdf48660ad6714b07-Abstract-Conference.html.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zixuan Wang, Eshaan Nichani, Alberto Bietti, Alex Damian, Daniel Hsu, Jason D Lee, and Denny Wu. Learning compositional functions with transformers from easy-to-hard data. In *Proceedings of Thirty Eighth Conference on Learning Theory*, volume 291 of *Proceedings of Machine Learning Research*, pp. 5632–5711. PMLR, 2025. URL https://proceedings.mlr.press/v291/wang25a.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, pp. 56721–56742, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d087955ee13fe9a7402eedec879b9c3-Abstract-Conference.html.

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.

Tong Yang, Yu Huang, Yingbin Liang, and Yuejie Chi. Agentic transformers provably learn to search via reinforcement learning. *arXiv preprint arXiv:2606.00183*, 2026. URL https://arxiv.org/abs/2606.00183.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023.

Yuekun Yao, Yupei Du, Dawei Zhu, Michael Hahn, and Alexander Koller. Language models can learn implicit multi-hop reasoning, but only if they have lots of training data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9684–9702, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.490. URL https://aclanthology.org/2025.emnlp-main.490/.

Jiaran Ye, Zijun Yao, Zhidian Huang, Liangming Pan, Jinxin Liu, Yushi Bai, Amy Xin, Weichuan Liu, Xiaoyin Che, Lei Hou, and Juanzi Li. How do transformers learn implicit reasoning? In *Advances in Neural Information Processing Systems*, volume 38, pp. 65810–65838. Curran Associates, Inc., 2025. doi: 10.52202/085713-2208. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5f1cb1d23261b19cbd45f90f7b4f251f-Abstract-Conference.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Curriculum Training Algorithm

The pseudocode below specifies the curriculum training procedure described in Section 3.2. Stages advance based on a rolling exact-match accuracy criterion over the W most recent training examples, after a per-stage minimum step floor (200 steps, see Appendix B) is reached.

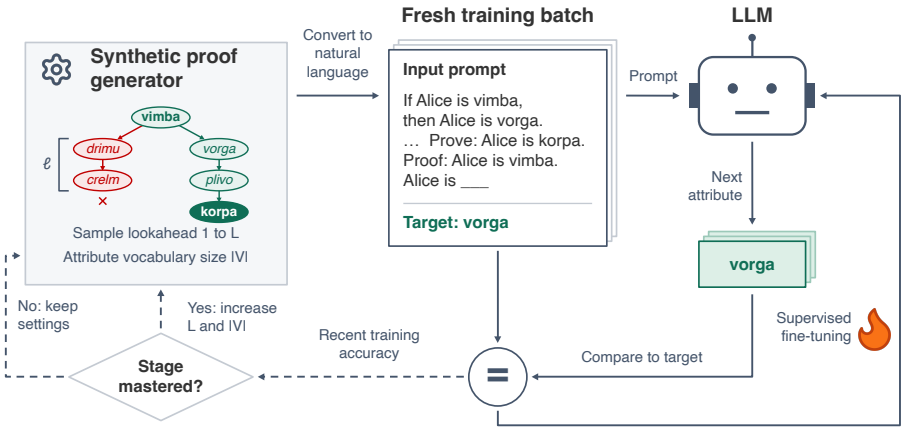

FIGURE 8: Training loop of our curriculum over lookahead and attribute vocabulary, as described in Section 3.2.

Algorithm 1 Curriculum Training over Lookahead and Attribute Vocabulary

Require: Model f_θ with parameters θ, step size s, number of stages K, threshold τ, window W, batch size B, minimum steps per stage M

Initialize θ; set $k \leftarrow 1$
while $k \leq K$ **do**
  $L_k \leftarrow k \cdot s$
  $|\mathcal{V}_k| \leftarrow 2(L_k + 1)$
  Sample $\ell_i \sim \text{Uniform}\{1, \ldots, L_k\}$ independently for $i = 1, \ldots, B$
  Generate batch $\{(x_i, y_i)\}_{i=1}^{B}$ with lookaheads ℓ_i and attribute labels drawn from a vocabulary of size $|\mathcal{V}_k|$, where x_i is the premises and query and y_i is the next proof step
  $\mathcal{L} \leftarrow \frac{1}{B} \sum_{i=1}^{B} \text{CrossEntropy}(f_\theta(x_i), y_i)$ on answer tokens
  Update θ to minimize $\mathcal{L}$
  if at least M steps have been taken in stage k **and** exact-match accuracy over the most recent W examples $\geq \tau$ **then**
    $k \leftarrow k + 1$
  end if
end while
return θ

B Experimental Details

We list the training hyperparameters for each experiment in Section 4. All experiments use AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) with a constant learning rate, no weight decay, gradient clipping at 1.0, and bf16 mixed precision; we find scheduled learning rates consistently degrade or fail to improve over constant LR. All experiments use a fixed random seed of 1234. Unless noted, every experiment uses 100 warmup steps, $\tau = 0.98$ rolling-accuracy advancement threshold over a rolling window of $W = 800$ examples, and a 200-step floor per stage. Training uses predominantly 2–4× NVIDIA H100 80GB GPUs, with 8× NVIDIA H200 used for `Pythia-2.8B` and 6× NVIDIA A100 80GB used for `Pythia-160M` and parts of the `Qwen3-1.7B` downstream training. Gradient accumulation is chosen to match the effective batch size listed per experiment. Downstream evaluation uses a single NVIDIA A100 GPU.

Notation. "Batch size" refers to the combined effective batch size across GPUs. All `Pythia` and `Qwen3` models are initialized from the released checkpoints (`Pythia`: base; `Qwen3`: the post-trained release) unless explicitly marked random init.

B.1 Joint Curriculum vs. Fixed-Vocabulary (Section 4.1)

Both configurations use `Qwen3-0.6B` (full FT) with the same optimization stack as Section 4.1: batch 768, LR 5×10^{-5}, curriculum step $s = 8$ to L=256. They differ only in attribute-vocabulary handling: the joint configuration scales the attribute vocabulary size with stage as $|\mathcal{V}_k| = 2(L_k + 1)$, while the fixed-vocabulary configuration holds $|\mathcal{V}|$ at the maximum from the start. The runs with $|\mathcal{V}_k| = bL_k + 2$ for $b \in \{3, 4\}$ use the same settings.

B.2 Curriculum vs. No-Curriculum (Section 4.1)

Table 5: Hyperparameters for Section 4.1 (`Qwen3-0.6B`).

Setting	Curriculum	No-curriculum
Model	`Qwen3-0.6B` (full FT)	`Qwen3-0.6B` (full FT)
Batch size	768	768
Learning rate	5×10^{-5}	5×10^{-5}
Curriculum step s	8	N/A
L	$8 \rightarrow 256$	$\{96, 104, 112, 120, 128, 256\}$

B.3 Scaling Behavior (Section 4.2)

`Qwen3-0.6B` uses batch 768 and LR 5×10^{-5} with $s = 1$ and a target of L=256; `Pythia-1.4B` uses the per-model settings in Table 6 with $s = 1$ and a target of L=96.

B.4 Curriculum Step Size Sweep (Section 4.3)

All six step sizes share the same `Qwen3-0.6B` optimization stack: batch 768, LR 5×10^{-5}, curriculum to L=256.

