

# Ladder Up, Memory Down: Low-Cost Fine-Tuning With Side Nets

Anonymous ACL submission

## Abstract

Fine-tuning large language models (LLMs) is often limited by the memory available on commodity GPUs. Parameter-efficient fine-tuning (PEFT) methods such as QLoRA reduce the number of trainable parameters, yet still incur high memory usage induced by the backward pass in the full model. We revisit Ladder Side Tuning (LST), a rarely explored PEFT technique that adds a lightweight side network, and show that it matches QLoRA’s compute scaling slope while cutting peak memory by 50%. Across different downstream benchmarks spanning natural language understanding, mathematical and LLM-critic tasks, LST has accuracy close to QLoRA, with a small trade-off, while offering improved memory feasibility. This efficiency enables fine-tuning of 7B-parameter models on a single 12 GB consumer GPU with 2k-token contexts, requiring no gradient checkpointing—conditions under which QLoRA exhausts memory. Beyond memory efficiency, we also establish scaling laws showing that LST has similar scaling slopes to QLoRA. We exploit Ladder’s architectural flexibility by introducing xLadder, a depth-extended variant that increases effective depth via cross-connections and can shorten chain-of-thought (CoT) at fixed parameter count. Upon acceptance, we will open source our implementation [https://anonymous.4open.science/r/arr\\_repo-64AD](https://anonymous.4open.science/r/arr_repo-64AD).

## 1 Introduction

Fine-tuning large language models (LLMs) is crucial for adapting general models to specific tasks, but it is increasingly limited by GPU memory. LLMs with more than 7B parameters (Grattafiori et al., 2024; Qwen et al., 2025; Abdin et al., 2024) strain commodity hardware, forcing smaller batches and shorter contexts. Parameter-efficient fine-tuning (PEFT) reduces the number of trainable parameters with techniques such as adapters (Li and Liang, 2021; Lester et al., 2021; Liu et al.,

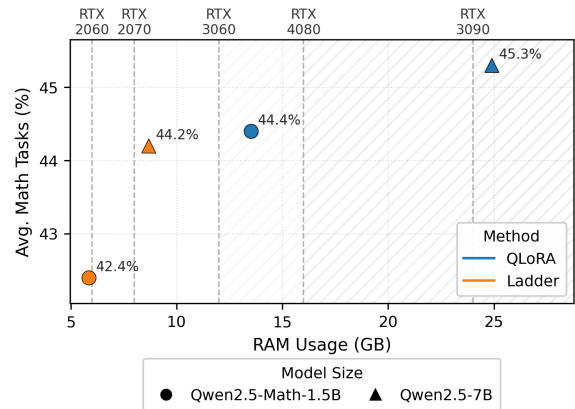


Figure 1: Memory-accuracy frontier on 3 math datasets. All runs use a batch size of 1 and a 2k-token context window. RAM usage is computed with no gradient checkpointing, as detailed in Appendix E. Dashed vertical lines mark the memory budgets of popular consumer GPUs.

2022; Houlsby et al., 2019), LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023), but they still require backpropagating through the full backbone LLM. As a result, peak memory remains dominated by activation storage rather than weights, making fine-tuning dependent on gradient checkpointing or multiple GPUs.

Ladder-based side tuning takes a different path: it removes backward passes through the backbone, cutting peak memory by roughly half. As shown in Figure 2, Ladder Side Tuning (LST) (Sung et al., 2022) connects each Transformer block with lightweight projections into a shallow side network. Gradients flow only through the side path, keeping the backbone in inference mode and discarding intermediate activations on the fly, which typically halves memory (Figure 1). Quantized Side Tuning (QST) (Zhang et al., 2024b) adds 4-bit quantization to LST.

Despite their computational promise, ladder methods remain under-explored in the literature.

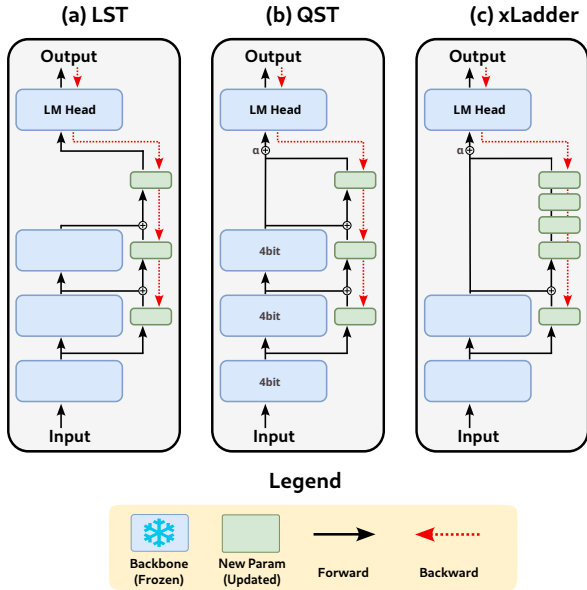


Figure 2: Architecture comparison of Ladder-based methods.

Prior studies reported empirical gains on isolated tasks but left two fundamental gaps: (i) no theoretical analysis of the memory savings achieved under generation tasks, and (ii) no scaling-law analysis to understand how ladder methods behave as data, model size, or compute resources grow.

In this work, we bring two main novel insights about the Ladder family of methods:

1. **Scaling & efficiency.** Across compute, memory and downstream accuracy, Ladder scales reliably well; its main benefit is memory savings.
2. **Design extensions.** A unifying formalism on Ladder that motivates new variants such as xLadder for trading layers for tokens.

In the following, unless specified otherwise, Ladder denotes a fully-connected side network architecture, as shown in Appendix I, Algorithm 1.

## 2 Related Work

**Parameter-Efficient Fine-Tuning (PEFT).** Full fine-tuning of LLMs is often impractical due to memory and compute costs. PEFT mitigates this by training small modules on top of a frozen backbone, such as prompt/prefix tuning (Lester et al., 2021; Li and Liang, 2021), IA<sup>3</sup> (Liu et al., 2022), and LoRA (Hu et al., 2022). QLoRA further combines LoRA with 4-bit weight quantization to cut memory for weights and optimizer states (Dettmers et al., 2023).

**Ladder and Side-Style Tuning.** An orthogonal line of work replaces backpropagation through the backbone with a lightweight side network. Ladder Side Tuning (LST) (Sung et al., 2022) attaches linear projections to each Transformer block and trains only the side path, thereby discarding intermediate activations. Subsequent work has extended and refined this paradigm. Quantized Side Tuning (QST) (Zhang et al., 2024b) adds 4-bit quantization to scale the approach to larger models. Calibration Side Tuning (CST) (Chen, 2024) introduces tiny decoder-block calibrators specifically for vision-language transfer tasks. Low-rank Attention Side Tuning (LAST) (Tang et al., 2024) replaces the side MLP with a low-rank attention projector for encoder-decoder tasks. We step back from prior work and study fundamental properties of these architectures.

**Memory-Efficient Fine-Tuning.** Activation memory represents a major bottleneck when fine-tuning large models, particularly with long contexts. It can be reduced by recomputing, with gradient checkpointing (Chen et al., 2016) or by using reversible layers (Gomez et al., 2017; Kitaev et al., 2020). ZeRO-offload (Ren et al., 2021) offloads optimizer states to CPU memory. Network compression methods such as pruning (Frankle et al., 2020), and distillation (Sanh et al., 2020; Hinton et al., 2015) shrink the inference model, but still require storing activations during training. Some approaches bypass backpropagation by using only forward passes (Malladi et al., 2023), but at the cost of noisier gradients or training steps. These gradients approximation approaches are actually complementary to Ladder methods, which use a highly reduced set of gradients, either exact or approximate.

**Fine-Tuning for Reasoning.** Recent advances in math and multi-step reasoning have relied on reinforcement learning (RL) (Chu et al., 2025; Lightman et al., 2024; Yuan et al., 2025) or self-play (Chen et al., 2024) to improve LLMs math reasoning abilities. The use of Supervised Fine-Tuning (SFT) with long distill traces from larger models enables training reasoning LLMs at lower cost than with RL (Muennighoff et al., 2025; Ye et al., 2025; Guha et al., 2025; DeepSeek-AI et al., 2025). With larger and larger contexts to train on, Ladder methods can prove useful to reduce the memory footprint of such SFT reasoning LLMs.

**Scaling Laws on Supervised Fine-Tuning (SFT).** While power-law scaling relationships are well-established for pretraining (Kaplan et al., 2020;

Hoffmann et al., 2022) extending these analyses to fine-tuning presents unique challenges. Several recent works have begun establishing fine-tuning and PEFT scaling laws (Gadre et al., 2024b; Bhagia et al., 2024; Wyatte et al., 2025; Gadre et al., 2024a; Zhang et al., 2024a; Qi et al., 2025), but without considering ladder architectures: given their major differences with other PEFT adapters, it is far from obvious whether scaling laws are the same. We give insights to answer this open question next.

### 3 Ladder Connections Formalism

The Ladder architecture (see Appendix I, Algorithm 1) provides a general interface for coupling a frozen  $L$ -layer backbone transformer with a smaller, trainable  $l$ -layer side transformer. Let  $\mathcal{C}$  denote the set of cross-layer connections ( $i \rightarrow j$ ) from backbone layer  $i$  to side layer  $j$ . A standard Ladder (depth-aligned) ladder, such as LST (Sung et al., 2022) and QST (Zhang et al., 2024b), uses  $\mathcal{C}_{\text{standard}} = \{i \rightarrow i\}_{1 \leq i \leq L=l}$ .

We explore an *extended ladder*, or **xLadder**, that increases the effective depth of computation in the side network by feeding it representations from later backbone layers. Specifically, we define

$$\mathcal{C}_{\text{xLadder}} = \{(L - \delta + 1) \rightarrow 1, (L - \delta + 2) \rightarrow 2, \dots, L \rightarrow \delta\}$$

where  $\delta = l/2$  by default. This produces an  $(L + \delta)$ -layer computation graph: the backbone contributes its last  $\delta$  layers while the side model processes these signals through its first  $\delta$  layers, effectively adding reasoning depth without training additional backbone parameters. More details in the Appendix I, Algorithm 3.

Middle layers of LLMs are known to encode more abstract, syntax-agnostic concepts, which have been shown to enhance reasoning capabilities (Jawahar et al., 2019; Voita et al., 2019). The first and last layers focus on surface-level lexical patterns. By integrating those mid-level abstractions, xLadder may increase the number of latent reasoning steps, hence reducing the need for long, complex chain-of-thoughts (CoT). Such exchange of layers for tokens is actually promoted in recent reasoning LLMs, such as GLM-4.5<sup>1</sup>. Under the assumption that the extra side depth  $\delta$  is kept fixed (or grows sublinearly) w.r.t. model scale, xLadder inherits the same memory and compute scaling as Ladder, up to constant factors.

<sup>1</sup>see <https://z.ai/blog/glm-4.5>

## 4 Scaling Laws

### 4.1 Experimental Setup

**Datasets.** We train on the EvoLM math reasoning corpus (Qi et al., 2025), a 500,000 instance collection created by augmenting the MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021) with problems extracted from MetaMathQA (Yu et al., 2024), OpenMathInstruct2 (Toshniwal et al., 2025) and NuminaMath (LI et al., 2024). EvoLM is released at multiple predefined subset sizes, ranging from 100k to 500k samples, enabling the derivation of scaling laws, using the original math problems and their solutions. Unlike several recent math reasoning datasets, EvoLM does not contain any intermediate trace produced by larger teacher models; therefore, no distillation is involved, and SFT only exploits the original reasoning traces that are provided within the datasets.

For downstream performance evaluation, we additionally fine-tune on two GLUE (Wang et al., 2018) tasks: CoLA (Corpus of Linguistic Acceptability) and QQP (Quora Question Pairs). Both are binary classification problems, CoLA assesses grammatical acceptability, whereas QQP tests semantic equivalence of question-answer pairs. Together, these datasets contribute roughly to 800k samples, which is sufficient to derive scaling laws.

**Models.** We experiment primarily with models from the Qwen family, including Qwen-2.5-Math 1.5B (Yang et al., 2024), and Qwen-2.5 7B (Qwen et al., 2025). For downstream tasks, we evaluate from the Qwen non-math, Llama and OPT series, such as Qwen-2.5 0.5B, 1.5B, 3B (Qwen et al., 2025); Llama-3.2 1B, 3B (Grattafiori et al., 2024) and OPT-1.3B, 2.7B, 6.7B (Zhang et al., 2022). The Qwen models are chosen for their strong performance on reasoning tasks, while the Llama and OPT models provide a broader range of parameter sizes and architectures for comparison.

**Implementation.** The model is trained for one epoch on subsets of the EvoLM dataset (Qi et al., 2025) that range from 100k to 500k samples to plot scaling laws: see Appendix D.5 for details. The best model is chosen based on the validation subset of EvoLM. The prompt template encourages the answer to be in the `\boxed{\}` format. The exact prompt used can be found in Appendix H.1. The model is trained on a single NVIDIA A100-80GB GPU. More details on the training hyperparameters, model configurations and hardware can be found in Appendix D.1 and D.2.

For the classification tasks, the best model is selected based on early stopping on the validation loss. We run each experiment five times for CoLA and twice for QQP, and we report the average score. More details on hyperparameters can be found in Appendix D.3.

