Actes de CORIA-TALN 2026
Nantes, France, 29 juin au 3 juillet 2026

Oui à l'Échelle, Non à la Mémoire: Affinage Léger des LLMs par Réseaux Latéraux

Estelle Zheng[1, 2] Sébastien Warichet[2] Emmanuel Helbert[2] Christophe Cerisara[1]
(1) Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
(2) Alcatel-Lucent Enterprise, Illkirch / Colombes, France
{estelle.zheng,christophe.cerisara}@loria.fr,
{estelle.zheng,sebastien.warichet,emmanuel.helbert}@al-enterprise.com

RÉSUMÉ

L'affinage des grands modèles de langues (LLMs) est souvent limité par la mémoire disponible sur les GPUs. Les méthodes d'affinage fin efficace (PEFT), telles que QLoRA, réduisent le nombre de paramètres pouvant être entraînés, tout en utilisant une grande quantité de mémoire lors de l'entraînement dû à la rétropropagation du modèle dans son intégralité. Nous revisitons l'architecture Ladder (LST), une technique PEFT rarement explorée qui ajoute un petit réseau latéral. Nous démontrons que sa pente des lois d'échelles correspond à celle de QLoRA, tout en réduisant son pic d'utilisation mémoire. Sur la tâche de résolutions des mathématiques, LST a des performances proches de QLoRA, tout en offrant une possibilité d'affiner sur des GPU grand public. Nous exploitons une extension de l'architecture de la Ladder en introduisant xLadder, une variante qui augmente la profondeur du réseau latéral tout en raccourcissant sa chaîne de pensée (CoT).

ABSTRACT

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

Fine-tuning large language models (LLMs) is often constrained by the memory available on GPUs. Parameter-efficient fine-tuning (PEFT) methods, such as QLoRA, reduce the number of trainable parameters, yet still incur a high memory usage during training induced by the backward pass in the full model. We revisit the Ladder Side Tuning (LST) architecture, a rarely explored PEFT technique that adds a small side network. We demonstrate that its scaling law slope matches that of QLoRA, while reducing its peak memory usage. On mathematical problem solving tasks, LST achieves performance close to QLoRA, while offering the possibility of fine-tuning on consumer-grade GPUs. We explore an extension of the Ladder architecture by introducing xLadder, a variant that increases the depth of the side network while reducing its chain of thought (CoT).

MOTS-CLÉS : affinage, efficacité, peu coûteux, LLM.

KEYWORDS: fine-tuning, memory-efficient, cheap, LLM, peft.

1 Introduction

L'affinage des grands modèles de langage (LLMs) est essentielle pour adapter des modèles généraux à des tâches spécifiques, mais elle est de plus en plus limitée par la mémoire GPU. Les LLMs comportant plus de 7 milliards de paramètres (Grattafiori *et al.*, 2024; Qwen *et al.*, 2025; Abdin *et al.*, 2024) nécessitent beaucoup de ressources, imposant des tailles de batch plus petites et des longueurs

Actes des 33ème Conférence sur le Traitement Automatique des Langues Naturelles.
Volume 1 : articles scientifiques originaux, pages 419–438.

de contexte plus courtes.

Les méthodes d'affinage efficace en paramètres (PEFT) réduisent le nombre de paramètres entraînables grâce à des techniques telles que les adaptateurs (Li & Liang, 2021; Lester *et al.*, 2021; Liu *et al.*, 2022; Houlsby *et al.*, 2019), LoRA (Hu *et al.*, 2022) et QLoRA (Dettmers *et al.*, 2023), mais elles nécessitent toujours une rétropropagation à travers l'intégralité du LLM. En conséquence, le pic de mémoire est surtout impacté par le stockage des activations plutôt que par les poids, ce qui limite fortement les configurations d'entraînement possibles.

Ladder adopte une approche différente : elle supprime la rétropropagation à travers le réseau central, réduisant ainsi le pic de mémoire d'environ moitié. Comme illustré en Figure 1, Ladder Side Tuning (LST) (Sung *et al.*, 2022) connecte chaque bloc Transformers à un réseau latéral peu profond au moyen de projections légères. Les gradients circulent uniquement dans le réseau latéral, alors que le réseau central est en mode inférence, supprimant à la volée les activations intermédiaires, ce qui divise généralement la mémoire par deux (Figure 3a). Quantized Side Tuning (QST) (Zhang *et al.*, 2024b) ajoute une quantification en 4-bits à LST. Malgré leur potentiel computationnel, les méthodes Ladder restent encore peu explorées dans la littérature. Les travaux antérieurs ont rapporté des gains empiriques sur des tâches isolées, mais ont laissé deux lacunes fondamentales : (i) l'absence d'analyse théorique des économies de mémoire dans les tâches de génération, et (ii) l'absence d'analyse de lois d'échelle permettant de comprendre le comportement de ces méthodes lorsque les données, la taille du modèle ou les ressources de calcul augmentent.