B.5 Model Size Sweep (Section 4.4)

Batch size and learning rate are tuned per model size by a small grid search; for each model, we select the (batch size, learning rate) pair that produces the fastest curriculum advancement during early stages.

Table 6: Per-model hyperparameters for Section 4.4 (`Pythia` model-size sweep). All sizes: s=1, AdamW, constant LR.

Model	Batch size	LR
`Pythia-160M`	192	1×10^{-4}
`Pythia-410M`	192	5×10^{-5}
`Pythia-1.4B`	192	2.7×10^{-5}
`Pythia-2.8B`	1,536	2.7×10^{-5}

B.6 Initialization (Section 4.5)

Random initialization makes little progress under the small learning rate suited to fine-tuning, so each condition uses its own best-found learning rate and batch size.

TABLE 7: Hyperparameters for Section 4.5 (s=1; curriculum to L=96 for `Pythia-1.4B` and to L=256 for `Qwen3-0.6B`). The `Qwen3-0.6B` pretrained row reuses the training from Section 4.2.

Condition	Batch size	LR	Init
`Pythia-1.4B`, pretrained	192	2.7×10^{-5}	`Pythia-1.4B` base
`Pythia-1.4B`, random init	768	1×10^{-4}	default initializer
`Qwen3-0.6B`, pretrained	768	5×10^{-5}	`Qwen3-0.6B` post-trained release
`Qwen3-0.6B`, random init	1,536	3×10^{-4}	default initializer

B.7 DOWNSTREAM REASONING TRANSFER (SECTION 4.6)

`Qwen3-1.7B` uses a 6% `Dolci-Instruct-SFT` mix-in alongside the graph-search curriculum; the matched-compute baseline trains the same model on the same mix without curriculum, directly at L=48. The instruction-only control trains the same model with the same optimizer settings on 100% `Dolci-Instruct-SFT`, with no graph-search data. All three configurations wrap inputs in a chat template with `enable_thinking=False`.

TABLE 8: Hyperparameters for Section 4.6. All: AdamW, constant LR 5×10^{-5}, batch 768, chat template with `enable_thinking=False`. The curriculum uses s=1 up to L=48.

Configuration	Mix
`Qwen3-1.7B`, curriculum	6%
`Qwen3-1.7B`, no curriculum (direct at L=48)	6%
`Qwen3-1.7B`, instruction only	100%

C DETAILED EXPERIMENTAL RESULTS

This appendix tabulates the core experiments of Section 4, with one subsection per experiment.

C.1 FULL PER-L CURRICULUM VS. NO-CURRICULUM SWEEP (SECTION 4.1)

Figure 9 shows training-compute-vs-accuracy curves for the complete sweep of target lookaheads $L \in \{96, 104, 112, 120, 128, 256\}$ from Section 4.1. The body of the paper shows three representative panels (L=96, 128, 256) for compactness; the additional intermediate values (L=104, 112, 120) included here exhibit the same qualitative pattern: training with curriculum reaches the 98% threshold 1.25–2.27× faster than training without curriculum at every $L \leq 128$ (Table 1).

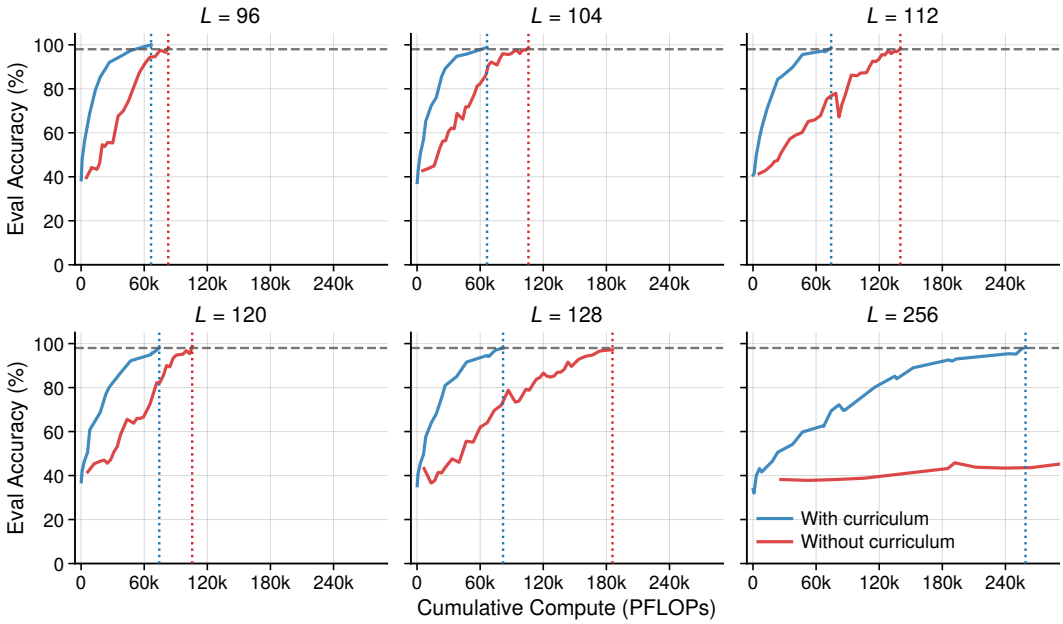

FIGURE 9: Same layout as Figure 2, for the full sweep $L \in \{96, 104, 112, 120, 128, 256\}$.

C.2 JOINT CURRICULUM VS. FIXED ATTRIBUTE VOCABULARY (SECTION 4.1)

Table 9 reports, for the joint curriculum and for the lookahead-only curriculum with a fixed attribute vocabulary (Section 4.1), the cumulative compute at which each configuration completes stage L, and the stage completed at the shared budget.

Condition	8	16	24	32	48	64	80	96	112	128	L at 100K
Joint curriculum, $\lvert\mathcal{V}_k\rvert = 2(L_k + 1)$	144	983	3.14K	6.19K	13.6K	23.5K	37.7K	67.3K	74.3K	87.5K	**128**
Lookahead only, fixed $\lvert\mathcal{V}\rvert$	401	2.54K	4.30K	7.66K	20.6K	43.7K	61.6K	88.8K	not reached	not reached	96

TABLE 9: Cumulative training compute (PFLOPs) at which each configuration completes stage L for `Qwen3-0.6B` with s=8, and the stage completed within a shared budget of 100K PFLOPs. The fixed-vocabulary configuration ends with L=96 completed after 107K PFLOPs. PFLOPs are computed as $6N$ per token with actual parameter counts.

C.3 MODEL SIZE (SECTION 4.4)

Table 10 reports, for the four `Pythia` sizes trained under the curriculum with s=1, the achieved lookahead L at the end of training and the cumulative training compute at which that stage is completed. Table 11 reports the stage completed at fixed compute budgets. Both use the convention of Section 4 (the reported L is the last stage completed).

Model	Achieved lookahead L	Completed at (PFLOPs)
`Pythia-160M`	9	921K
`Pythia-410M`	46	1.72M
`Pythia-1.4B`	75	4.97M
`Pythia-2.8B`	133	2.55M

TABLE 10: Achieved lookahead L at the end of training for each `Pythia` size under the curriculum with s=1, and the cumulative training compute (PFLOPs, computed as $6N$ per token) at which that stage is completed.