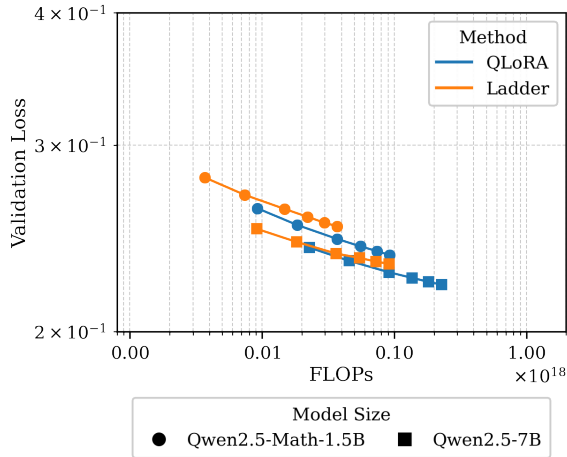


Figure 3: Scaling laws of QLoRA and Ladder methods.

## 4.2 Test Loss Scaling Laws

Zhang et al. (2024a) derive, among others, the following scaling law for LoRA fine-tuning:

$$\hat{\mathcal{L}}(D_f) = \frac{A}{D_f^\lambda} + E \quad (1)$$

This law, parameterized by scalars  $A$ ,  $\lambda$  and  $E$ , describes how the fine-tuning loss  $\hat{\mathcal{L}}(D_f)$  varies with the dataset size  $D_f$ . However, fair comparison of PEFT methods should be done at same data and compute. Let  $\zeta_m$  be the number of floating-point operations (FLOPs) per token of a given PEFT method  $m$  with  $n$  trainable parameters applied on a backbone LLM of size  $N$ . Since QLoRA backpropagates through the backbone while the Ladder only backpropagates through the side network, the standard FLOPs estimates are (Kaplan et al., 2020):

$$\begin{aligned} \zeta_{\text{QLoRA}} &= 6(N + n) \\ \zeta_{\text{Ladder}} &= 2N + 6n \end{aligned}$$

The total training compute  $C = \zeta_m \times D_f$  can be substituted into Eq. (1) to get a compute-based law:

$$\mathcal{L}(C) = \frac{A\zeta_m^\lambda}{C^\lambda} + E = \frac{B}{C^\lambda} + E \quad (2)$$

with  $B = A\zeta_m^\lambda$ . The backbone is frozen ( $N$  is constant) and  $C$  depends on  $D_f$ , which varies.  $n$  is estimated as usual from isoFLOPs (see Appendix D.5).

Using Eq. (2) and the curves in Figure 3, we estimate the scaling law coefficients for both Ladder and QLoRA methods by means of nonlinear least squares regression and Huber loss to better validate the method, as reported in Table 1.

Model	$\lambda$	$B$	$E$
<i>Curve Fit</i>			
QLoRA-Qwen-1.5B	0.26 [0.21, 0.31]	869	0.21 $\pm 0.01$
QLoRA-Qwen-7B	0.27 [0.24, 0.31]	695	0.20 $\pm 0.00$
Ladder-Qwen-1.5B	0.22 [0.12, 0.31]	189	0.21 $\pm 0.01$
Ladder-Qwen-7B	0.26 [0.10, 0.48]	534	0.21 $\pm 0.02$
<i>Huber Loss Fit</i>			
QLoRA-Qwen-1.5B	0.26 [0.18, 0.29]	701	0.21 $\pm 0.02$
QLoRA-Qwen-7B	0.26 [0.15, 0.28]	698	0.20 $\pm 0.02$
Ladder-Qwen-1.5B	0.22 [0.07, 0.28]	189	0.21 $\pm 0.03$
Ladder-Qwen-7B	0.26 [0.21, 0.52]	546	0.21 $\pm 0.02$

Table 1: Test loss scaling on different Qwen models for QLoRA and Ladder. 95% Bootstrap CI is reported for  $\lambda$  and std is reported for  $E$ .

Figure 3 and Table 1 show that the scaling slopes  $\lambda$  for both methods are very similar, indicating closely matched scaling properties. While the compute efficiency  $B$  differs, this mainly reflects vertical shifts in the loss-compute curves without altering the scaling behavior itself. The asymptotic floor  $E$  is also comparable across methods and sizes.

**Finding 1.** Despite the lack of gradients in the backbone LLM, Ladder has a similar exponent to QLoRA, with a constant offset gap.

## 4.3 Downstream Performance Scaling Law

Prior works (Qi et al., 2025; Gadre et al., 2024b; Bhagia et al., 2024; Wyatt et al., 2025) have shown that LLMs’ fine-tuning performance is preferably measured with accuracy rather than loss when studying their scaling properties. We derive from the error power law proposed by Gadre et al. (2024a) the following scaling relation on downstream tasks:

$$Err(C) = \kappa C^{-\beta} + E_\infty, \text{ with } Acc = 1 - Err(C) \quad (3)$$

where  $Err(C)$  is the error,  $C$  is the compute budget, and  $\kappa$ ,  $\beta$ , and  $E_\infty$  are parameters to fit.

We fit Eq. (3), estimating the scaling exponent  $\beta$ , the asymptotic error  $E_\infty$ , and the prefactor  $\kappa$ . Across methods and sizes, the inferred ceilings are near the task maximum within uncertainty, so we

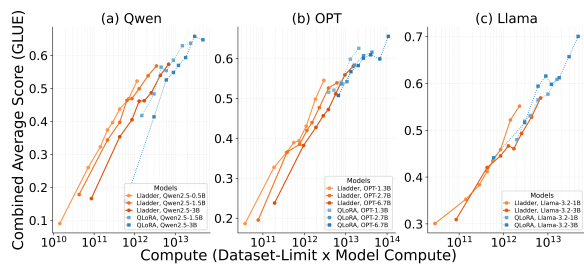


Figure 4: Scaling relation between the Ladder compute and the downstream tasks accuracy for 3 LLM series: (a) Qwen models, (b) OPT models, (c) Llama models.

focus on  $\beta$ ;  $\kappa$  primarily induces vertical shifts, compute efficiency, without altering exponent-driven scaling behavior.

Figure 4 reports training compute  $C$  for 7 epochs. For Ladder, we exploit the frozen-backbone property to cache backbone computations (see Appendix I, Algorithm 2), so the plotted compute reflects this reuse and is therefore substantially smaller than running the backbone forward pass every epoch. Full estimates appear in Appendix D.4.

**Finding 2.** Ladder and QLoRA offer comparable accuracy-compute scaling trends (i.e., similar slopes), while QLoRA attains higher accuracy at matched compute.

#### 4.4 Memory Scaling Relations

Fine-tuning large pretrained models is memory-intensive. The training-time GPU memory (VRAM) can be decomposed into three main terms:  $\mathbf{M}_1$ : model weights,  $\mathbf{M}_2$ : optimizer states, and  $\mathbf{M}_3$ : intermediate activations. PEFT methods aim to cut training cost by shrinking one or more of these terms.

LoRA (Hu et al., 2022) lowers  $\mathbf{M}_2$  by updating only a small set of low-rank adapter parameters, while QLoRA (Detmers et al., 2023) additionally reduces  $\mathbf{M}_1$  by quantizing the (frozen) backbone weights. However, in both cases the model’s intermediate activations  $\mathbf{M}_3$  are still materialized in (near) full precision, which dominates the memory footprint. In practice,  $\mathbf{M}_3$  is often the largest term during fine-tuning.

Ladder approaches such as LST (Sung et al., 2022) and QST (Zhang et al., 2024b) use a side lightweight model to reduce the memory footprint of the optimizer state ( $\mathbf{M}_2$ ) and intermediate activations ( $\mathbf{M}_3$ ). QST further reduces  $\mathbf{M}_1$  by quantizing the backbone model weights, while LST maintains full precision backbone weights. The resulting

memory savings are derived in Appendix E and visualized in Figure 5.

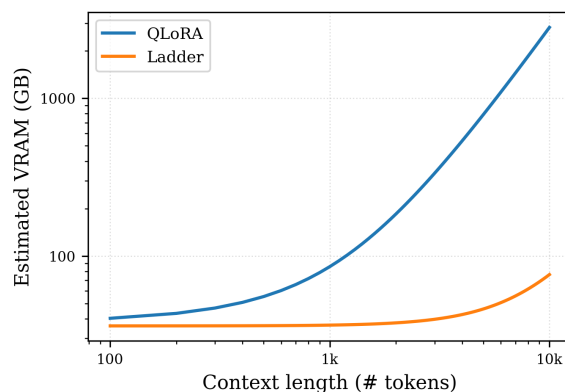


Figure 5: GPU memory required by QLoRA and Ladder during training of a 70B-parameter decoder-only transformer. We assume 8-bit AdamW, vanilla attention, and no gradient checkpointing. See Appendix E for the full derivation.

To ensure a fair comparison, we evaluate QLoRA and Ladder under the same training configuration (identical batch size and sequence length). Figure 1 shows that, at comparable model sizes, QLoRA slightly outperforms Ladder on MATH-500, but at the cost of more compute (see Section 4.2) and substantially higher VRAM. In practice, fine-tuning a 7B model with Ladder becomes possible on low-end consumer GPUs, while it may be impossible with QLoRA on the same GPU. Although memory can be traded for compute with gradient checkpointing (see Appendix E for plots that include gradient checkpointing), the memory gap between QLoRA and Ladder remains significant regardless of the memory optimization method used. To keep comparisons fair, we restrict our analysis to vanilla attention. There are many memory-saving techniques (Sliding Window Attention, Grouped Query Attention (GQA) (Ainslie et al., 2023), Flash Attention1-3 (Dao et al., 2022), PagedAttention (Kwon et al., 2023), RadixAttention (Zheng et al., 2024), FlexAttention (Dong et al., 2024), caching and NVMe offloading<sup>2</sup>), but they differ in assumptions and costs, so a comprehensive comparison is beyond this section.

**Finding 3.** Ladder methods offer significantly better memory efficiency than QLoRA, enabling fine-tuning large models on low-end consumer-grade GPUs.

<sup>2</sup>See, e.g., <https://github.com/Mega4alik/ollm>

Model	Pure SFT?	MATH-500	AIME24	AIME25	AMC23	Minerva	Olympiad Bench
<i>Based on Qwen2.5-7B</i>							
Qwen Base (Qwen et al., 2025)	✓	61.2 ± 0.4	8.7 ± 2.4	7.9 ± 4.3	32.5 ± 1.5	16.9 ± 1.2	30.2 ± 2.3
Qwen Instruct (Qwen et al., 2025)	✗	75.2 ± 0.5	8.7 ± 2.4	8.7 ± 4.3	38.5 ± 1.4	35.8 ± 1.0	38.7 ± 1.0
s1.1-7B (*) (Muennighoff et al., 2025)	✗	80.8 ± 0.6	19.0 ± 3.2	21.0 ± 5.5	59.5 ± 3.7	37.5 ± 1.1	48.2 ± 1.4
Full SFT (*) (Wang et al., 2025)	✗	58.6	10.0	7.1	45.3	24.6	27.6
QLoRA	✓	<b>70.4</b> ± 1.4	<b>8.7</b> ± 1.8	6.0 ± 4.3	41.5 ± 1.4	<b>33.5</b> ± 1.3	32.0 ± 0.8
Ladder (ours)	✓	68.9 ± 1.6	8.0 ± 1.8	<b>8.0</b> ± 1.8	<b>47.5</b> ± 7.3	29.3 ± 1.2	<b>34.3</b> ± 0.5
<i>Based on Qwen2.5-Math-1.5B</i>							
Qwen Base (Yang et al., 2024)	✓	42.0 ± 4.7	11.3 ± 3.6	5.7 ± 2.1	37.5 ± 3.5	12.1 ± 1.7	23.8 ± 0.8
Qwen Instruct (Yang et al., 2024)	✗	74.8 ± 0.5	8.7 ± 2.4	7.3 ± 5.1	42.0 ± 2.5	31.6 ± 5.7	37.9 ± 2.2
Full SFT (*) (Wang et al., 2025)	✓	49.0	7.9	2.1	35.8	14.3	23.2
QLoRA	✓	<b>70.4</b> ± 0.5	6.7 ± 2.4	6.7 ± 3.3	45.0 ± 2.5	29.1 ± 0.6	<b>33.7</b> ± 1.6
Ladder (ours)	✓	67.2 ± 1.4	<b>8.0</b> ± 5.1	<b>8.0</b> ± 3.0	<b>46.0</b> ± 4.9	<b>29.4</b> ± 1.8	32.7 ± 1.8

Table 2: Experiments results on different math reasoning benchmark. We report Pass@1 accuracy (mean ± std) of all methods across 5 random seeds. We report in (\*) the results from their original papers with their corresponding std. We note that some RL-based models are used as baselines, while ours is solely based on SFT.

## 5 Evaluation on Downstream Tasks

### 5.1 Math Reasoning Tasks

**Setup.** We train a Ladder adapter on the 1,000 training samples of the s1K-1.1 dataset (Muennighoff et al., 2025), which is a curated version of 59k math reasoning examples from multiple sources and using the trace of larger reasoning models as answer. The context length on this dataset is limited to 2,122 tokens corresponding to the longest example in the training set. Hyperparameters are chosen based on the validation loss and accuracy on a subset of the EvoLM dataset (Qi et al., 2025). The prompt template used for training and evaluation follows the answer in a `\boxed{}` format.