Dans ce travail, nous apportons deux contributions principales concernant la famille des méthodes Ladder :

1. **Lois à l'échelle et efficacité.** En termes de calcul et de mémoire, Ladder présente un bon passage à l'échelle ; son principal avantage réside dans son efficacité en mémoire.
2. **Extensions de conception.** xLadder qui est une extension de la Ladder, permettant d'échanger des couches contre des tokens.

Dans la suite, sauf indication contraire, Ladder désigne une architecture de réseau latéral entièrement connectée.

2 État de l'Art

Affinage fin à efficacité paramétrique (PEFT). L'affinage complet (*full fine-tuning*) des LLM est souvent irréalisable en raison des coûts en mémoire et en calcul. Le PEFT atténue ce problème en entraînant de petits modèles au-dessus d'un réseau de base gelé, tels que le *prompt/prefix tuning* (Lester *et al.*, 2021; Li & Liang, 2021), IA^3(Liu *et al.*, 2022), et LoRA (Hu *et al.*, 2022). QLoRA combine en outre LoRA avec une quantification des poids sur 4 bits afin de réduire la mémoire requise pour les poids et les états de l'optimiseur (Dettmers *et al.*, 2023).

Affinage type Ladder. Une ligne de travail orthogonale remplace la rétropropagation à travers le réseau de base par un réseau latéral léger. Ladder Side Tuning (LST) (Sung *et al.*, 2022) attache des projections linéaires à chaque bloc Transformers et n'entraîne que le chemin latéral, écartant ainsi les activations intermédiaires. Des travaux ultérieurs ont étendu et affiné ce paradigme. Quantized Side Tuning (QST) (Zhang *et al.*, 2024b) ajoute une quantification sur 4 bits pour adapter l'approche à des modèles plus grands. Calibration Side Tuning (CST) (Chen, 2024) introduit de minuscules calibrateurs

de blocs décodeurs spécifiquement pour les tâches de transfert vision à TAL. Low-rank Attention Side Tuning (LAST) (Tang *et al.*, 2024) remplace le MLP latéral par un projecteur d'attention de bas rang pour les tâches encodeur-décodeur. Nous prenons du recul par rapport aux travaux antérieurs et étudions les propriétés fondamentales de ces architectures.

Affinage et efficacité en mémoire. La mémoire des activations représente la charge la plus importante lors de l'affinage des LLMs, en particulier avec de longs contextes. Elle peut être réduite par recalcul, grâce à la sauvegarde des activations (Chen *et al.*, 2016) ou par l'utilisation de couches réversibles (Gomez *et al.*, 2017; Kitaev *et al.*, 2020). ZeRO-offload (Ren *et al.*, 2021) décharge les états de l'optimiseur vers la mémoire CPU. Les méthodes de compression de réseau telles que le *pruning* (Frankle *et al.*, 2020) et la distillation (Sanh *et al.*, 2020; Hinton *et al.*, 2015) réduisent le modèle d'inférence, mais nécessitent toujours le stockage des activations pendant l'entraînement. Certaines approches contournent la rétropropagation en n'utilisant que des passes avant (Malladi *et al.*, 2023), mais au prix de gradients plus bruités ou d'étapes d'entraînement supplémentaires. Ces approches d'approximation des gradients sont en réalité complémentaires aux méthodes Ladder, qui utilisent un ensemble fortement réduit de gradients, qu'ils soient exacts ou approximatifs.

Ajustement fin pour le raisonnement. Les récentes avancées en tâche de mathématiques et en raisonnement multi-étapes reposent sur l'apprentissage par renforcement (RL) (Chu *et al.*, 2025; Lightman *et al.*, 2024; Yuan *et al.*, 2025) ou le *Self-Play* (Chen *et al.*, 2024) pour améliorer les capacités de raisonnement mathématique des LLM. L'utilisation de l'apprentissage supervisé (SFT) avec de longues traces de raisonnement, souvent distillé à partir de modèles plus grands. Cela permet d'entraîner des LLM de raisonnement à moindre coût qu'avec le RL (Muennighoff *et al.*, 2025; Ye *et al.*, 2025; Guha *et al.*, 2026; DeepSeek-AI *et al.*, 2025). Avec des contextes de plus en plus longs à entraîner, les méthodes Ladder peuvent s'avérer utiles pour réduire l'empreinte mémoire de tels LLM de raisonnement entraînés par SFT.