	L completed at cumulative PFLOPs						End of training		
Model	100K	200K	300K	500K	750K	1M	L	PFLOPs	Tokens
`Pythia-160M`	7	7	8	8	8	9	9	995,662	1,022B
`Pythia-410M`	30	34	37	40	42	44	46	2,017,056	828B
`Pythia-1.4B`	34	41	46	52	56	58	75	5,000,829	589B
`Pythia-2.8B`	31	46	60	73	84	94	133	2,561,267	154B

TABLE 11: Curriculum stage completed by each `Pythia` size (s=1) at fixed cumulative training compute and at the end of training. `Pythia-410M` leads `Pythia-1.4B` only below ∼30K PFLOPs, and `Pythia-2.8B` overtakes `Pythia-1.4B` between 100K and 200K PFLOPs. `Pythia-2.8B` completes L=130 at 2.40M PFLOPs and L=133 at 2.55M PFLOPs; `Pythia-1.4B` completes L=75 at 4.97M PFLOPs.

Table 12 reports the cumulative compute at which each `Pythia` size completes stage L.

Model	8	16	24	32	48	64	96	128
`Pythia-160M`	253K	not reached	not reached	not reached	not reached	not reached	not reached	not reached
`Pythia-410M`	748	7.64K	34.3K	134K	not reached	not reached	not reached	not reached
`Pythia-1.4B`	1.07K	7.75K	28.2K	73.6K	385K	1.65M	not reached	not reached
`Pythia-2.8B`	2.62K	15.2K	33.6K	102K	212K	347K	1.04M	2.30M

TABLE 12: Cumulative training compute (PFLOPs) at which each `Pythia` model completes stage L under the curriculum with s=1. Training ends with L=9 (160M, 996K PFLOPs), 46 (410M, 2.02M), 75 (1.4B, 5.00M) and 133 (2.8B, 2.56M) completed. PFLOPs are computed as $6N$ per token with actual parameter counts.

Table 13 reports the same stage completions in training tokens.

Model	8	16	24	32	48	64	96	128
`Pythia-160M`	260	not reached	not reached	not reached	not reached	not reached	not reached	not reached
`Pythia-410M`	0.31	3.1	14.1	55.1	not reached	not reached	not reached	not reached
`Pythia-1.4B`	0.13	0.91	3.3	8.7	45.4	194.8	not reached	not reached
`Pythia-2.8B`	0.16	0.91	2.0	6.2	12.7	20.9	62.5	138.4

TABLE 13: Training tokens (billions) at which each `Pythia` model completes stage L under the curriculum with s=1; the same stage completions as Table 12, in tokens rather than PFLOPs.

C.4 INITIALIZATION (SECTION 4.5)

Table 14 reports, for `Pythia-1.4B` and `Qwen3-0.6B` initialized from the released checkpoint or from random weights (Section 4.5), the cumulative compute at which each condition completes stage L and the stage completed at two budgets.

Model	Initialization	8	16	24	32	48	64	96	128	192	244	L at 300K	L at 1.5M
`Pythia-1.4B`	pretrained	1.07K	7.75K	28.2K	73.6K	385K	1.65M	not reached	not reached	not reached	not reached	46	**63**
`Pythia-1.4B`	random	79.7K	636K	1.64M	4.41M	not reached	not reached	not reached	not reached	not reached	not reached	13	23
`Qwen3-0.6B`	pretrained	618	2.35K	5.78K	10.8K	25.4K	48.0K	121K	228K	628K	1.49M	145	**244**
`Qwen3-0.6B`	random	74.8K	285K	618K	1.36M	2.22M	not reached	not reached	not reached	not reached	not reached	16	35

TABLE 14: Cumulative training compute (PFLOPs) at which each condition completes stage L under the curriculum with s=1, and the stage completed at 300K and 1.5M PFLOPs. Training from random initialization ends with L=35 completed (`Pythia-1.4B`, 6.02M PFLOPs) and L=61 completed (`Qwen3-0.6B`, 4.82M PFLOPs). PFLOPs are computed as $6N$ per token with actual parameter counts.

D FUNCTIONAL FORMS FOR ACHIEVED LOOKAHEAD

We fit achieved lookahead L as a function of cumulative training compute C for the two trajectories of Section 4.2, `Qwen3-0.6B` and `Pythia-1.4B`, and for the less trained `Pythia-410M` and `Pythia-2.8B`, using the power law and saturating exponential defined there. This appendix reports the full comparison (RMSE with confidence intervals, AICc, and held-out prediction of both cumulative lookahead and per-stage training iterations), along with the per-size saturating exponential fits underlying the ceiling analysis of Section 4.2 (Figure 10).

Data and procedure. Stage completions (one point per stage, $L \in [1, L_{\text{end}}]$, at the compute where the 98% gate is passed) yield $n = 244$ points for `Qwen3-0.6B` and $n = 46$, 75 and 133 for `Pythia-410M`, `-1.4B` and `-2.8B`. We fit each form with SciPy `differential_evolution` (Sobol init, popsize=30, maxiter=200, seed=42) polished by Levenberg–Marquardt, retaining the lowest-RSS solution. Confidence intervals on RMSE are 95% moving-block bootstrap intervals (block length $\approx n^{1/3}$, 300–500 resamples), which partly account for the serial correlation of a single trajectory; with residual autocorrelation persisting over many lags they should be read as descriptive, not as calibrated 95% intervals. ΔAICc treats the stage points as independent, which they are not (residual autocorrelation exceeds 0.8 at lag 1), so we read it as a ranking of in-sample fit rather than as a calibrated test; the formal comparison is the held-out test on per-stage training iterations in Appendix D.1. Because the power-law offset is at its bound $L_0 = 0$ in every fit and the saturating exponential approaches aC^k as C_0 grows, the in-sample comparison is one-sided by construction and only the size of the RMSE gap is informative. Held-out comparisons fit both forms on the first 70% of the stage points, ordered by compute, and evaluate their predictions on the remainder (Table 15).

	Held-out RMSE			
Trajectory	Power law	Sat. exp.	Ratio	p
`Qwen3-0.6B`	23.88	17.08	1.40	7×10^{-10}
`Pythia-410M`	8.23	0.84	9.80	0.018
`Pythia-1.4B`	16.14	0.68	23.77	1.5×10^{-4}
`Pythia-2.8B`	9.25	3.72	2.49	0.65

TABLE 15: Held-out prediction error for achieved lookahead L on all four s=1 trajectories, fitting on the first 70% of stage completions and predicting the remainder. Ratio is power-law over saturating-exponential RMSE. p: significance of the saturating exponential's advantage under the per-stage iteration test of Appendix D.1, two-sided.

Trajectory	n	L_{end}	Power-law RMSE [95% CI]	Saturating exponential RMSE [95% CI]	ΔAICc
`Qwen3-0.6B`	244	244	6.10 [3.27, 6.75]	2.22 [1.57, 2.35]	493
`Pythia-410M`	46	46	1.93 [0.71, 2.05]	0.47 [0.32, 0.53]	130
`Pythia-1.4B`	75	75	2.98 [1.69, 3.24]	0.53 [0.40, 0.57]	259
`Pythia-2.8B`	133	133	2.67 [1.92, 3.16]	2.15 [1.46, 2.54]	57

TABLE 16: Fit comparison for the four long s=1 trajectories. ΔAICc is (power-law − saturating exponential); positive favors the saturating exponential, and since both forms have three parameters it reflects fit quality alone. RMSE confidence intervals are 95% moving-block bootstrap. Both in-sample criteria favor the saturating exponential on every trajectory. Held-out prediction is compared in Table 15 and Appendix D.1.