We evaluate on the MATH-500 dataset (Hendrycks et al., 2021), which is a benchmark comprising competition math problems of varying difficulty (we evaluate on the same 500 samples selected by OpenAI in prior work (Lightman et al., 2024)), and on the AIME’24 and AIME’25 datasets (Mathematical Association of America, 2024), each containing 30 high-school-level math problems. We also evaluate on the AMC’23 dataset (American Mathematics Competitions, 2023), with 40 examples from math competitions, and on the Minerva Math dataset (Lewkowycz et al., 2022), containing 272 undergraduate-level problems; and on the OlympiadBench dataset (He et al., 2024), comprising 674 examples.

Following Hochlehnert et al. (2025) on evalua-

tion metrics, we report the Pass@1 accuracy, which is the percentage of problems solved correctly on the first attempt. All datasets are evaluated over 5 random seeds to ensure robustness, and the results are averaged. We evaluate with greedy decoding with `max_new_tokens` of 2,048 tokens using Math-Verify (HuggingFace, 2025a) consistent with prior work indicating this budget is sufficient on these datasets (Yang et al., 2025).

We use Qwen series models such as Qwen 2.5 Math 1.5B (Yang et al., 2024), and Qwen 2.5 7B (Qwen et al., 2025) as backbone models. More details on the training hyperparameters can be found in Appendix F.

**Results.** Table 2 contrasts our Ladder models with QLoRA and other published baselines on several math reasoning benchmarks. Scores for s1.1-7B are taken directly from the Sober leaderboard (Hochlehnert et al., 2025), while the Full SFT results are from Wang et al. (2025) which does not report standard deviations. Both experimental setups slightly differ from ours, particularly in their generation token length of 32,768 tokens, which is significantly longer than our 2,048 tokens.

Despite these differences: on Qwen2.5-7B, Ladder improves mean Pass@1 on AIME25, AMC23, and OlympiadBench, while trailing on MATH-500 and Minerva; on Qwen2.5-Math-1.5B it improves on AIME24/25 and AMC23/Minerva but underperforms on MATH-500 and OlympiadBench. Ladder is generally close to QLoRA but exhibits

Method	MATH-500	AIME24	AIME25	AMC23	Minerva	Olympiad Bench
Ladder@4	67.3 ± 1.0	5.3 ± 1.8	4.0 ± 4.3	<b>39.5 ± 4.5</b>	28.2 ± 2.1	31.0 ± 0.6
xLadder@4	<b>68.4 ± 0.8</b>	<b>9.3 ± 3.7</b>	<b>6.0 ± 5.5</b>	39.0 ± 3.8	<b>28.8 ± 1.9</b>	<b>31.3 ± 1.0</b>

Table 3: Results of Ladder@4 and xLadder@4 on math reasoning benchmark. The backbone model is Qwen2.5-Math-1.5B, and Ladder is connected to the first 4 layers of the backbone, while xLadder is connected to the first 2 layers with 2 additional layers. The results are averaged over 5 random seeds.

benchmark-dependent trade-offs: most gaps are not robust across seeds, while we observe consistent deficits on MATH-500 and Minerva, and a consistent gain on OlympiadBench for 7B (Appendix G).

Prior works suggest that RL-based methods can yield larger gains than SFT-based methods on reasoning tasks (Chu et al., 2025; Luo et al., 2025; Trung et al., 2024). Combining the Ladder with RL might be a promising solution to democratize reasoning models on consumer-grade GPUs with limited memory. However, this would require adapting inference-optimized framework tools, such as SGLang (Zheng et al., 2024) and vLLM (Kwon et al., 2023), to the Ladder architecture and we leave this research track for future work.

**Finding 4.** On math reasoning benchmarks, Ladder can be competitive to QLoRA. Ladder’s primary advantage is training feasibility on consumer-grade GPUs.

## 5.2 Experiments on xLadder

**Setup & Results.** We compare Ladder and xLadder on the same math reasoning benchmark using depth  $l=4$  and Qwen2.5-Math-1.5B. Ladder@4 connects to the first four backbone layers ( $\mathcal{C} = \{i \rightarrow i\}_{1 \leq i \leq 4}$ ), while xLadder@4 connects to only the first two layers and adds two extra layers in the side network ( $\mathcal{C} = \{i \rightarrow i\}_{1 \leq i \leq 2}$ ), so both variants have the same parameter count (see Appendix B.1 for configuration details). Table 3 and Figure 6 show that xLadder improves accuracy on most datasets and tends to produce shorter CoT on both correct and incorrect answers. This is consistent with prior observations that longer CoT is not always beneficial and can correlate with errors (Wu et al., 2025), suggesting xLadder increases effective reasoning depth without requiring additional generated tokens.

We have further completed these results to compare QLoRA, Ladder and xLadder on additional benchmarks in Appendix A. More precisely, we

report comparative results on CoLA and an original LLM critic dataset. The behaviour of xLadder observed on these datasets is consistent with the previous observations.

**Finding 5.** For a given number of parameters, adjusting cross-connections in xLadder to increase the depth of the forward pass might improve performance and reduce the number of tokens generated in the CoT.

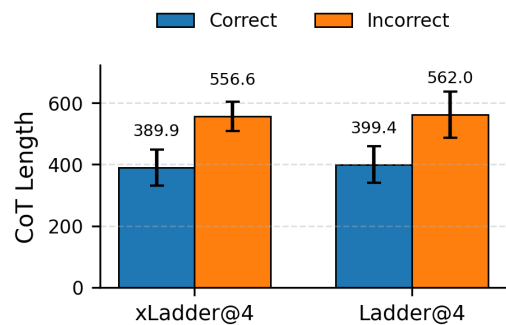


Figure 6: Reasoning average CoT length over 3 math datasets for correct and incorrect answers.

## 6 Ablation Studies

### 6.1 Ablation on Connection Patterns

**Setup.** First, to study the impact of depth, we progressively increase the number of Ladder layers, assuming a simple parallel contiguous connection pattern starting at the first layer (Figure 2, middle):  $\mathcal{C} = \{i \rightarrow i\}_{1 \leq i \leq l}$ , progressively increasing  $l$

Second, we test by fixing the Ladder depth to 4 contiguous layers and slide it up:  $\mathcal{C} = \{i + S \rightarrow i\}_{1 \leq i \leq 4}$  by progressively increasing  $S$ . We train on math reasoning tasks and on classification tasks. Results are given in Appendix B.1.

**Finding.** On math tasks, deeper Ladders connected either early or late give the best results. However, on classification benchmarks, the optimal connection pattern is different. We conclude that there is

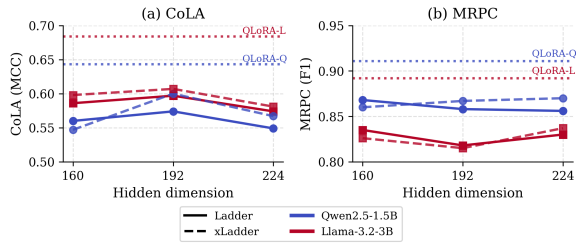


Figure 7: Width analysis on CoLA and MRPC datasets between Ladder and xLadder for Qwen-2.5-1.5B and Llama3.2-3B.

no universally optimal connection pattern, which is dataset and task-dependent.

## 6.2 Ablation on (x)Ladder Width

**Setup.** We vary next the Ladder width from 160 to 224, assuming a fixed fully-connected pattern (Figure 2, middle). For xLadder, we further insert 8 additional layers on top. More details on the training setup can be found in Appendix B.2. Results on classification tasks are shown in Figure 7.

**Finding.** On these tasks, we do not observe a clear consistent effect of hidden size; scores fluctuate modestly. Under the same setup, xLadder and Ladder perform comparably, with differences small relative to run-to-run variation.

## 6.3 Ablation on Weight Initialization

**Setup.** On the same classification task, we ablate weight initialization for Ladder and xLadder by comparing different initialization strategies: uniform, Kaiming, Xavier and Orthogonal initialization. We also vary the weight initialization scale. Results are shown in Figure 8.

**Finding.** Small weight scales hurt performance; progressively increasing the scale makes accuracy rise quickly and then plateau at moderate scales; pushing to the largest scales degrades again. Differences between initialization methods are small (within variance), and xLadder and Ladder have the same behaviour. Therefore, initialization scale is more important to tune than initialization variant.

## 7 Discussion: Ladder vs. QLoRA

**When to use what.** QLoRA is a strong default when full-backbone backprop fits the target batch size and context length, and peak accuracy is the priority. In our scaling results (Sec. 4.2), Ladder shows a similar accuracy-compute trend but can exhibit a gap at matched compute, so its main advantage is memory feasibility. Using the decom-

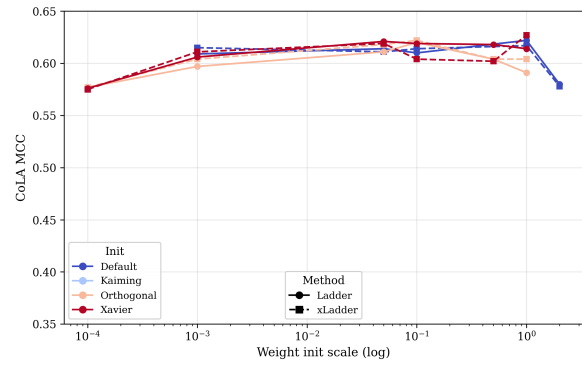


Figure 8: Weight initialization comparison on CoLA.

position in Sec. 4.4, QLoRA reduces  $M_1/M_2$  but still backpropagates through the backbone, so activation memory  $M_3$  can dominate for long contexts or larger batches. Ladder keeps the backbone in inference mode and backpropagates only through the side network, reducing the activation footprint and enabling settings that otherwise do not fit.

These approaches are also complementary to generic memory optimizations (e.g., gradient checkpointing), which trade additional compute for lower memory.

**Latency and inference overhead.** Our work focuses on training-time memory and does not benchmark inference latency. At deployment, LoRA/QLoRA can be merged into the base weights, incurring no extra per-token compute beyond the backbone forward pass (about  $2N$  FLOPs/token from Sec. 4.2). Ladder evaluates a side path with  $n$  parameters at inference, adding  $2n$  FLOPs/token, i.e., a relative overhead of  $n/N \ll 1$  in our regimes. We leave a latency-quality-memory evaluation to future work.

## 8 Conclusion

We revisit Ladder adapters and show the main key properties: its compute-performance has a similar trend to QLoRA, often with a vertical gap. Our memory analysis and reasoning benchmarks indicate Ladder is competitive while being more memory-efficient. We introduce xLadder, which deepens reasoning per time-step, hence tends to reduce CoT length. Although broader validation across models and tasks is still needed, finetuning-time depth extension emerges as a new research option to enhance reasoning, in addition to pre-training and test-time strategies. In future works, we plan to keep on exploring alternative connection patterns and extend the set of models and tasks.

## 557 Limitations

558 **Compatibility with attention optimizations.** Our  
559 memory footprint analysis assumes vanilla atten-  
560 tion, without optimizations such as FlashAttention,  
561 Flex-Attention, GQA, NVMe-offloading, etc. We  
562 acknowledge that such optimizations are impor-  
563 tant to take into account in order to better quantify  
564 the real practical memory gain brought by Ladder;  
565 however, it is extremely challenging to take them  
566 all into account, because there exist many such op-  
567 timizations, new ones are frequently proposed, and  
568 they might be dependent on specific hardware. This  
569 is why we studied in the Appendix one of the most  
570 common such optimization: gradient checkpointing.  
571 Nevertheless, we acknowledge that additional  
572 studies are required to get a better understanding  
573 of the respective trade-offs of every optimization  
574 approach. We let these studies for the future, not-  
575 ing that most of them are also complementary to  
576 Ladder, which can therefore be also combined with  
577 them to potentially cumulate memory gains.

578 **Scaling laws robustness.** Our study focuses on  
579 fine-tuning scenarios in hardware constrained set-  
580 tings, and only explores a few model families. The  
581 provided scaling laws could be more robust by val-  
582 idating them on more model families, both within  
583 the same range of sizes (e.g., Qwen2.5-1.5B and  
584 Llama-3.2-1B), as well as with larger models (e.g.,  
585 Qwen2.5-14B and Phi4-14B). We plan to broaden  
586 the scope of our proposed scaling laws with addi-  
587 tional models and tasks in future works.

588 **Reasoning and depth.** We illustrate Ladder’s  
589 flexibility by proposing a depth-extension method,  
590 xLadder. However, the research question about  
591 the impact of transformer depth on reasoning is  
592 actually highly complex. For instance, naively in-  
593 creasing depth may induce training instabilities,  
594 vanishing gradient... Furthermore, it is not clear  
595 what is the best compromise between adding more  
596 layers and adding more tokens to the CoT. Finally,  
597 many methods in the literature explore alternative  
598 ways to increase depth, such as Loop transformers,  
599 which loop over the same layers multiple times,  
600 latent reasoning models (e.g., Coconut) that stays  
601 in the latent embedding domain... In this work,  
602 we have only scratched the surface of this broad  
603 research question, and a solid analysis of our xLad-  
604 der proposal would require a whole paper on its  
605 own, with more experiments and comparison with  
606 the state-of-the-art. This is also a major track of  
607 research we plan to investigate in the future.