3 Méthodologie

Sur la Figure 1, la méthode du Ladder Side Tuning (LST) (Sung *et al.*, 2022) a deux points distincts :

1. **Geler les paramètres du grand modèle (*backbone*)** : permettant ainsi au modèle de ne pas apprendre sur l'ensemble des paramètres ;
2. **Rétropropagation sur un modèle petit latéral (*side*)** : la mise à jour des gradients ne se fait que sur le petit modèle, évitant ainsi complètement le calcul sur le grand modèle.

Il existe plusieurs variétés de cette architecture qu'on va appeler **Ladder**, notamment une version 4-bit quantisée (Zhang *et al.*, 2024b) et une extension possible qui permet de réduire les tokens en augmentant la profondeur du petit modèle. [1]

De façon formelle, l'architecture Ladder fournit une interface générale permettant de coupler un grand modèle gelé à L couches (en bleu sur la Figure 1) avec un Transformers latéral plus petit et entraînable à l couches (en vert sur la Figure 1). Soit $\mathcal{C}$ l'ensemble des connexions inter-couches $(i \rightarrow j)$ allant de

1. La chaîne de pensée des LLMs peut s'apparenter à une réflexion explicite, où le modèle va sortir des tokens pour réfléchir. Donc, il existe une relation entre le nombre de tokens sortis et la longueur de la chaîne de pensée.

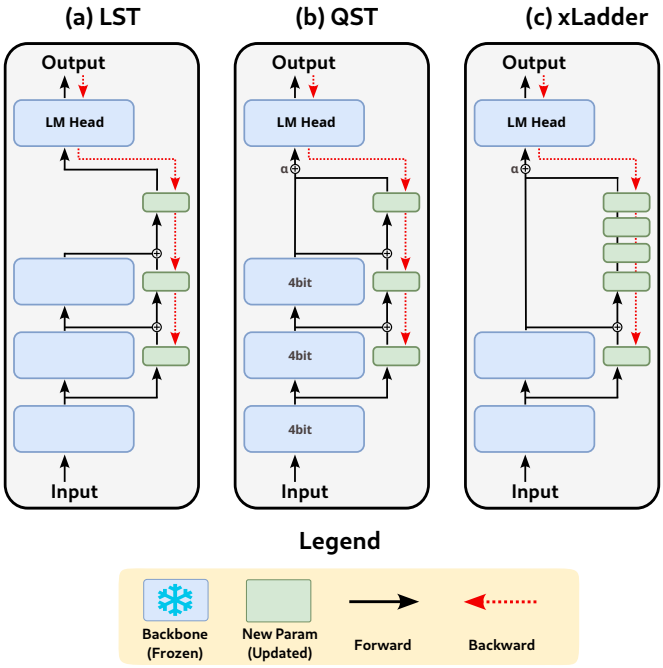


FIGURE 1 – Comparaison des architectures des méthodes Ladder.

la couche i central vers la couche j du module latéral. Une Ladder standard (alignée en profondeur), telle que LST (Sung *et al.*, 2022) et QST (Zhang *et al.*, 2024b), utilise $\mathcal{C}_{\text{standard}} = \{i \to i\}_{1 \le i \le L=l}$.

Nous explorons une *Ladder étendue*, ou **xLadder**, qui augmente la profondeur effective de calcul dans le réseau latéral en l'alimentant avec des représentations issues de couches plus profondes du réseau central. Plus précisément, nous définissons :

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

où $\delta = l/2$ par défaut.

Cela produit un graphe de calcul à $(L + \delta)$ couches : le backbone contribue avec ses δ dernières couches tandis que le modèle latéral traite ces signaux à travers ses δ premières couches, ajoutant ainsi de la profondeur de raisonnement sans entraîner de paramètres supplémentaires du backbone.

Il est connu que les couches intermédiaires des grands modèles de langage encodent des concepts plus abstraits et moins dépendants de la syntaxe, ce qui améliore les capacités de raisonnement (Jawahar *et al.*, 2019; Voita *et al.*, 2019). Les premières et dernières couches se concentrent davantage sur des motifs lexicaux de surface. En intégrant ces abstractions de niveau intermédiaire, xLadder pourrait augmenter le nombre d'étapes latentes de raisonnement, réduisant ainsi le besoin de chaînes de pensée (*Chain-of-Thought*, CoT) longues et complexes.

Un tel échange de couches contre des tokens est d'ailleurs encouragé dans des LLMs récents orientés vers le raisonnement, tels que GLM-4.5 [2]. Sous l'hypothèse que la profondeur latérale supplémentaire δ est maintenue fixe (ou croît de manière sous-linéaire) par rapport à la taille du modèle, xLadder hérite des mêmes lois d'échelle en mémoire et en calcul que Ladder, à des facteurs constants près.