Trajectory	$L_{\max}$	C_0 (PFLOPs)	k	$L_{\max}$ 95% CI
`Qwen3-0.6B`	323.6	7.7×10^5	0.547	[304, 404]
`Pythia-410M`	51.2	1.4×10^5	0.341	[49.2, 56.9]
`Pythia-1.4B`	81.9	5.0×10^5	0.371	[77.8, 83.0]
`Pythia-2.8B`	235.1	3.8×10^6	0.497	[175, 315]

TABLE 17: Fitted saturating exponential parameters, $L = L_{\max}\big(1 - \exp(-(C/C_0)^k)\big)$, for the four long s=1 trajectories. $L_{\max}$ is the fitted ceiling plotted in Figure 10. The last column gives 95% moving-block bootstrap percentile intervals for $L_{\max}$ (block length $\approx n^{1/3}$, 300 resamples). Their lower ends are stable under longer blocks, while the upper ends for `Qwen3-0.6B` and `Pythia-2.8B` widen substantially, so those upper bounds are indicative only.

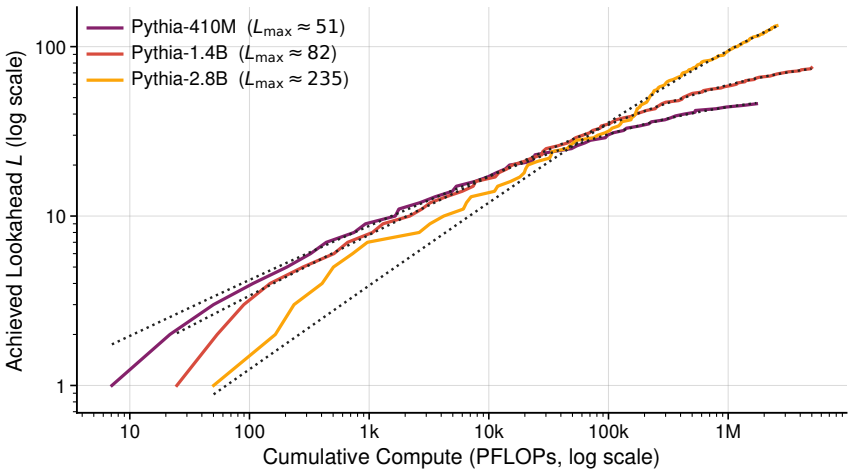

FIGURE 10: Saturating exponential fits (black dotted, over the observed range) to the achieved-lookahead trajectories of `Pythia-410M`, `-1.4B`, and `-2.8B` trained with s=1 (log–log). Legends give each fitted ceiling $L_{\max}$ (Table 17).

What the comparison shows. On every trajectory the saturating exponential attains lower RMSE, is preferred by AICc (ΔAICc $= 57$–493; Table 16) and predicts the held-out stages with lower RMSE (Table 15). The in-sample gap is largest on the three trajectories nearest their fitted ceilings and small on `Pythia-2.8B` (RMSE 2.67 vs. 2.15), which sits at only $\sim$57% of its fitted ceiling and whose RMSE intervals overlap; the two forms separate more cleanly as a model nears its ceiling. Appendix D.1 reports a significance test on per-stage training iterations. Figure 10 plots the per-size saturating exponential fits (parameters in Table 17); the resulting ceilings ($L_{\max} \approx 51$, 82, 235) underlie the ceiling analysis in Section 4.2. The bootstrap intervals in Table 17 show that the lower bound of each ceiling is well determined, while the upper bound is poorly determined for the two trajectories furthest from saturation, `Qwen3-0.6B` and `Pythia-2.8B`. `Pythia-160M` plateaus at L=9 (Section 4.4) and is excluded, since neither form is meaningful for a capacity-limited trajectory.

D.1 HELD-OUT PREDICTION OF PER-STAGE TRAINING ITERATIONS

A Diebold–Mariano test on the held-out residuals of $L(C)$ is not valid for these trajectories. Its variance estimate assumes a weakly dependent loss differential, whereas the held-out residuals along a single trajectory share one extrapolation error and are strongly autocorrelated (lag-1 autocorrelation 0.95 for `Qwen3-0.6B` and 0.92 for `Pythia-2.8B`; Table 18). We therefore compare the two forms on a quantity whose residuals are approximately independent, the number of training iterations required to complete each stage, $\Delta I(L) = I(L) - I(L-1)$ with $I(0) = 0$, where $I(L)$ is the cumulative number of optimizer steps at the completion of stage L. We use iterations rather than compute because the compute cost of a stage is its iteration count multiplied by the tokens per step, and tokens per step grow with L as the sequence length grows, so per-stage iterations isolate the learning dynamics from that growth. Each form implies a per-stage iteration count $g(L) - g(L-1)$, where g is the inverse of the form fitted with cumulative iterations as the independent variable. We fit both forms to $\log \Delta I(L)$ by least squares on the first 70% of stages, ordered by compute, and predict the remaining stages. Table 18 shows that on per-stage iterations the residuals of the saturating exponential are consistent with independence on three trajectories and marginal on `Pythia-1.4B` ($p = 0.046$).

	Cumulative lookahead $L(C)$		Per-stage iterations $\Delta I(L)$	
Trajectory	lag-1 autocorr.	Ljung–Box p	lag-1 autocorr.	Ljung–Box p
`Qwen3-0.6B`	0.95	2×10^{-199}	-0.14	0.15
`Pythia-410M`	0.25	0.36	-0.28	0.36
`Pythia-1.4B`	0.33	0.08	-0.33	0.046
`Pythia-2.8B`	0.92	4×10^{-94}	-0.10	0.13

TABLE 18: Residual dependence of the saturating exponential fit on cumulative lookahead and on per-stage training iterations (log units). Ljung–Box tests use five lags. Per-stage iterations remove the serial dependence that invalidates a Diebold–Mariano test on cumulative lookahead.

For each held-out stage we form the loss differential $d_L = e_{\mathrm{PL}}(L)^2 - e_{\mathrm{SE}}(L)^2$ in log units, which is positive when the saturating exponential predicts that stage better. Because the advantage may grow with depth, we regress d_L on the centered stage index with Newey–West standard errors (Bartlett kernel, bandwidth $\lfloor 4(T/100)^{2/9} \rfloor$) and report the intercept α, which equals the mean of d_L, and the slope β (Table 19). All p-values are two-sided and are stable for bandwidths from 0 to four times this rule. The saturating exponential has lower held-out RMSE on per-stage iterations for three trajectories and ties on `Pythia-2.8B`. The advantage is significant for `Qwen3-0.6B`, `Pythia-410M` and `Pythia-1.4B`, and on all three it grows with depth ($\beta > 0$). Across train/held-out splits from 50% to 80% of the stages, α is significant at every split for `Qwen3-0.6B` and `Pythia-410M`, at the 70% split only for `Pythia-1.4B`, and at no split for `Pythia-2.8B`. The `Pythia-1.4B` result is sensitive to the split because that trajectory's per-stage costs vary by two orders of magnitude between adjacent stages, so a held-out set of 8 to 24 stages is dominated by a few of them. For `Pythia-2.8B` the two forms predict per-stage iterations identically, because the saturating exponential fitted to that trajectory's iterations has no identified ceiling and reduces to the power law. Per-stage iterations are a noisy quantity, with held-out RMSE near 1 in log units for both forms, so this test has less power than the comparison on cumulative lookahead but rests on assumptions the data satisfy.