## References

- 608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663
- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. [Phi-4 technical report](#). Technical report, Microsoft Research.
- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.
- Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. [The falcon series of open language models](#). Technical report, The Falcon Team.
- American Mathematics Competitions. 2023. American mathematics contest 12. <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>. Accessed: 2025-06-25.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, and 32 others. 2022. [Constitutional ai: Harmlessness from ai feedback](#). *Preprint*, arXiv:2212.08073.
- Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A. Smith, Dirk Groeneveld, Pang Wei Koh, Jesse Dodge, and Hannaneh Hajishirzi. 2024. [Establishing task scaling laws via compute-efficient model ladders](#). *Preprint*, arXiv:2412.04403.
- Feng Chen. 2024. [Cst: Calibration side-tuning for parameter and memory efficient transfer learning](#). *Preprint*, arXiv:2402.12736.
- Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. [Training deep nets with sublinear memory cost](#). *Preprint*, arXiv:1604.06174.
- Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. [Self-play fine-tuning converts weak language models to strong language models](#). In *Forty-first International Conference on Machine Learning*.
- Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. [SFT memorizes](#),

664	<a href="#">RL generalizes: A comparative study of foundation model post-training.</a> In <i>Forty-second International Conference on Machine Learning</i> .	<a href="#">reliably with over-training and on downstream tasks.</a> <i>Preprint</i> , arXiv:2403.08540.	721 722
667	Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. <a href="#">Training verifiers to solve math word problems.</a> <i>Preprint</i> , arXiv:2110.14168.	Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. <a href="#">The reversible residual network: Backpropagation without storing activations.</a> In <i>Advances in Neural Information Processing Systems</i> , volume 30. Curran Associates, Inc.	723 724 725 726 727
673	Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. <a href="#">Flashattention: Fast and memory-efficient exact attention with io-awareness.</a> In <i>Advances in Neural Information Processing Systems</i> , volume 35, pages 16344–16359. Curran Associates, Inc.	Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. <a href="#">The llama 3 herd of models.</a> Technical report, Meta AI.	728 729 730 731 732 733 734 735
679	DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. <a href="#">Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.</a> Technical report, DeepSeek-AI.	Etash Guha, Ryan Marten, Sedrick Keh, Negin Raouf, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. <a href="#">Openthoughts: Data recipes for reasoning models.</a> <i>Preprint</i> , arXiv:2506.04178.	736 737 738 739 740 741 742 743
687	Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. <a href="#">QLoRA: Efficient finetuning of quantized LLMs.</a> In <i>Thirty-seventh Conference on Neural Information Processing Systems</i> .	Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. <a href="#">OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems.</a> In <i>Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.	744 745 746 747 748 749 750 751 752 753
691	Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. 2024. <a href="#">Flex attention: A programming model for generating optimized attention kernels.</a> <i>Preprint</i> , arXiv:2412.05496.	Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. <a href="#">Measuring mathematical problem solving with the math dataset.</a> <i>Preprint</i> , arXiv:2103.03874.	754 755 756 757 758
695	Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. 2025. <a href="#">Looped transformers for length generalization.</a> In <i>The Thirteenth International Conference on Learning Representations</i> .	Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. <a href="#">Distilling the knowledge in a neural network.</a> <i>Preprint</i> , arXiv:1503.02531.	759 760 761
699	Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. <a href="#">Linear mode connectivity and the lottery ticket hypothesis.</a> In <i>Proceedings of the 37th International Conference on Machine Learning</i> , volume 119 of <i>Proceedings of Machine Learning Research</i> , pages 3259–3269. PMLR.	Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. <a href="#">A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility.</a> <i>Preprint</i> , arXiv:2504.07086.	762 763 764 765 766
705	Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Alexandros G. Dimakis, Gabriel Ilharco, Shuran Song, Thomas Kollar, Yair Carmon, and 4 others. 2024a. <a href="#">Language models scale reliably with over-training and on downstream tasks.</a> <i>CoRR</i> , abs/2403.08540.	Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. <a href="#">An empirical analysis of compute-optimal large language model training.</a> In <i>Advances in Neural Information Processing Systems</i> .	767 768 769 770 771 772 773 774 775 776
714	Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, and 6 others. 2024b. <a href="#">Language models scale</a>		

777	Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski,	<i>Proceedings of Machine Learning Research</i> , pages	833
778	Bruna Morrone, Quentin De Laroussilhe, Andrea	26874–26901. PMLR.	834
779	Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019.		
780	<a href="#">Parameter-efficient transfer learning for NLP</a> . In	Brian Lester, Rami Al-Rfou, and Noah Constant. 2021.	835
781	<i>Proceedings of the 36th International Conference</i>	<a href="#">The power of scale for parameter-efficient prompt</a>	836
782	<i>on Machine Learning</i> , volume 97 of <i>Proceedings</i>	<a href="#">tuning</a> . In <i>Proceedings of the 2021 Conference on</i>	837
783	<i>of Machine Learning Research</i> , pages 2790–2799.	<i>Empirical Methods in Natural Language Processing</i> ,	838
784	PMLR.	pages 3045–3059, Online and Punta Cana, Domini-	839
		can Republic. Association for Computational Lin-	840
785	Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-	guistics.	841
786	Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu		
787	Chen. 2022. <a href="#">LoRA: Low-rank adaptation of large</a>	Aitor Lewkowycz, Anders Johan Andreassen,	842
788	<a href="#">language models</a> . In <i>International Conference on</i>	David Dohan, Ethan Dyer, Henryk Michalewski,	843
789	<i>Learning Representations</i> .	Vinay Venkatesh Ramasesh, Ambrose Slone, Cem	844
		Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu,	845
790	HuggingFace. 2025a. <a href="#">Math-verify</a> . Accessed: 2025-06-	Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra.	846
791	25.	2022. <a href="#">Solving quantitative reasoning problems with</a>	847
		<a href="#">language models</a> . In <i>Advances in Neural Information</i>	848
792	HuggingFace. 2025b. <a href="#">Open r1: A fully open reproduc-</a>	<i>Processing Systems</i> .	849
793	<a href="#">tion of deepseek-r1</a> . Accessed: 2025-06-25.		
		Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin,	850
794	Ganesh Jawahar, Benoît Sagot, and Djamé Seddah.	Roman Soletskyi, Shengyi Costa Huang, Kashif Ras-	851
795	2019. <a href="#">What does BERT learn about the structure of</a>	sul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin,	852
796	<a href="#">language?</a> In <i>Proceedings of the 57th Annual Meet-</i>	Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lam-	853
797	<i>ing of the Association for Computational Linguistics</i> ,	ple, and Stanislas Polu. 2024. <a href="#">Numinamath</a> .	854
798	pages 3651–3657, Florence, Italy. Association for		
799	Computational Linguistics.	Xiang Lisa Li and Percy Liang. 2021. <a href="#">Prefix-tuning:</a>	855
		<a href="#">Optimizing continuous prompts for generation</a> . In	856
800	Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B.	<i>Proceedings of the 59th Annual Meeting of the Asso-</i>	857
801	Brown, Benjamin Chess, Rewon Child, Scott Gray,	<i>ciation for Computational Linguistics and the 11th</i>	858
802	Alec Radford, Jeffrey Wu, and Dario Amodei. 2020.	<i>International Joint Conference on Natural Language</i>	859
803	<a href="#">Scaling laws for neural language models</a> . <i>Preprint</i> ,	<i>Processing (Volume 1: Long Papers)</i> , pages 4582–	860
804	arXiv:2001.08361.	4597, Online. Association for Computational Lin-	861
		guistics.	862
805	Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei,	Ziyue Li, Yang Li, and Tianyi Zhou. 2025. <a href="#">Skip a layer</a>	863
806	Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao	<a href="#">or loop it? test-time depth adaptation of pretrained</a>	864
807	Dong, Hongning Wang, Jie Tang, and Minlie Huang.	<a href="#">llms</a> . <i>Preprint</i> , arXiv:2507.07996.	865
808	2024. <a href="#">Critiquellm: Towards an informative critique</a>		
809	<a href="#">generation model for evaluation of large language</a>	Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri-	866
810	<a href="#">model generation</a> . <i>Preprint</i> , arXiv:2311.18702.	son Edwards, Bowen Baker, Teddy Lee, Jan Leike,	867
		John Schulman, Ilya Sutskever, and Karl Cobbe.	868
811	Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya.	2024. <a href="#">Let’s verify step by step</a> . In <i>The Twelfth Inter-</i>	869
812	2020. <a href="#">Reformer: The efficient transformer</a> . In <i>Inter-</i>	<i>national Conference on Learning Representations</i> .	870
813	<i>national Conference on Learning Representations</i> .		
		Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mo-	871
814	Vijay Korthikanti, Jared Casper, Sangkug Lym,	hta, Tenghao Huang, Mohit Bansal, and Colin Raffel.	872
815	Lawrence McAfee, Michael Andersch, Mohammad	2022. <a href="#">Few-shot parameter-efficient fine-tuning is bet-</a>	873
816	Shoeybi, and Bryan Catanzaro. 2022. <a href="#">Reducing ac-</a>	<a href="#">ter and cheaper than in-context learning</a> . In <i>Advances</i>	874
817	<a href="#">tivating recomputation in large transformer models</a> .	<i>in Neural Information Processing Systems</i> .	875
818	<i>Preprint</i> , arXiv:2205.05198.		
		Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-	876
819	Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying	Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei	877
820	Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.	Lin, Shifeng Chen, Yansong Tang, and Dongmei	878
821	Gonzalez, Hao Zhang, and Ion Stoica. 2023. Effi-	Zhang. 2025. <a href="#">Wizardmath: Empowering mathemat-</a>	879
822	cient memory management for large language model	<a href="#">ical reasoning for large language models via rein-</a>	880
823	serving with pagedattention. In <i>Proceedings of the</i>	<a href="#">forced evol-instruct</a> . In <i>The Thirteenth International</i>	881
824	<i>ACM SIGOPS 29th Symposium on Operating Systems</i>	<i>Conference on Learning Representations</i> .	882
825	<i>Principles</i> .		
		Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex	883
826	Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas	Damian, Jason D Lee, Danqi Chen, and Sanjeev	884
827	Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop,	Arora. 2023. <a href="#">Fine-tuning language models with just</a>	885
828	Ethan Hall, Victor Carbune, Abhinav Rastogi, and	<a href="#">forward passes</a> . In <i>Advances in Neural Information</i>	886
829	Sushant Prakash. 2024. <a href="#">RLAIF vs. RLHF: Scaling</a>	<i>Processing Systems</i> , volume 36, pages 53038–53075.	887
830	<a href="#">reinforcement learning from human feedback with AI</a>	Curran Associates, Inc.	888
831	<a href="#">feedback</a> . In <i>Proceedings of the 41st International</i>		
832	<i>Conference on Machine Learning</i> , volume 235 of		

889	Mathematical Association of America. 2024. American invitational mathematics examination. <a href="https://artofproblemsolving.com/wiki/index.php?title=AME_Problems_and_Solutions">https://artofproblemsolving.com/wiki/index.php?title=AME_Problems_and_Solutions</a> . Accessed: 2025-06-25.	946
890		947
891		948
892		949
893		950
894	Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. <i>s1: Simple test-time scaling</i> . <i>Preprint</i> , arXiv:2501.19393.	951
895		952
896		953
897		954
898		955
899	Zhenting Qi, Fan Nie, Alexandre Alahi, James Zou, Himabindu Lakkaraju, Yilun Du, Eric Xing, Sham Kakade, and Hanlin Zhang. 2025. <i>Evolm: In search of lost language model training dynamics</i> . <i>Preprint</i> , arXiv:2506.16029.	956
900		957
901		958
902		959
903		960
904	Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. <i>Qwen2.5 technical report</i> . Technical report, Qwen Team.	961
905		962
906		963
907		964
908		965
909		966
910		967
911	Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. <i>ZeRO-Offload: Democratizing Billion-Scale model training</i> . In <i>2021 USENIX Annual Technical Conference (USENIX ATC 21)</i> , pages 551–564. USENIX Association.	968
912		969
913		970
914		971
915		972
916		973
917		974
918	Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. <i>Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter</i> . <i>Preprint</i> , arXiv:1910.01108.	975
919		976
920		977
921		978
922	Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. <i>LST: Ladder side-tuning for parameter and memory efficient transfer learning</i> . In <i>Advances in Neural Information Processing Systems</i> .	979
923		980
924		981
925		982
926	Ningyuan Tang, Minghao Fu, Ke Zhu, and Jianxin Wu. 2024. <i>Low-rank attention side-tuning for parameter-efficient fine-tuning</i> . <i>Preprint</i> , arXiv:2402.04009.	983
927		984
928		985
929	Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisanin, Alexan Ayrapetyan, and Igor Gitman. 2025. <i>Openmathinstruct-2: Accelerating AI for math with massive open-source instruction data</i> . In <i>The Thirteenth International Conference on Learning Representations</i> .	986
930		987
931		988
932		989
933		990
934		991
935	Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. <i>ReFT: Reasoning with reinforced fine-tuning</i> . In <i>Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 7601–7614, Bangkok, Thailand. Association for Computational Linguistics.	992
936		993
937		994
938		995
939		996
940		997
941		998
942	Elena Voita, Rico Sennrich, and Ivan Titov. 2019. <i>The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives</i> . In <i>Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)</i> , pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.	999
943		1000
944		1001
945		
	Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. <i>GLUE: A multi-task benchmark and analysis platform for natural language understanding</i> . In <i>Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP</i> , pages 353–355, Brussels, Belgium. Association for Computational Linguistics.	
	Yubo Wang, Ping Nie, Kai Zou, Lijun Wu, and Wenhu Chen. 2025. <i>Unleashing the reasoning potential of pre-trained llms by critique fine-tuning on one problem</i> . <i>Preprint</i> , arXiv:2506.03295.	
	Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. <i>When more is less: Understanding chain-of-thought length in llms</i> . <i>Preprint</i> , arXiv:2502.07266.	
	Dean Wyatte, Fatemeh Tahmasbi, Ming Li, and Thomas Markovich. 2025. <i>Scaling laws for discriminative classification in large language models</i> . <i>Applied AI Letters</i> , 6(1):e109. E109 AAIL-2024-026.R1.	
	An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. <i>Qwen2.5-math technical report: Toward mathematical expert model via self-improvement</i> . Technical report, Qwen Team.	
	Wen Yang, Minpeng Liao, and Kai Fan. 2025. <i>Markov chain of thought for efficient mathematical reasoning</i> . In <i>Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)</i> , pages 7132–7157, Albuquerque, New Mexico. Association for Computational Linguistics.	
	Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. <i>Limo: Less is more for reasoning</i> . <i>Preprint</i> , arXiv:2502.03387.	
	Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhen-guo Li, Adrian Weller, and Weiyang Liu. 2024. <i>Metamath: Bootstrap your own mathematical questions for large language models</i> . <i>Preprint</i> , arXiv:2309.12284.	
	Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. 2025. <i>Free process rewards without process labels</i> . In <i>Forty-second International Conference on Machine Learning</i> .	
	Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024a. <i>When scaling meets LLM finetuning</i>	

1002        [The effect of data, model and finetuning method](#). In  
1003        *The Twelfth International Conference on Learning*  
1004        *Representations*.

1005        Susan Zhang, Stephen Roller, Naman Goyal, Mikel  
1006        Artetxe, Moya Chen, Shuohui Chen, Christopher De-  
1007        wan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mi-  
1008        haylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel  
1009        Simig, Punit Singh Koura, Anjali Sridhar, Tianlu  
1010        Wang, and Luke Zettlemoyer. 2022. [Opt: Open pre-](#)  
1011        [trained transformer language models](#). Technical re-  
1012        port, Meta AI.

1013        Zhengxin Zhang, Dan Zhao, Xupeng Miao, Gabriele  
1014        Oliaro, Zhihao Zhang, Qing Li, Yong Jiang, and Zhi-  
1015        hao Jia. 2024b. [Quantized side tuning: Fast and](#)  
1016        [memory-efficient tuning of quantized large language](#)  
1017        [models](#). In *Proceedings of the 62nd Annual Meet-*  
1018        *ing of the Association for Computational Linguis-*  
1019        *tics (Volume 1: Long Papers)*, pages 1–17, Bangkok,  
1020        Thailand. Association for Computational Linguistics.

1021        Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan  
1022        Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin,  
1023        Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang,  
1024        Joseph E Gonzalez, and Ion Stoica. 2023. [Judging](#)  
1025        [llm-as-a-judge with mt-bench and chatbot arena](#). In  
1026        *Advances in Neural Information Processing Systems*,  
1027        volume 36, pages 46595–46623. Curran Associates,  
1028        Inc.

1029        Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue  
1030        Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos  
1031        Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark  
1032        Barrett, and Ying Sheng. 2024. [Sglang: Efficient](#)  
1033        [execution of structured language model programs](#). In  
1034        *Advances in Neural Information Processing Systems*,  
1035        volume 37, pages 62557–62583. Curran Associates,  
1036        Inc.

## A Additional Experiments

### A.1 Classification task: LLM critic

**Setup.** We build an LLM critic dataset by sampling from the MATH-500 dataset (Hendrycks et al., 2021). First, several pretrained solvers (i.e., LLMs) are asked to produce a Chain-of-Thought (CoT) along with an answer for each question in the MATH-500 dataset. We keep only generations in which the `\boxed{}` answer occurs within the first 2,048 tokens, then verify the correctness of the answers using Math-Verify (HuggingFace, 2025a). The resulting corpus is balanced (50/50 correct-incorrect) with an equal number of correct and incorrect answers, leading to a dataset of 380 training and 396 test samples, with an average length of 600 tokens. This dataset will be freely available on HuggingFace.

For the critic itself, we fine-tune Qwen2.5-0.5B and Qwen2.5-3B backbones with the Ladder and QLoRA methods on 3 epochs with a learning rate of  $5E-4$ . These hyperparameters were selected based on the convergence of the training curve. The task is binary: predict whether the input CoT is correct or incorrect.

The prompt used for training and evaluation is provided in Appendix H.3.

**Results.** Table 4 shows the results of our LLM critic on the test set. Both fine-tuning methods significantly improve the near-random performance of the base models to over 80% accuracy. As both methods achieved similar results, we can conclude that training data quality and implementation constraints (i.e., limited GPUs) are more important than the fine-tuning method itself.

Model	Method	% Acc
Qwen2.5-0.5B	Base	59.0
Qwen2.5-0.5B	Ladder	<b>81.8</b>
Qwen2.5-0.5B	xLadder	79.0
Qwen2.5-0.5B	QLoRA	79.0
Qwen2.5-3B	Base	56.0
Qwen2.5-3B	Ladder	81.0
Qwen2.5-3B	xLadder	82.0
Qwen2.5-3B	QLoRA	<b>82.8</b>

Table 4: LLM critic results on 396 test samples. 95% Wald-Confidence interval is  $\pm 3.9\%$ .

LLMs are increasingly used to assess the quality of the generation of other LLMs, e.g., LLM-as-a-Judge tasks (Zheng et al., 2023), reward modeling for RL fine-tuning (Lee et al., 2024; Bai et al.,

2022), and grading critics (Ke et al., 2024). Our critic fits this trend, it provides a fast, and 82% accurate filter that screens out most flawed CoTs before passing the higher confidence ones to a more expensive process such as human evaluation or RL-based fine-tuning.

The critic has a residual error of 18% and the modest training dataset size leaves room for robustness improvement under distribution shift, such as the use of more diverse pretrained LLMs or the use of more complex prompts.

### A.2 Classification task: CoLA

**Setup.** In the GLUE benchmark, we focus on the CoLA dataset, which is a binary classification task on the grammatical acceptability of sentences. We use the same parameters provided in prior LST work (Sung et al., 2022).

We reimplement the architecture as the original code was not compatible with recent versions of the Python libraries. The original LST architecture is implemented with a side network that is connected to every layer of the backbone model, and it is initialized by pruning the backbone model weights. Additional experiments with a slightly different and simpler architecture are also conducted, especially with an uniform initialization and an unweighted gated sum  $\alpha$  between the ladder and backbone activations, instead of a weighted sum as in the original LST architecture.

Model	Method	Reproduction	Original
OPT1.3B	QLoRA	0.630	$0.621 \pm 0.023$
OPT1.3B	LST	0.579	$0.595 \pm 0.031$
OPT2.7B	QLoRA	0.656	$0.637 \pm 0.026$
OPT2.7B	LST	0.592	$0.607 \pm 0.035$
OPT6.7B	QLoRA	0.665	$0.643 \pm 0.028$
OPT6.7B	LST	0.596	<i>unreported</i>

Table 5: Reproduction of the results reported in (Sung et al., 2022) on the CoLA benchmark with our Ladder codebase. The metric used is MCC.

For reproduction purposes, the backbone models used are OPT family models (Zhang et al., 2022), which are 1.3B, 2.7B and 6.7B parameters. Additional newer models such as Llama3.2-3B and Qwen2.5-1.5B are also used to compare the Ladder architecture with the QLoRA method.

We trained on 7 epochs with early stopping with a learning rate of  $1E-5$ , the AdamW optimizer and a cosine learning-rate schedule. All the results are

4 seeds average. This is performed on a single ADA RTX5000 32GB GPU.

**Results.** Re-running the original LST experiments on the CoLA benchmark, we notice a difference in the results compared to the original paper as shown in Table 5. This difference is likely due to the use of different library versions and GPU kernels, which can lead to slight variations in the results. However, the differences observed on the Ladder models are all within the range of variability reported in the paper, which validates our reproduction and open-source codebase.

Model	Method	MCC
Llama3.2-3B	QLoRA	<b>0.69</b>
Llama3.2-3B	Ladder	0.61
Llama3.2-3B	xLadder	<u>0.64</u>
Qwen2.5-1.5B	QLoRA	<b>0.62</b>
Qwen2.5-1.5B	Ladder	0.58
Qwen2.5-1.5B	xLadder	<u>0.59</u>

Table 6: Additional results on CoLA with newer models and a simpler ladder: uniform initialization, sum gates between ladder and backbone  $\alpha = 0.5$ . The Wald 95% Confidence Interval is  $\pm 0.03$ .

Furthermore, comparing the Ladder with the xLadder, we observe that the xLadder architecture seems to consistently outperform the Ladder architecture as shown in Table 6. This suggests that the additional layers and connections in the xLadder architecture provide a performance boost over the standard Ladder architecture. Finally, when comparing the Ladder method with QLoRA, we observe that the ladder-based methods achieve slightly lower results than QLoRA on the CoLA classification task. This suggests that initialization by pruning tends to give slightly better results.

## B Additional Ablation Studies

### B.1 More Details on Depth & Starting Placement

#### B.1.1 Math Reasoning Task

**Setup.** We train Qwen2.5-Math-1.5B on the open-s1 dataset (HuggingFace, 2025b; Muennighoff et al., 2025) (a filtered 52k version of s1) for 1 epoch, with learning rate  $2E-4$  and batch size 2, on a single NVIDIA A100-80GB. Then, we fixed the Ladder depth to 4 layers, and vary the starting index along the backbone model. We report the accuracy on the MATH-500 dataset. QLoRA

is trained with standard hyperparameters  $r = 16$ ,  $\alpha = 32$  and dropout=0.005.

**Results.** Figure 9 confirms the U-shape accuracy curve. This shows that either fully connected Ladder or very few connected layers yield the best results. The noisiness of the curve is due to the limited one training seed run, and its sensitivity to random factors such as training seed or batch samples.

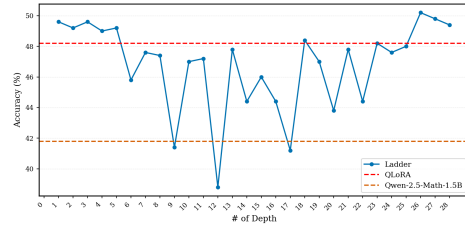


Figure 9: Accuracy of different number of layers connected on MATH500. Red dashed line represents the QLoRA baseline. Orange dashed line represents the base model accuracy.

This U-shape trend remains when varying the starting position of the connected layers as shown in Figure 10. The xLadder seems to be more sensitive to the starting position, which suggests that the additional layers and connections in the xLadder architecture may amplify the effects of starting position, making it more important to choose an optimal starting point for maximum performance.

#### B.1.2 GLUE Classification Tasks

**Setup.** We train Qwen2.5-1.5B and Llama3.2-3B on the CoLA and MRPC datasets from the GLUE benchmark (Wang et al., 2018) for 7 epochs, with learning rate  $1E-4$  and a cosine learning rate schedule. All the results are 4 seeds average. The training is performed on a single ADA RTX5000 32GB GPU. We report the MCC (Mathews Correlation Coefficient) for CoLA. Ladder is fixed to depth 5. **Results.** Figure 11 shows that increasing the depth of non-connected layers on the xLadder architecture does not significantly impact the performance on both CoLA and MRPC datasets.

Figure 12 shows the starting position of the connected layers for both Ladder and xLadder architectures. The trend is different from the one observed on math reasoning tasks, as the optimal starting position seems to be more centered in the middle layers. This suggests that the optimal starting position for connected layers may vary depending on the specific task and dataset, and that different tasks may require different strategies for selecting

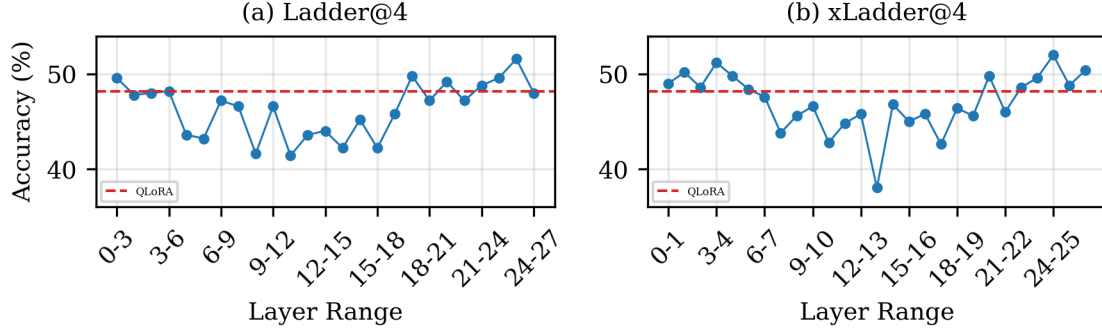


Figure 10: Starting position of the connected layers on MATH-500. The red dashed line represents the QLoRA baseline. On the left, (a) is Ladder for a depth of 4, and on the right, (b) xLadder for a depth of 4 including 2 additional layers.

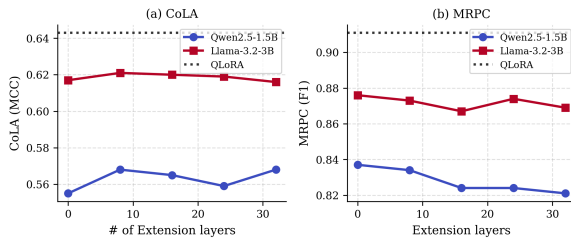


Figure 11: Performance on CoLA and MRPC on the number of extended layers for the xLadder architecture. Ladder depth is fixed to 5.