		Held-out RMSE ($\log \Delta I$)			Mean advantage α		Trend β (per stage)	
Trajectory	T	Sat. exp.	Power law	Ratio	Estimate [95% CI]	p	Estimate [95% CI]	p
`Qwen3-0.6B`	73	0.93	0.96	1.04	$0.068\ [0.049, 0.087]$	7×10^{-10}	$0.0025\ [0.0016, 0.0034]$	1×10^{-6}
`Pythia-410M`	14	1.14	1.90	1.67	$2.33\ [0.47, 4.18]$	0.018	$0.51\ [0.08, 0.95]$	0.025
`Pythia-1.4B`	23	1.06	1.76	1.66	$1.97\ [1.08, 2.86]$	1.5×10^{-4}	$0.18\ [0.05, 0.32]$	0.010
`Pythia-2.8B`	40	1.13	1.13	1.00	$6\times10^{-6}\ [-2\times10^{-5}, 3\times10^{-5}]$	0.65	$-9\times10^{-7}\ [-3\times10^{-6}, 1\times10^{-6}]$	0.37

TABLE 19: Held-out prediction of per-stage training iterations. T is the number of held-out stages. α and β are the intercept and slope of d_L regressed on the centered stage index, with Newey–West standard errors; $\alpha > 0$ favors the saturating exponential and $\beta > 0$ indicates an advantage that grows with depth.

D.2 COMPUTE PER CURRICULUM STAGE

Figure 11 shows the mean compute per completed curriculum stage for `Pythia-1.4B` with $s = 1$, averaged over consecutive groups of five stages. The last five completed stages, L=71–75, consume 2.10M PFLOPs, approximately 42% of its 5M-PFLOP budget. Thus, nearly half the training compute produces only five additional stages of progress, illustrating the sharply diminishing returns to further training in this regime. Within this budget, the trajectory reaches L=75, 91% of the fitted asymptote $L_{\max} \approx 82$, without having plateaued. The flattening captured by the saturating exponential is therefore accompanied by a substantial observed increase in the compute required for further curriculum progress.

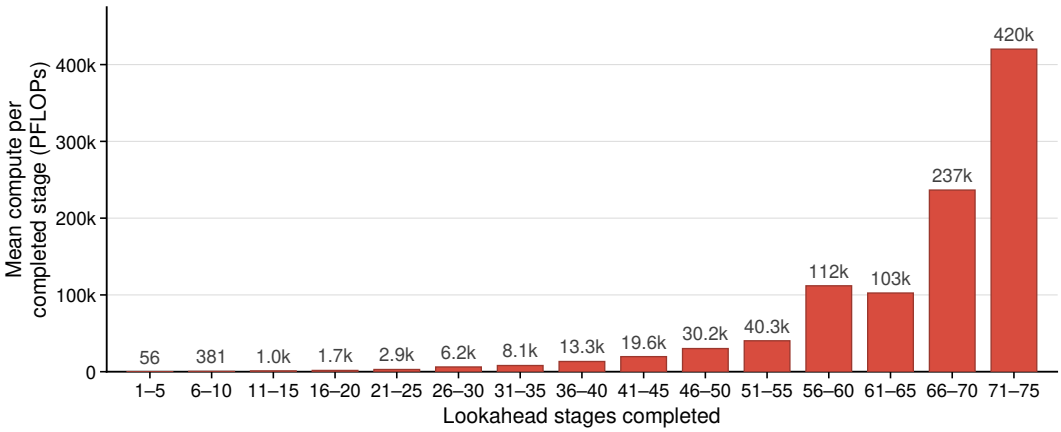

FIGURE 11: Mean training compute per completed curriculum stage for `Pythia-1.4B` with s=1. Each bar averages five consecutive completed stages; labels show the mean in PFLOPs. Compute for a stage is measured between consecutive stage completions, with the first stage measured from the start of training. The unfinished stage L=76 is excluded.

E DOWNSTREAM EVALUATION METHODOLOGY

Next-step proof prediction. On the synthetic next-step prediction task (Section 3.1) we use greedy autoregressive decoding, and the first whitespace- or punctuation-delimited word of the generated continuation is compared case-insensitively with any accepted gold attribute.

We evaluate on 14 reasoning benchmarks (full list in Table 4). Thirteen are scored by log-likelihood over a fixed candidate label set; the open-ended `NLGraph` benchmark uses its authors' per-task scorers. Samples are drawn with a fixed random seed, and the number of items per benchmark is listed in Table 23.

Why constrained log-likelihood rather than CoT. Several of these benchmarks are designed for chain-of-thought or free-form decoding. Our curriculum-trained models do not produce verbose CoT natively. The curriculum task contains no signal for natural-language reasoning style, so unconstrained free-form decoding would entangle output style with underlying capability. For benchmarks with a fixed answer set we therefore use log-likelihood scoring over the candidate labels, isolating reasoning from format-following. This is the same convention used by `lm-evaluation-harness` (Gao et al., 2026) for its `loglikelihood` request type.

Fused single-pass scoring. For each example we run the model once on the prompt and compute the per-token log-probabilities of every candidate label. When a label tokenizes to a single token (the common case for True/False/Unknown, multiple-choice letters, etc.) we read its log-prob directly from the next-token distribution. For multi-token labels we fall back to teacher-forced log-likelihood with the same approach as `lm-evaluation-harness`: walk forward to the first divergent token between $\mathrm{tokenize}(\text{prefix})$ and $\mathrm{tokenize}(\text{prefix} + \text{label})$, then sum log-probs of the suffix tokens. This handles BPE boundary merges correctly and remains a single forward pass per choice.

Reported metric. The headline accuracy reported in Table 4 is log-likelihood accuracy: argmax over candidate labels by total log-probability. This matches the dominant convention for log-likelihood-scored benchmarks in the literature.

Per-benchmark prompting protocol. For each benchmark we adapt its canonical LLM evaluation protocol to a constrained-log-likelihood setting: we keep the canonical demonstration count and prompt format where compatible, and substitute Direct prompting where the canonical convention uses chain-of-thought. The same protocol is applied to the base model, every curriculum checkpoint, the no-curriculum baseline, and the instruction-only control for each benchmark. Per-benchmark choices are summarized in Table 20.