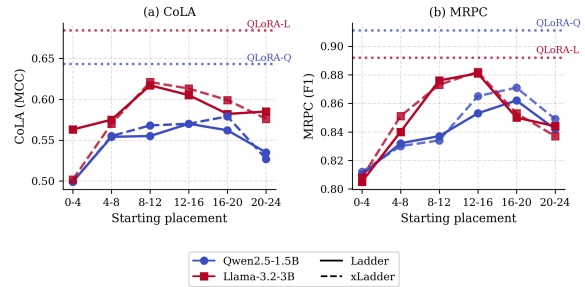


Figure 12: Starting position of the connected layers on CoLA and MRPC datasets between Ladder and xLadder for Qwen-2.5-1.5B and Llama3.2-3B. Ladder is fixed to depth 5, and xLadder to depth 5 with 8 additional layers.

1188 the starting position of connected layers in Ladder-  
1189 based architectures.

## 1190 B.2 Implementation Details on Width and 1191 Initialization Ablations

1192 The same setup is used for both experiments. We  
1193 use Qwen2.5-1.5B and Llama3.2-3B backbones  
1194 and train on the CoLA and MRPC datasets from the  
1195 GLUE benchmark (Wang et al., 2018) for 7 epochs,  
1196 with learning rate 1E-4 and a cosine learning rate  
1197 schedule. All the results are 4 seeds average. The  
1198 training is performed on a single ADA RTX5000  
1199 32GB GPU. We report the MCC (Mathews Cor-  
1200 relation Coefficient) and F1 scores for CoLA and  
1201 MRPC, respectively.

1202 For QLoRA, we use standard hyperparameters  
1203  $r = 16$ ,  $\alpha = 32$  and dropout=0.005.

1204 For Ladder and xLadder, we use a fully con-  
1205 nected architecture and a random uniform initializa-  
1206 tion; for xLadder, we further add 8 non connected  
1207 layers on top.

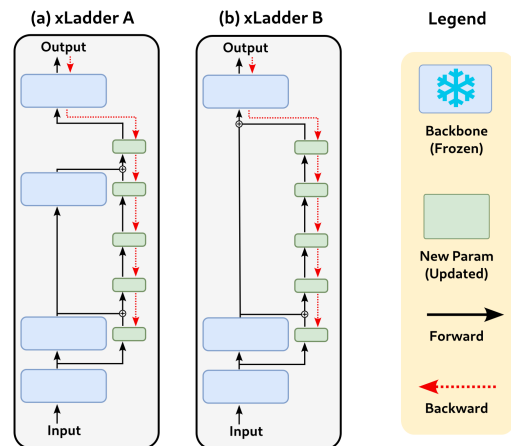


Figure 13: Overview of the xLadder architecture.

1208

## C Details on xLadder Architecture

1209

### C.1 Definition

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

As defined in Section 3, the xLadder architecture is an extension of the Ladder architecture that introduces additional layers to enhance reasoning depth while maintaining the efficiency of the original Ladder design. The xLadder architecture is designed to improve token efficiency in reasoning tasks by introducing a new concept called *free layers*, i.e., non-connected layers. Figure 13 illustrates the xLadder architecture, which consists of a frozen backbone model  $f$  and a side model  $g$  built on top of the backbone. The side model is composed of a series of *connected layers*  $k$  that are directly connected to the backbone model and a series of *free layers*  $m$  that are not connected to the backbone model.

We created two variants of the xLadder architecture where the difference comes from the last layer: in the first variant (Figure 13 (a)), the last layer of the backbone model is always connected to the side model before it goes to the backbone head, while in the second variant (Figure 13 (b)), the last layer of the side model is not connected to the backbone’s last layer. The first variant inspired by the original LST architecture (Sung et al., 2022), while the second variant is inspired by the QST architecture (Zhang et al., 2024b). The choice of the last layer connection depends on the specific task and the desired reasoning depth.

The connected layers are responsible for processing the input tokens and generating intermediate representations, while the free layers can be engaged to enhance the model’s reasoning capabilities without requiring additional tokens. Therefore, this fine-tuning technique is memory efficient and allows for effective training without excessive resource consumption, while still maintaining performance comparable to QLoRA within confidence intervals, as shown in Figure 14.

### C.2 Investigating Depth Extension

Depth (number of layers) limits how many sequential reasoning steps a Transformer can apply to a fixed input. Simply adding layers often harms optimization and memory (e.g., vanishing gradients) (Fan et al., 2025). A common workaround is chain-of-thought (CoT): the model appends intermediate tokens and reprocesses them, which boosts performance but increases latency and memory; some works also loop selected layers at test time (Li

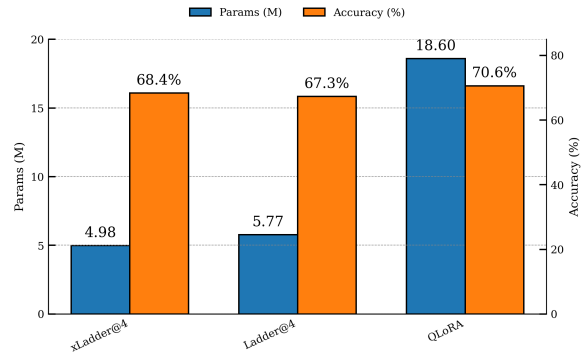


Figure 14: Avg@5 accuracy on MATH-500 for xLadder@4, Ladder@4 and QLoRA with CI95=4.00%. The backbone model is Qwen2.5-Math-1.5B. QLoRA requires more than 3 times the number of parameters to achieve similar results.

et al., 2025).

1258

### C.3 xLadder Configuration

1259

We simplify the original Ladder architecture (Sung et al., 2022; Zhang et al., 2024b) in four ways: (i) we do not connect all layers to the side model, and (ii) we fix the width of the connected layers to a constant, (iii) we replace the different projection methods to a linear projection, and (iv) we initialize the side model with a uniform initialization instead of using the backbone model weights (i.e., pruning method).

1260

1261

1262

1263

1264

1265

1266

1267

1268

Each connected layer  $k$ , which are smaller width-wise by design, is connected to the backbone model  $f$  via a linear projection, which allows the model to leverage the pre-trained knowledge of the backbone while still being able to adapt to specific tasks. Compared to current ladder approaches, this simplified architecture is more flexible and efficient since it does not require all layers to be trainable.

1269

1270

1271

1272

1273

1274

1275

1276

The additional free layers  $m$  are designed to be the same as the connected layers, but without the backbone connection. Therefore, the computational complexity of these layers is  $\mathcal{O}(d_s^2)$ , where  $d_s$  is the dimension of the hidden states in the side model.

1277

1278

1279

1280

1281

1282

The side model  $g$  last layer is summed with the output of the backbone model  $f$  to form the final output. As stated in QST (Zhang et al., 2024b), this design is effective against initialisation drift.

1283

1284

1285

1286

## D Scaling Law Implementation Details

### D.1 Test Loss Scaling Hyperparameters Details

For the scaling laws experiments in test loss scaling, we train for one epoch on each  $D_f = \{50k, 100k, 200k, 300k, 400k\}$  sample with a random seed of 39. The best model is selected based on the validation loss after each epoch on a subset of EvoLM dataset (Qi et al., 2025). To speed up the training, different batch sizes are used as stated in Table 7 while learning rate is kept constant at  $2E-5$  with a cosine scheduler with warmup ratio of 0.1. The optimizer is 8-bit AdamW with weight decay of 0.01. The sequence length is set to the maximum length of the dataset, which is 1,839 tokens. QLoRA configuration is set to 4-bit quantization with  $r = 16$  and  $\alpha = 32$  on all linear layers.

All those runs can be performed on a single NVIDIA A100 GPU with 80GB of memory, with training taking approximately an hour for the Ladder and two hours for the QLoRA model on the 100k dataset sample.

Model	Method	Micro Batch Size	Gradient Acc
Qwen2.5-1.5B	Ladder	8	3
Qwen2.5-1.5B	QLoRA	4	5
Qwen2.5-7B	Ladder	4	3
Qwen2.5-7B	QLoRA	2	5

Table 7: Hyperparameters for the test loss scaling experiments. The effective batch size is the product of the micro batch size and the gradient accumulation steps.

### D.2 Test Loss Scaling Ladder Configuration Details

To configure the Ladder architecture the most optimal way, we perform a hyperparameter search on the depth while keeping constant the width to 256. The depth is defined as the number of layers in the Ladder architecture. The search is performed on a 10k dataset sample, and the results are shown in Figure 15. The best depth configuration is found to be around 6 layers for 1.5B model, while 4 layers for 7B model. We decide to use those depth configurations for the Ladder architecture in the scaling laws experiments.

The overall configuration of the Ladder is that the side network is uniformly initialized, as well as its linear projection, with unweighted gated sum (i.e., no more  $\alpha$ ) as shown in Figure 2.

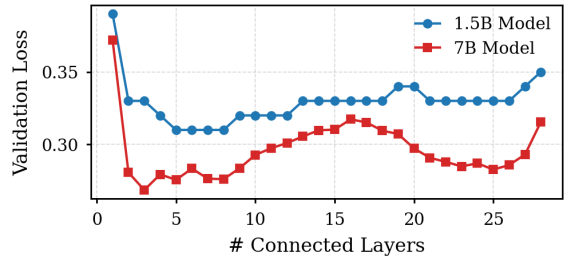


Figure 15: Test loss for different Ladder depth-configurations on the 10k dataset sample. The depth goes from 1 to  $L = 28$ , where  $L$  is the number of layers in the backbone model.

### D.3 Downstream Tasks Hyperparameters Details

On downstream tasks, we trained on multiple epochs on two datasets: 7 epochs on CoLA, and 4 epochs on QQP. The best model is selected based on the validation loss computed on each epoch. Ladder use a batch size of 4, while QLoRA could not learn properly with small batch size, so we use a batch size of 32. Their learning rate is set to  $1E-5$  with a cosine scheduler. The optimizer used is normal AdamW.

The QLoRA configuration is the same as previously, with 4-bit quantization and  $r = 16$  and  $\alpha = 32$  on all linear layers. The Ladder architecture is a fully connected ladder on each layer of the backbone model, with a uniform initialization of the side network and its linear projection.

### D.4 Downstream Tasks Scaling Laws Equations Fit

Table 8 shows the fitted scaling law parameters for the Ladder and QLoRA methods on the downstream task experiments. We report solely the exponent  $\beta$  of the error-compute power law stated in Section 4.3 Eq.(3), fitted using the Scipy’s `curve_fit` function. The confidence intervals are computed using a bootstrap method with 95% confidence level.

Given the sparse sampling of compute, especially for QLoRA, the bootstrap intervals are broad; accordingly, Ladder and QLoRA exhibit similar scaling behavior within statistical uncertainty. All the fitted plots are shown in Figures 16, 17, and 18.

### D.5 Details on Scaling Laws

#### D.5.1 Intuitive Explanation of IsoFLOP

Because our focus is performance as a function of compute, we follow the first scaling relation of Ka-

Model	Ladder $\beta$	QLoRA $\beta$
Qwen-1.5B	0.15 [-0.1, 0.3]	0.45 [0.2, 0.7]
Qwen-3B	0.15 [-0.1, 0.3]	0.33 [0.1, 0.6]
OPT-1.3B	0.12 [-0.4, 0.6]	0.26 [0.0, 0.5]
OPT-2.7B	0.13 [-0.2, 0.4]	0.24 [-0.3, 1.3]
OPT-6.7B	0.13 [-0.1, 0.3]	0.22 [0.3, 1.2]
Llama-1B	0.09 [-0.8, 1.0]	0.45 [0.2, 0.7]
Llama-3B	0.1 [-0.3, 0.5]	0.13 [-0.5, 0.7]

Table 8: Scaling law parameter for QLoRA and Ladder methods on downstream tasks. The 95% Bootstrap CI is reported.

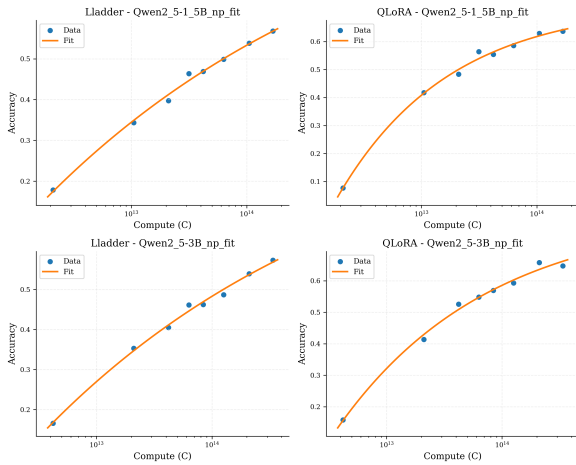


Figure 16: Downstream scaling law for Qwen series models comparing Ladder and QLoRA methods.

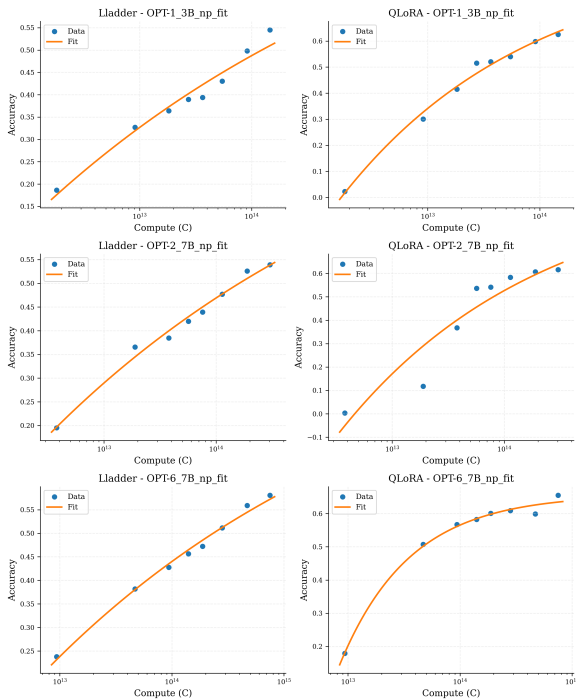


Figure 17: Downstream scaling law for OPT series models comparing Ladder and QLoRA methods.

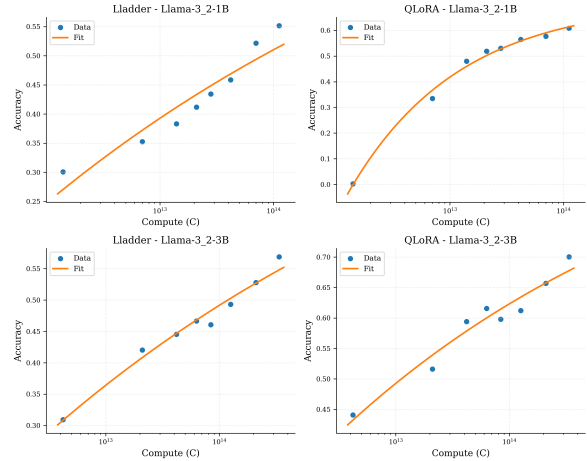


Figure 18: Downstream scaling law for Llama series models comparing Ladder and QLoRA methods.

plan et al. (2020). For text generation, we measure test loss  $Y(C)$  at increasing fixed compute budgets  $C$  under two idealized assumptions: effectively unbounded data  $D$  and unbounded model size  $N$ .

- **Infinite data.** Each training is run for a single epoch, over as many data samples as possible until the target compute budget  $C$  is exhausted.
- **Infinite model size.** For each  $C$ , we search over model sizes  $N$  and take  $N^*$  that minimizes  $Y(C)$ . In practice, larger  $N$  increases per-sample cost, so at fixed  $C$  a larger  $N$  implies fewer training samples  $D$ .

Varying  $N$  at fixed  $C$  yields an **isoFLOP** test-loss curve with a characteristic U-shape: small  $N$  underfits (insufficient capacity), while very large  $N$  overfits the compute budget (too little data), both producing higher loss. The compute-optimal frontier is then obtained by taking, for each  $C$ , the minimum test loss along its isoFLOP curve.

## D.5.2 Practical Implementation of IsoFLOP for xLadder

The “infinite model size” assumption is ill-suited to parameter-efficient methods such as LST: on isoFLOP curves, the compute-optimal  $N^*$  typically lies far beyond reasonable ladder sizes. In practice, the best points would require very large adapter networks, undermining memory budgets and the premise of parameter efficiency.

To empirically trace an isoFLOP curve, we increase the effective parameter count  $N$  by (i) adding non-connected extension layers and (ii) widening their hidden size.

**Setup.** We use a Qwen2.5-14B backbone (48 layers, hidden size 5120). Each ladder layer mirrors a Qwen2.5 block with a reduced hidden size of 320. The Ladder has a depth of 5, connected to backbone layers  $\{0, 11, 23, 35, 47\}$ . We then insert  $l$  non-connected extension layers immediately before the last connected ladder layer (i.e., 47), varying  $l \in \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200\}$ .

We fine-tune on a subset of the French Claire Dialogue corpus<sup>3</sup> to adapt the backbone to French dialogue. We select the first 10k utterances shorter than  $4 \times 2048$  characters and evaluate test loss on the processed test split, yielding 10 points along the scaling curve.

**Results.** A representative isoFLOP curve is shown in Fig. 19. VRAM usage grows with  $l$ , and we stop when out-of-memory (OOM) occurs on a 32 GB GPU.

As shown in Figure 19, memory constraints within a parameter-efficient setting restrict us to the left side of the isoFLOP curve. Consequently, we select the largest feasible  $N$ , which is not compute-optimal but is the closest attainable point to the true optimum  $N^*$ .

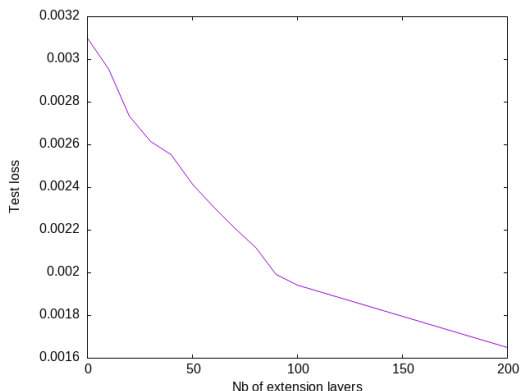


Figure 19: IsoFLOP of the xLadder.

Following prior work, we assume that post-Ladder performance follows a compute scaling law, analogous to full and other PEFT fine-tuning regimes. We cannot fully validate this assumption here: doing so would require broader sweeps over *models*, *datasets*, and *scales*, plus out-of-fit extrapolation tests to assess generalization of the fitted law. Given limited compute, we adopt the standard scaling formulation and fit it for Ladder fine-tuning. This is a reasonable working assumption, supported

<sup>3</sup><https://huggingface.co/datasets/OpenLLM-France/Claire-Dialogue-French-0.1>

by extensive evidence of scaling behavior in related training settings.

## E Memory requirements computation

We formalize peak per-GPU memory during fine-tuning under two regimes: (i) QLoRA, and (ii) Ladder (Side) Tuning. We decompose peak memory into frozen model parameters, trainable parameters, optimizer state, gradients, and activations:

$$M_{\text{peak}} = M_{\text{model}} + M_{\text{optim}} + M_{\text{grads}} + M_{\text{acts}} \quad (4)$$

We rely on the formulas given in Eleuther blog post<sup>4</sup> and section 4.1 of NVIDIA activations paper (Korthikanti et al., 2022) to compute the memory requirements of a transformer model during training assuming a vanilla attention mechanism and without a gated linear unit (GLU) or other nonlinearities.

As stated by the Falcon Team (Almazrouei et al., 2023), for GLU the memory requirements are roughly x1.5 times larger, as it doubles the number of parameters in the MLP.

We assume a training in bf16, a 8-bit AdamW optimizer, no gradient checkpointing.

**QLoRA Memory Requirements.** The main frozen parameters are quantized in 4 bits (i.e., int4), while the additional trained parameters are stored in 2 bytes (i.e., bf16). The memory required for the parameters is:

$$M_{\text{model}} = M_{\text{frozen}} + M_{\text{trainable}}$$

$$M_{\text{model}} = \frac{N}{2} + 2n$$

with  $N$  number of main (frozen) parameters and  $n$  number of additional trainable parameters.

Classic AdamW stores 12 bytes per trainable parameter corresponding to 4 bytes for parameters, 4 bytes for the momentum and 4 bytes for the variance. While, AdamW in 8-bit stores 6 bytes per trainable parameter, so the memory required for the optimizer is:

$$M_{\text{optim}} = 6n$$

Gradients are typically stored on the full back-propagation path and requires 2 bytes (i.e., bf16) per parameter:

$$M_{\text{grads}} = 2n$$

<sup>4</sup>see <https://blog.eleuther.ai/transformer-math/#training>

To avoid incurring extra costs during backpropagation, activations of all parameters are typically computed, during the forward pass and kept in memory to later compute the gradients; this requires:

$$M_{\text{acts}} = M_{\text{acts,baseline}} + M_{\text{acts,qlora}}$$

$$M_{\text{acts,baseline}} = s b h L \left( 34 + 5 \frac{as}{h} \right)$$

For QLoRA activations, each adapted linear adds (conceptually) the intermediate  $Z = XA \in \mathbb{R}^{(bs) \times r}$ , with  $n = 2hrmL$  trainable parameters then:

$$A_{\text{QLoRA}} = s b r m L$$

$$M_{\text{acts,QLoRA}} = 2 A_{\text{QLoRA}} = \frac{n}{h} s b$$

with  $s$  = sequence length,  $b$  = batch size (or micro batch size),  $h$  = hidden size,  $a$  = number of attention heads,  $r$  = rank used in QLoRA.

Following (4), the total memory required by QLoRA is:

$$M_{\text{peak,QLoRA}} = \frac{N}{2} + 10n + sbhL \left( 34 + 5 \frac{as}{h} \right) + \frac{sbn}{h} \quad (5)$$

**Ladder Memory Requirements.** We assume the same number of additional parameters  $n$  in QLoRA and Ladder. The memory for parameters and optimizer is the same.

Because there is no backpropagation in the main LLM, then the gradients memory is only:

$$M_{\text{grads,ladder}} = 2n$$

But the main difference comes from the activations: they do not need to be stored at all in the main LLM, only in the ladder. Therefore, for the same low-rank dimension  $r$ ,  $l$  layers and  $a_{lad}$  attention head in the ladder, we have:

$$M_{\text{acts}} = M_{\text{acts,ladder}}$$

$$M_{\text{acts,ladder}} = s b r l \left( 34 + 5 \frac{a_{lad}s}{r} \right)$$

Following (4), the total memory required by Ladder is:

$$M_{\text{peak,ladder}} = \frac{N}{2} + 10n + sbrl \left( 34 + 5 \frac{as}{r} \right) \quad (6)$$

Concretely, let's consider LLama3.1-70B:  $N = 70.10^9$ ,  $h = 8192$ ,  $L = 80$ ,  $a = 64$ . Its context length is  $s = 128,000$  tokens, and we assume  $b = 1$ . Let's assume adding  $n = 10^8$  parameters,

i.e., 0.1% of the parameters, with  $r = 8$ ,  $a_{lad} = 1$  and  $l = L$  layers in the ladder. Then, we have:

From (5), we can compute the memory requirements for QLoRA:

$$M_{\text{peak,QLoRA}} = \frac{N}{2} + 10n + sbhL \left( 34 + 5 \frac{as}{h} \right) + sb \left( \frac{n}{h} \right)$$

$$= 26.2 \times 10^3 s^2 + 22 \times 10^6 s + 37.75 \times 10^9$$

From (6), we can compute the memory requirements for Ladder:

$$M_{\text{peak,ladder}} = \frac{N}{2} + 10n + sbrl \left( 34 + 5 \frac{a_{lad}s}{r} \right)$$

$$= 400s^2 + 21.76 \times 10^3 s + 36 \times 10^9$$

## Memory Requirements With Gradient Checkpointing.

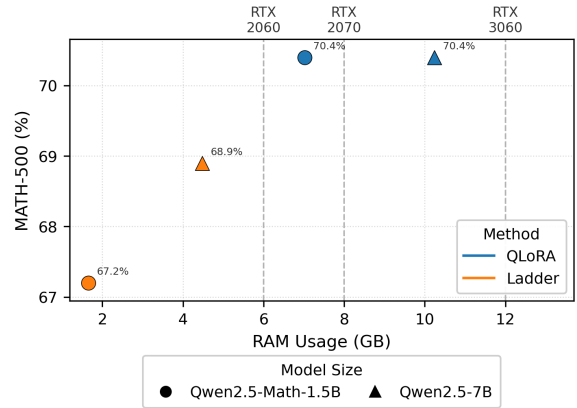


Figure 20: Accuracy on MATH-500 with gradient checkpointing enabled and a batch size of 16, with a 2k-token context length. RAM usage is computed for vanilla transformers, vertical lines represent which fine-tuning method can run on which consumer GPUs.

Gradient checkpointing, also known as activation checkpointing, trade memory for compute by freeing, during forward, all parameters' activations except at the output of every layer. Then, during backward within each layer, this layer's activations are recomputed.

The above equations are thus simplified; for QLoRA, memory requirements are:

- Activations:  $2sbhL + sb\left(\frac{n}{h}\right)$  1535
- Gradients:  $2n$  1536
- Parameters:  $N/2 + 2n$  1537

• Optimizers:  $6n$

For (x)Ladder, memory requirements are:

• Activations:  $2sbrl$

• Gradients:  $2n$

• Parameters:  $N/2 + 2n$

• Optimizers:  $6n$

Gradient checkpointing trades GPU memory for compute. For completeness, we report xLadder under checkpointing in Figure 20.

**Other Memory Optimizations.** As explained in the Limitations section 8, many various memory optimizations exist (e.g., FlashAttention, GQA, Page-dAttention), they are complementary to Ladder architecture and can be used jointly with it.

A pipeline-style strategy is particularly well suited to fine-tuning with Ladder: stream one frozen backbone layer at a time over the corpus, write intermediate activations to disk, then train the (small) Ladder alone on these cached activations. This collapses GPU RAM to roughly a single-layer footprint at the cost of I/O and storage. In our setting, about 200 GB of fast NVMe sufficed for 30k sequences, enabling training on a very low-end GPU within a few days.

We limit ourselves in this work to vanilla attention and gradient checkpointing, leaving analysis of all other optimizations for future work.