Benchmark	Canonical LLM protocol	Our protocol
Reasoning benchmarks		
ProofWriter OWA	none	0-shot Direct
ProofWriter CWA	none	0-shot Direct
ParaRules	none	0-shot Direct
ZebraLogic-MC	none	3-shot Direct
PrOntoQA	8-shot CoT (Saparov et al., 2023)	8-shot Direct
FOLIO	8-shot (Han et al., 2024)	8-shot Direct
JustLogic	3-shot (Chen et al., 2025)	3-shot Direct
ProverQA	2-shot (Qi et al., 2025)	2-shot Direct
CLUTRR	none	5-shot Direct
LogicBench-BQA	0-shot CoT (Parmar et al., 2024)	3-shot Direct
RuleTaker	none	5-shot Direct
Web of Lies	3-shot (Suzgun et al., 2023)	3-shot Direct
LogiQA	0-shot (Zhong et al., 2023)	0-shot Direct
NLGraph	1–5-shot (Wang et al., 2023a)	1–5-shot, generation
General-capability benchmarks		
MMLU	0-shot (Gao et al., 2026)	0-shot
IFEval	0-shot (Zhou et al., 2023)	0-shot, generation
ARC-E	none	0-shot
ARC-C	none	0-shot
HellaSwag	none	0-shot
WinoGrande	none	0-shot
PIQA	none	0-shot
BoolQ	none	0-shot
LAMBADA	none	0-shot

TABLE 20: Per-benchmark prompting protocol. Where a benchmark has a canonical zero-shot protocol, we keep it; where it has a canonical few-shot protocol, we use its shot count; where the canonical protocol uses chain-of-thought or no LLM protocol exists, we use Direct prompting. All demonstrations show the question and label only and are removed from the scored items. The seven general-capability benchmarks without a canonical protocol use the `lm-evaluation-harness` default of zero-shot.

Sampling and stratification. Item counts per benchmark are listed in Table 23. For benchmarks with subsets (per-depth, per-task, or per-puzzle-size), we draw a fixed-seed random sample within each subset rather than a deterministic prefix. `ProofWriter OWA` and `ParaRules` are label-balanced within each depth; `PrOntoQA` and `Web of Lies` use 500 items per depth; `NLGraph` uses 125 items per sub-task.

NLGraph (generation). `NLGraph` contains 8 sub-tasks (shortest path, Hamilton, connectivity, etc.) and the canonical scoring is per-task: path validation for Hamilton, weight match for shortest path, regex-based answer extraction for others. We use the authors' canonical `k-shot-prompt.txt` files (with task-dependent k, no CoT) and their per-task scorers ported from the official repo. The accuracy we report on `NLGraph` is the average of these per-task scores.

F CONFIDENCE INTERVALS ON DOWNSTREAM RESULTS

Table 21 reports the accuracies of Table 4 with 95% confidence intervals, together with the base model and the instruction-only control, and marks every curriculum checkpoint that is significantly above training without curriculum, not only the best one per benchmark. The base model and the instruction-only control are not part of the matched comparison, and they score above curriculum $L{=}48$ on four and six of the 14 benchmarks, respectively.

TABLE 21: Downstream reasoning accuracy (%) of `Qwen3-1.7B` from Table 4, with 95% confidence intervals, where $\pm$ is the half-width $1.96\sqrt{p(1-p)/n}$ for accuracy p and the per-benchmark sample size n in Table 23. Base is the untrained `Qwen3-1.7B` under the same prompts, with the answer cue prefilled in its own markdown answer format. Instruction-only is trained on the instruction data alone at compute matched to curriculum L=48. *: curriculum checkpoint significantly above no curriculum (two-sided exact McNemar test on paired items, $p < 0.05$). †Extended by us to greater depths (Appendix G).

Benchmark	Base	Curriculum L				No curr.	Instr. only
		8	16	32	48		
ProofWriter OWA	40.3±1.8	41.1±1.8	33.6±1.7	39.3±1.7	54.9±1.8*	40.6±1.8	47.8±1.8
ProofWriter CWA	52.7±1.8	66.8±1.7*	57.2±1.8*	67.6±1.7*	67.9±1.7*	55.2±1.8	58.0±1.8
ParaRules	43.8±1.4	46.1±1.4	36.2±1.4	39.1±1.4	55.1±1.4*	50.4±1.5	57.1±1.4
ZebraLogic-MC	25.5±2.7	47.4±3.1	53.4±3.1*	51.8±3.1*	53.5±3.1*	48.3±3.1	53.2±3.1
PrOntoQA†	53.1±1.5	52.5±1.5	55.3±1.5*	56.8±1.5*	57.1±1.5*	52.0±1.5	52.2±1.5
FOLIO	45.3±6.8	45.8±6.9*	46.3±6.9*	45.8±6.9*	45.8±6.9*	41.4±6.8	45.3±6.8
JustLogic	60.9±1.4	51.1±1.4*	52.6±1.4*	45.5±1.4	53.5±1.4*	46.1±1.4	55.7±1.4
ProverQA	48.0±2.5	42.0±2.5	44.4±2.5	46.3±2.5	47.8±2.5	49.2±2.5	52.0±2.5
CLUTRR	28.6±2.8	25.5±2.7	27.2±2.8	23.2±2.6	28.6±2.8	26.7±2.7	22.3±2.6
LogicBench-BQA	73.9±2.2	74.2±2.2*	72.5±2.3	74.1±2.2*	73.6±2.2*	71.2±2.3	69.5±2.3
RuleTaker	67.8±2.9	60.5±3.0	60.6±3.0	63.5±3.0*	60.0±3.0	58.8±3.1	70.4±2.8
Web of Lies†	53.0±1.5	55.5±1.5	56.5±1.5	57.2±1.5	57.9±1.5	57.8±1.5	58.8±1.5
LogiQA	39.4±3.8	42.6±3.8	44.9±3.8	44.9±3.8	42.4±3.8	43.4±3.8	43.2±3.8
NLGraph	35.8±3.0	41.5±3.1	43.5±3.1	44.3±3.1	43.7±3.1	46.2±3.1	39.5±3.0

F.1 GENERAL MODEL CAPABILITY

Table 22 reports the accuracies summarized in Section 4.6. The benchmarks are `MMLU`, `ARC-Easy` and `ARC-Challenge`, `HellaSwag`, `WinoGrande`, `PIQA`, `BoolQ`, `LAMBADA` and `IFEval`. Each is evaluated on the non-finetuned `Qwen3-1.7B`, on an instruction-only control, on the model trained without curriculum, and on four curriculum stage checkpoints. The first eight benchmarks probe knowledge and single-step commonsense rather than multi-step deductive chaining, so we do not expect the curriculum to improve them.

Because the curriculum trains the model almost exclusively on synthetic proof data, it could plausibly narrow the capabilities of the model in other respects. We evaluate `Qwen3-1.7B` on 8 standard benchmarks spanning world knowledge (`MMLU`), science QA (`ARC-Easy/Challenge`), commonsense inference (`HellaSwag`, `WinoGrande`, `PIQA`), reading comprehension (`BoolQ`) and discourse-level language modeling (`LAMBADA`), plus `IFEval` for instruction following, using an identical `lm-evaluation-harness` protocol for every model evaluated. Mean accuracy over the eight is above the non-finetuned model's at every curriculum stage (63.4 to 64.2 vs. 61.7; Table 22). The largest single-benchmark drop below it is `WinoGrande` at L=48 (59.8 vs. 61.7). `IFEval` is also above the non-finetuned model at every stage (34.6 to 41.2 vs. 33.1). Because our training mixture also contains instruction data, a comparison against the non-finetuned model alone would leave open the possibility that this instruction data offsets damage done by the synthetic data. We therefore fine-tune the same base model on 100% `Dolci-Instruct-SFT`, with no synthetic data and no curriculum but otherwise identical hyperparameters, and evaluate it at cumulative training compute matched to the L=48 curriculum checkpoint. It scores within one point of that checkpoint on the eight-benchmark mean (64.2 vs. 63.4), so training on the synthetic data adds little measurable cost on these benchmarks. The model trained without curriculum on the same data at the same compute scores essentially the same as the curriculum (63.2), so the curriculum itself adds no further cost. On `IFEval`, the L=48 checkpoint scores below the instruction-only control (41.2 vs. 46.4), which trains on instruction data alone, and slightly above the model trained without curriculum (40.1). Thus, we find no evidence that gains in implicit reasoning capability are at the expense of general model capability.

TABLE 22: General model capability of `Qwen3-1.7B` across curriculum stages. Accuracy on eight standard benchmarks (zero-shot, `lm-evaluation-harness` v0.4.11, identical settings for every model) for the non-finetuned model, four curriculum stage checkpoints, the model trained without curriculum, and an instruction-only control, the last two compute-matched to the L=48 checkpoint. "Mean" is the unweighted mean of the eight language understanding benchmarks. `IFEval` (prompt-level strict accuracy) is reported separately and is excluded from the mean.

Model	MMLU	ARC-E	ARC-C	HellaSwag	WinoG.	PIQA	BoolQ	LAMBADA	**Mean**	IFEval
Base `Qwen3-1.7B`	55.5	72.6	42.7	60.5	61.7	72.5	77.6	50.3	61.7	33.1
Curriculum, L=8	56.9	76.2	48.0	62.4	63.2	72.8	76.4	57.4	64.2	35.1
Curriculum, L=16	57.1	75.7	45.9	62.5	62.4	73.3	77.1	58.0	64.0	39.0
Curriculum, L=32	57.3	74.3	45.8	62.5	61.6	73.0	77.8	59.2	63.9	34.6
Curriculum, L=48	56.1	74.4	45.6	64.0	59.8	73.3	77.2	56.5	63.4	41.2
No curriculum	57.5	75.5	45.3	62.7	62.9	73.7	73.2	55.1	63.2	40.1
Instruction-only	56.4	75.7	48.3	64.0	60.4	73.3	79.2	56.4	64.2	46.4

G BENCHMARK DESCRIPTIONS

Table 23 summarizes the fourteen downstream reasoning benchmarks and the nine general-capability benchmarks: the reasoning type each targets, the answer format we score, the split or subset we evaluate, and the number of evaluation items n. The prompting protocol for each benchmark is given in Table 20.

Benchmark	Reasoning type	Answer format	Split / subset	n
Reasoning benchmarks				
ProofWriter OWA† (Tafjord et al., 2021)	Synthetic deductive	True/False/Unknown	OWA, depths 0–5, label-balanced	3000
ProofWriter CWA (Tafjord et al., 2021)	Synthetic deductive	True/False	CWA, depths 0–5	3000
ParaRules† (Clark et al., 2020)	Synthetic deductive, paraphrased	True/False/Unknown	ProofWriter OWA NatLang test, depths 0–5, label-balanced	4566
ZebraLogic-MC (Lin et al., 2025a)	Constraint satisfaction	Multiple choice	`mc_mode`	1000
PrOntoQA† (Saparov et al., 2023)	Synthetic deductive	True/False	generated, 8 depths up to 64	4000
FOLIO (Han et al., 2024)	First-order logic	True/False/Unknown	validation	203
JustLogic (Chen et al., 2025)	Knowledge-independent deductive	True/False/Uncertain	depths 1–7	4897
ProverQA (Qi et al., 2025)	First-order logic with distractors	3-way multiple choice	dev, easy/medium/hard	1497
CLUTRR (Sinha et al., 2019)	Inductive kinship	Relation label set	full	1000
LogicBench-BQA (Parmar et al., 2024)	Single-rule deductive	Yes/No	BQA	1512
RuleTaker (Clark et al., 2020)	Synthetic deductive	True/False	test, depths D0–D5	1000
Web of Lies† (Suzgun et al., 2023)	Truth-teller chains	Yes/No	generated, 8 depths up to 64, label-balanced	4000
LogiQA (Liu et al., 2020; Zhong et al., 2023)	Mixed (exam)	4-way multiple choice	AGIEval split	648
NLGraph (Wang et al., 2023a)	Graph reasoning	Open-ended (8 sub-tasks)	full	1000
General-capability benchmarks				
MMLU (Hendrycks et al., 2021)	World knowledge	4-way multiple choice	test	14042
ARC-Easy (Clark et al., 2018)	Science QA	4-way multiple choice	test	2376
ARC-Challenge (Clark et al., 2018)	Science QA	4-way multiple choice	test	1172
HellaSwag (Zellers et al., 2019)	Commonsense inference	4-way completion	validation	10042
WinoGrande (Sakaguchi et al., 2020)	Commonsense inference	2-way completion	validation	1267
PIQA (Bisk et al., 2020)	Physical commonsense	2-way multiple choice	validation	1838
BoolQ (Clark et al., 2019)	Reading comprehension	Yes/No	validation	3270
LAMBADA (Paperno et al., 2016)	Reading comprehension	Last-word prediction	test (OpenAI)	5153
IFEval (Zhou et al., 2023)	Instruction following	Generation, prompt-level strict	full	541

TABLE 23: The fourteen downstream reasoning benchmarks and the nine general-capability benchmarks, with the number of evaluation items n. General-capability benchmarks use the default `lm-evaluation-harness` splits. `ProofWriter` appears twice because its open- and closed-world splits differ in label set. `NLGraph` is scored with the authors' per-task scorers, and we report the mean over its eight sub-tasks (Appendix E). †Modified by us, as described below.

PrOntoQA. The released `PrOntoQA` data stops at 4 to 5 hops, and the shipped generator cannot go further, because it samples a random ontology over 17 concept names and takes a random proof walk through it. We replace the ontology sampler with a chain of exactly $k+1$ concepts, script the walk to the unique k-step proof, and enlarge the vocabulary to 624 names and 274 adjectives. Rendering, inflection, parsing, distractors and label negation use the official code. An independent solver verifies every item for its label, for a proof depth equal to k, for consistency, and for distractor sign. The set covers eight depths, $k \in \{1, 2, 4, 8, 16, 32, 48, 64\}$, 500 items each, balanced on both the label and the query polarity.

ProofWriter OWA and ParaRules. We use the published data but draw equal numbers of True, False and Unknown items within each official `QDep` stratum. Without this, the Unknown share falls from 69% at depth 0 to 3% at depth 5, because an unprovable question inherits the depth of its deepest failing search, so a model that always answers Unknown would score 66% at depth 0 and 4% at depth 5. Balancing fixes chance at 33.3% at every depth. The closed-world split is already near 50/50 at every depth and needs no resampling. We apply the same balancing to `ParaRules`, whose items come from the `ProofWriter` open-world NatLang test split.

Web of Lies. Our `Web of Lies` set is generated from the BBH `web_of_lies` template and its 50-name pool, as truth-value chains of $k \in \{1, 2, 4, 8, 16, 32, 48, 64\}$ speakers with 500 items per depth and exactly balanced Yes/No labels. Our solver reproduces all 250 published BBH items, all of which have k=4.

All other benchmarks are the published data as released. Source repositories: `ProofWriter` from AllenAI `proofwriter-dataset-V2020.12.3`; `FOLIO` from `yale-nlp/FOLIO`; `ZebraLogic` from `WildEval/ZebraLogic`; `RuleTaker` from `tasksource/ruletaker` (fallback `hitachi-nlp/ruletaker`); `CLUTRR` from `kendrivp/CLUTRR_v1_extracted` (fallback `kliang5/CLUTRR_huggingface_dataset`); `LogiQA` from Zhong et al. (2023); `LogicBench` and `NLGraph` from their authors' releases.