## F Implementation Math Reasoning hyperparameters

The training is performed for 3 epochs, and the model is evaluated on the validation set after each epoch. The sequence length is set to the maximum length of the dataset, which is 2122 tokens. For Ladder fine-tuning, the batch size is set to 8 (resp. 4) for 1.5B model (resp. 7B model), and the learning rate is set to  $2E-4$ . For QLoRA fine-tuning, the batch size is set to 4 (resp. 2) for 1.5B model (resp. 7B model), and the learning rate is set to  $2E-5$ . The Ladder architecture used in this section is a fully connected ladder on each layer of the backbone model uniformly initialized without any gated sum, but only unweighted gated sum. The backbone LLM is quantized in 8-bit precision. The model is trained on a single NVIDIA A100 GPU with 80GB of memory.

Benchmark	$\Delta$ Pass@1	McNemar $p$ -value
MATH-500	-3.3 [-4.0, -2.4]	0.018
AIME24	1.3 [-2.7, 4.7]	0.99
AIME25	1.3 [-2.7, 6.0]	0.99
AMC23	1.0 [-5.0, 5.0]	0.91
Minerva	-1.8 [-3.0, -0.5]	0.67
OlympiadBench	-1.0 [-1.9, 0.4]	0.37

Table 9: **Robustness and paired significance (Qwen2.5-Math-1.5B).**  $\Delta$  is Ladder–QLoRA Pass@1 (percentage points), with paired 95% bootstrap confidence intervals over 5 seeds. McNemar  $p$ -values are computed per seed from paired per-example correctness and combined across seeds using Fisher’s method.

Benchmark	$\Delta$ Pass@1	McNemar $p$ -value
MATH-500	-1.5 [-3.5, 0.8]	0.009
AIME24	-0.7 [-2.7, 1.3]	0.99
AIME25	2.0 [-1.3, 4.7]	0.99
AMC23	6.0 [ 0.0, 11.5]	0.568
Minerva	-4.2 [-5.4, -2.8]	0.016
OlympiadBench	2.3 [ 1.5, 3.0]	0.047

Table 10: **Robustness and paired significance (Qwen2.5-7B).**  $\Delta$  is Ladder–QLoRA Pass@1 (percentage points), with paired 95% bootstrap confidence intervals over 5 seeds. McNemar  $p$ -values are computed per seed from paired per-example correctness and combined across seeds using Fisher’s method.

## G Seed-level CIs for Ladder vs QLoRA.

Tables 9 and 10 report paired 95% bootstrap confidence intervals over seeds for the Pass@1 difference  $\Delta = \text{Ladder} - \text{QLoRA}$ . We observe that Ladder improves OlympiadBench at 7B, while QLoRA is consistently better on Minerva and MATH-500. The McNemar  $p$ -values further confirm these findings, showing statistically significant differences on MATH-500 at both scales, Minerva at 7B, and OlympiadBench at 7B, while other datasets show no significant difference between methods. In other words, the observed performance differences are robust to random seed variation, indicating that Ladder is in average competitive with QLoRA but not uniformly better. Overall, these results support our main conclusion: Ladder can be competitive, but its primary advantage is training-time memory feasibility rather than uniformly higher accuracy at matched compute.

1602  
1603  
1604  
1605  
1606

## H Experiment Prompts

### H.1 Math Reasoning Prompt

Only Qwen2.5 series models are used for math reasoning tasks. The prompt used is the same as the one used by Yang et al. (2024).

#### Prompt

```
<|im_start|>system
Please reason step by step, and put your
final answer within \boxed{ }.<|im_end|>
<|im_start|>user
{problem}<|im_end|>
<|im_start|>assistant
{answer}<|endoftext|><|im_end|>
```

#### Example

```
<|im_start|>system
Please reason step by step, and put your
final answer within \boxed{ }.<|im_end|>
<|im_start|>user
Given a rational number, write it as a
fraction in lowest terms and calculate
the product of the resulting numerator
and denominator. For how many rational
numbers between 0 and 1 will  $20!x^{\frac{1}{x}}$ 
be the resulting product?<|im_end|>
<|im_start|>assistant
Given a rational number between 0 and 1,
we need to write it as a fraction in
lowest terms  $\frac{a}{b}$  and
calculate the product of the resulting
numerator and denominator. We need to
determine how many such rational numbers
exist where the product  $a \times b = 20!$ .
1. Prime Factorization of  $20!$ :
- The prime factors of  $20!$  are 2, 3,
5, 7, 11,
13, 17, and 19. These are 8 distinct
primes.
2. Coprime Pairs:
- For  $a \times b = 20!$  and
 $\gcd(a, b) = 1$ ,
each prime factor of  $20!$  must be
assigned entirely
to either  $a$  or  $b$ . This
results in
 $2^8 = 256$  coprime pairs  $((a,
b))$ .
3. Counting Pairs  $(a < b)$ :
- Since  $a$  and  $b$  are coprime
and
 $a \times b = 20!$ , each pair  $((a,
b))$  and
 $((b, a))$  are distinct. To find
pairs where
 $a < b$ , we divide the total number of
pairs
by 2, excluding the case  $(a = b)$ 
(which is
impossible since  $20!$  is not a square).
```

1607

```
4. Result:
- The number of coprime pairs  $((a,
b))$  with
 $a < b$  is  $\frac{256}{2} = 128$ .
```

Thus, the number of rational numbers
between 0 and 1
for which the product of the numerator and
denominator in lowest terms is  $20!$  is
 $\boxed{128}$ .

1609

### H.2 GLUE CoLA Prompt

For the GLUE CoLA task, the CoLA sentence is directly given as a prompt to the LLM complemented with a trained Boolean Classification head.

1610

1611

1612

1613

#### Example

```
User: Our friends won't buy this
analysis, let alone the next one we
propose.
Assistant: 1 (acceptable)
```

1614

### H.3 MATH500 LLM Critic Dataset

The MATH500 dataset is used in Section A.1 to generate a new fine-tuning dataset for the LLM Critic task. This task involves evaluating a given MATH-500 question, a model's CoT and its answer to determine if the answer is correct or wrong.

1615

1616

1617

1618

1619

1620

The dataset is formatted as MCQ with two possible answers: the fine-tuned models has to generate a single token A for "Correct" or B for "Wrong". This is enforced by applying the softmax only to both logits for "A" and "B". The data is generated using Q4\_K GGUF versions of the following models via llama.cpp<sup>5</sup>:

1621

1622

1623

1624

1625

1626

1627

- phi-4-Q4\_k\_gguf 1628
- Meta-Llama-3.1-8B-Instruct-Q4\_k\_GGUF 1629
- Qwen3-32B-Q4\_k\_GGUF 1630
- Qwen3-14B-Q4\_k\_GGUF 1631
- deepseek-math-7b-instruct-Q4\_k\_GGUF 1632
- Qwen2.5-32B-Instruct-Q4\_k\_GGUF 1633
- Qwen2.5-Math-7B-Instruct-Q4\_k\_GGUF 1634

While the Q4\_K quantization may reduce the accuracy of model reasoning, it enables creating the dataset at a reasonable computation cost. The final dataset is balanced in terms of correct and incorrect answers, which are all contained within

1635

1636

1637

1638

1639

<sup>5</sup><https://github.com/ggml-org/llama.cpp>

1608

1640 a `\boxed{}` field. All answers exhibiting severe  
1641 repetition patterns are manually eliminated.

#### System Prompt all models except Qwen3

You are a highly intelligent and logical mathematical assistant. Please reason step by step to solve the problem. After your complete reasoning, put your final numerical answer within `\boxed{}`. If the answer is an expression, put the expression within `\boxed{}`.

1642

#### Qwen3 System Prompt

You are a highly logical and accurate mathematical reasoning engine. /think Please provide a detailed step-by-step solution. Ensure your final answer is enclosed within `\boxed{}`.

1643

#### User Prompt for MATH-500 Dataset Generation

Question: {problem}

Please reason step by step to solve the problem. After your reasoning, put your final answer in the format `\boxed{your_answer}`.

1644

#### Critic Task Prompt

User: {question}  
Assistant1: {generated model answer}

User LLM Critic: Is the tentative answer correct or wrong?

Choices:  
A: Correct  
B: Wrong

Assistant: {label}

1645

#### Example of Generation Prompt

You are a highly intelligent and logical mathematical assistant. Please reason step by step to solve the problem. After your complete reasoning, put your final numerical answer within `\boxed{}`. If the answer is an expression, put the expression within `\boxed{}`.

Question: Convert the point  $(0,3)$  in

1646

rectangular coordinates to polar coordinates. Enter your answer in the form  $(r,\theta)$ , where  $r > 0$  and  $0 \leq \theta < 2\pi$ .

Please reason step by step to solve the problem. After your reasoning, put your final answer in the format `\boxed{your_answer}`.

1647

#### Example of LLM Critic

Two-thirds of the students at Baker Middle School take music. There are 834 students who take music. How many students are there at Baker Middle School?

Here is a tentative answer to the math problem:

Let's break down the problem step by step:

1. We know that two-thirds of the students take music, and there are 834 students who take music.
2. Let's represent the total number of students at Baker Middle School as  $x$ .
3. Since two-thirds of the students take music, we can set up the equation:  $(2/3)x = 834$
4. To solve for  $x$ , we can multiply both sides of the equation by  $3/2$  to get rid of the fraction:  $x = 834 \times (3/2)$
5. Simplifying the right-hand side of the equation, we get:  $x = 834 \times 3/2$
6. Multiplying 834 by 3 gives us:  $x = 2502 / 2$
7. Dividing 2502 by 2 gives us:  $x = 1251$

Therefore, there are 1251 students at Baker Middle School.

`\boxed{1251}`

Question:

Is the tentative answer correct or wrong?

Choices:

- A: Correct  
B: Wrong

1648

Answer: A

1649

1650

## I Ladder Algorithms

---

### Algorithm 1: LADDER algorithm

---

**Input** : Tokens  $x$ ; backbone blocks  $\{\text{BackboneBlock}_\ell\}_{\ell=0}^{N-1}$ ; tap  $\mathcal{T}$ ; Ladder blocks  $\{\text{LadderBlock}_j\}_{j=0}^{L-1}$ ; down-projections  $\{\text{DownProj}_k\}$ ; up-projection  $\text{UpProj}$ .

**Output** : Logits.

$h \leftarrow \text{EMBED}(x)$ ;  $r \leftarrow \mathbf{0}$  // backbone embedding, ladder stream

**for**  $\ell \leftarrow 0$  **to**  $N - 1$  **do**

$h \leftarrow \text{BackboneBlock}_\ell(h)$

**if**  $\ell \in \mathcal{T}$  **then**

$k \leftarrow \text{TAPINDEX}(\ell)$

$r \leftarrow \text{LadderBlock}_k(r + \text{DownProj}_k(h))$  // accumulate taps in Ladder

$h \leftarrow h + \text{UpProj}(r)$

logits  $\leftarrow \text{LMHead}(h)$  // Norm possible

**return** logits

---

---

### Algorithm 2: LADDERCACHED with reusable cached backbone forward

---

**Input** : Dataset  $\mathcal{D}$ ; frozen backbone  $B$ ; Ladder module  $\mathcal{L}$ ; taps  $\mathcal{T}$ ; epochs  $E$ .

**Output** : Trained Ladder parameters (and saved checkpoint).

**Stage A: One-time caching (since  $B$  is frozen).**

$\mathcal{C} \leftarrow \text{CacheBackbone}(B, \mathcal{D}, \mathcal{T})$  // store tap states + base hidden for injection

**Stage B: Multi-epoch training using cache.**

**for**  $e \leftarrow 1$  **to**  $E$  **do**

**foreach**  $batch(c, y) \sim \mathcal{C}$  **do**

$(H_{\text{tap}}, h_{\text{base}}) \leftarrow c$  // cached features; labels  $y$

$r \leftarrow \text{Ladder}(H_{\text{tap}})$  // trainable; recomputed each step

$h \leftarrow h_{\text{base}} + \text{UpProj}(r)$  // single injection

logits  $\leftarrow \text{LMHead}(h)$

$\mathcal{L} \leftarrow \text{LOSS}(\text{logits}, y)$

UPDATE( $\mathcal{L}$ )

---

---

**Algorithm 3:** XLADDER: eXtended Ladder algorithm

---

**Input** : Tokens  $x$ ; backbone blocks  $\{\text{BackboneBlock}_\ell\}_{\ell=0}^{N-1}$ ; tap indices  $\mathcal{T} = [t_0 < \dots < t_{K-1}]$ ; xLadder depth  $L \geq K$  with blocks  $\{\text{LadderBlock}_j\}_{j=0}^{L-1}$ ; down-projections  $\{\text{DownProj}_i\}_{i=0}^{K-1}$ ; up-projection  $\text{UpProj}$ .

**Output** : Logits.

```

 $h \leftarrow \text{EMBED}(x); r \leftarrow \mathbf{0}; i \leftarrow 0$ 
// backbone state, ladder stream,
// tap counter

for  $\ell \leftarrow 0$  to  $N - 1$  do
   $h \leftarrow \text{BackboneBlock}_\ell(h)$ 
  if  $\ell \in \mathcal{T}$  then // normal Ladder
     $r \leftarrow \text{LadderBlock}_i(r + \text{DownProj}_i(h))$ 
    // consume tap  $i$ 

    if  $i = K - 1$  then // xLadder
      for  $j \leftarrow K$  to  $L - 1$  do
         $r \leftarrow \text{LadderBlock}_j(r)$ 
        // extended depth,
        // detached from
        // backbone
       $h \leftarrow h + \text{UpProj}(r)$ 
       $r \leftarrow \mathbf{0}$ 
     $i \leftarrow i + 1$ 

logits  $\leftarrow \text{LMHead}(h)$  // Norm possible

return logits

```

---