H Training Data Examples

We show generated training examples at three lookahead values to illustrate how problem complexity scales across curriculum stages. Each example consists of premises (rendered as conditional predicates), a query naming the start fact and goal, and a proof prefix ending where the model must predict. The model's target output is a single attribute (tokenized as one or more BPE tokens). Within a stage, lookaheads are sampled uniformly from 1 to the cap, so most stage-L instances have lookahead below L; the examples below are chosen with lookahead equal to the cap.

Stage cap L=2, example lookahead $\ell = 2$ (5 attributes).

> Suppose we have the following facts: If someone is itaema, then they are isum. If a person is ugphi, they are itaema. If Krystal is nwazao, then they are asico. If Krystal is ugphi, then they are nwazao. Given that Krystal is ugphi, and we want to prove Krystal is asico.
>
> Proof: Krystal is ugphi. Krystal is _____

Answer: **nwazao**.

Stage cap L=8, example lookahead $\ell = 8$ (17 attributes).

> Suppose we have the following facts: If someone is kfusau, then they are kacown. If a person is aqyqoj, they are adawr. Everyone that is fucboo is nuuht. Everyone that is kacown is upocoo. If a person is yuekl, they are xwohoo. If Steven is buey, then Steven is zacipo. Everyone that is xuilat is wxilvu. Everyone that is ahxbwu is ezhol. If Steven is adawr, then Steven is fucboo. If someone is aqyqoj, then they are ornce. If a person is ecqni, they are ahxbwu. If someone is upocoo, then they are yuekl. If Steven is ezhol, then Steven is xuilat. Everyone that is nuuht is kfusau. If someone is ornce, then they are buey. If Steven is zacipo, then they are ecqni. Given that Steven is aqyqoj, and we want to prove Steven is xwohoo.
>
> Proof: Steven is aqyqoj. Steven is _____

Answer: **adawr**.

Stage cap L=32, example lookahead $\ell = 32$ (65 attributes).

> Suppose we have the following facts: If someone is pyaqau, then they are tsawob. If Jennifer is etuupu, then they are poqgje. If a person is vaorea, they are zfaqa. If Jennifer is dduosi, then they are biisi. Everyone that is saiaqt is heeru. If someone is ussdoi, then they are gidxio. Everyone that is fosin is miaize. If someone is xdilxo, then they are ivgau. Everyone that is azlwoo is inaba. Everyone that is hzueda is azlwoo. If Jennifer is owaosy, then they are ahuvyi. If Jennifer is inaba, then Jennifer is agez. If Jennifer is iklau, then they are pyaqau. If Jennifer is ahuvyi, then Jennifer is afueji. If a person is miaize, they are efeh. If Jennifer is wimeu, then they are taew. If Jennifer is efeh, then Jennifer is saiaqt. Everyone that is ssoene is ruosa. Everyone that is idudz is ezejoi. If someone is hijnua, then they are ughtuf. If Jennifer is biisi, then they are faxids. Everyone that is zguul is dpielu. If someone is elorbi, then they are ryuwqo. If Jennifer is aqico, then Jennifer is oresu. If Jennifer is ruosa, then they are hijnua. If Jennifer is afueji, then they are oxuogj. If Jennifer is itcox, then Jennifer is etuupu. If someone is gidxio, then they are raeesb. If a person is heeru, they are owaosy. If Jennifer is liiz, then they are elorbi. If Jennifer is uyqbop, then Jennifer is ekice. If a person is raeesb, they are uyqbop. If a person is umuf, they are iyeriu. If Jennifer is emaca, then they are ssoene. Everyone that is yoxo is fosin. Everyone that is ekice is zaxbom. If Jennifer is dpielu, then they are obaqka. If someone is apenii, then they are luanx. If someone is faxids, then they are ussdoi. Everyone that is sbool is wimeu. If Jennifer is

xuuur, then they are emaca. If someone is oresu, then they are brapuz. If someone is obaqka, then they are yoxo. If someone is jiiel, then they are idudz. If someone is taew, then they are ufnsih. If a person is apenii, they are yutou. If Jennifer is ufnsih, then Jennifer is aqico. Everyone that is zaxbom is zguul. If someone is ivgau, then they are liiz. If Jennifer is noza, then Jennifer is azbigo. If Jennifer is brapuz, then they are jiiel. If a person is ezejoi, they are hzueda. If Jennifer is ikeyo, then Jennifer is sbool. Everyone that is agez is iklau. If Jennifer is yutou, then they are ikeyo. Everyone that is tsawob is umuf. If a person is luanx, they are xdilxo. If Jennifer is zfaqa, then they are neuklo. If Jennifer is oxuogj, then they are itcox. Everyone that is ughtuf is noza. If someone is ryuwqo, then they are enoxde. Everyone that is enoxde is dduosi. Everyone that is iyeriu is vaorea. If Jennifer is neuklo, then Jennifer is xuuur. Given that Jennifer is apenii, and we want to prove Jennifer is azbigo.

Proof: Jennifer is apenii. Jennifer is _____

Answer: **yutou**.

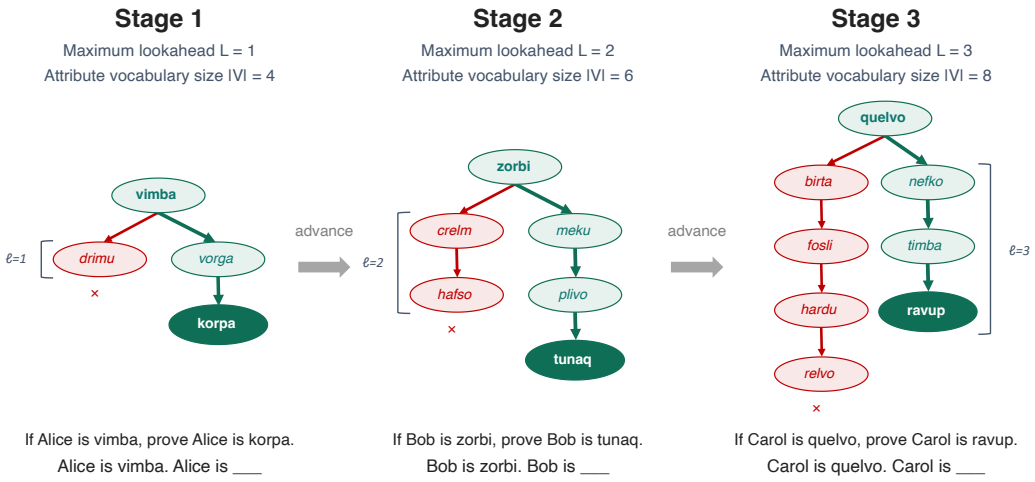

FIGURE 12: Curriculum over lookahead and attribute vocabulary size. The three stages use maximum lookahead $L{=}1, 2, 3$ and attribute vocabulary size $|\mathcal{V}_k| = 4, 6, 8$, respectively. The arrows indicate stage advancement. Each DAG is converted into natural-language premises; for example, the edge `vimba` $\rightarrow$ `vorga` reads "If Alice is vimba, then Alice is vorga." The abbreviated examples below the graphs show the goal and proof prefix; the model predicts the next attribute at the blank. Green marks goal-reaching paths and red marks distractor paths. Neither the colors nor the graph diagrams are part of the model input.