2. voir https://z.ai/blog/glm-4.5

4 Lois à l'Échelle

4.1 Paramètres Expérimentaux

Corpus et jeux de données. Le corpus de raisonnement mathématique EvoLM (Qi *et al.*, 2026) est une collection de 500 000 instances créée en augmentant les données de MATH (Hendrycks *et al.*, 2021) et GSM8K (Cobbe *et al.*, 2021) avec des problèmes extraits de MetaMathQA (Yu *et al.*, 2024), OpenMathInstruct2 (Toshniwal *et al.*, 2025) et NuminaMath (LI *et al.*, 2024). EvoLM existe sous plusieurs tailles de sous-ensembles prédéfinies, allant de 100 000 à 500 000 exemples, ce qui permet de dériver des lois d'échelle à partir des problèmes mathématiques originaux et de leurs solutions. Contrairement à plusieurs jeux de données récents en raisonnement mathématique, EvoLM ne contient aucune trace intermédiaire produite par des modèles plus performant ; aucune distillation n'est donc impliquée, et l'apprentissage supervisé (SFT) exploite uniquement les traces de raisonnement originales fournies dans le corpus.

Modèles. Nous expérimentons principalement avec des modèles de la famille Qwen, notamment **Qwen-2.5-Math 1.5B** (Yang *et al.*, 2024) et **Qwen-2.5 7B** (Qwen *et al.*, 2025).

Implémentation. Les modèles sont tous entraînés pendant une époque sur des sous-ensembles du jeu de données EvoLM (Qi *et al.*, 2026), allant de 100 000 à 500 000 échantillons, afin de tracer les courbes d'échelle. Le meilleur modèle est sélectionné sur la base du sous-ensemble de validation d'EvoLM. Le prompt encourage la production de la réponse au format `\boxed`. L'entraînement est effectué sur un unique GPU NVIDIA A100-80GB.

4.2 Mise à l'Échelle et Fonction de Perte

Zhang *et al.* (2024a) dérivent, entre autre, la loi d'échelle suivante pour l'affinage avec LoRA :

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

Cette loi, paramétrée par les scalaires A, λ et E, décrit comment la fonction de perte de l'affinage $\hat{\mathcal{L}}(D_f)$ varie en fonction de la taille du corpus D_f. Toutefois, une comparaison équitable des méthodes de PEFT doit être réalisée à données et calcul équivalents.

Soit ζ_m le nombre d'opérations en FLOPs par token pour une méthode PEFT m donnée, comportant n paramètres entraînables et appliquée à un LLM (réseau central) de taille N. Étant donné que QLoRA rétropropage les gradients à travers le réseau central tandis que Ladder ne rétropropage qu'à travers le réseau latéral, les estimations standards du nombre de FLOPs sont (Kaplan *et al.*, 2020) :

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

Le coût total d'entraînement $C = \zeta_m \times D_f$ peut être substitué dans l'Éq. (1) afin d'obtenir une loi d'échelle exprimée en fonction du calcul :

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

avec $B = A\zeta_m^{\lambda}$. Le *backbone* est gelé (i.e., N est constant) et C dépend de D_f, qui varie.

En utilisant l'Éq. (2) et les courbes de la Figure 2, nous estimons les coefficients de la loi d'échelle pour les méthodes Ladder et QLoRA au moyen d'une régression non linéaire par moindres carrés, et aussi avec une perte de Huber [3] afin de mieux valider la méthode, comme indiqué dans le Tableau 1.

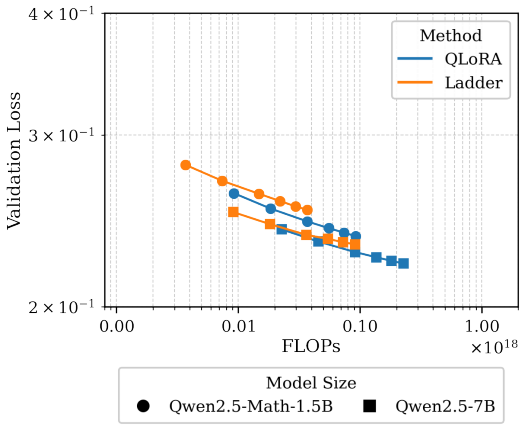


FIGURE 2 – Lois par courbes des méthodes QLoRA et Ladder.

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

TABLE 1 – Paramètres de la loi à l'échelle en fonction de perte comparant QLoRA et Ladder pour différents modèles Qwen. L'intervalle de confiance à 95% en bootstrap et l'écart-type sont donnés.

La Figure 2 et le Tableau 1 montrent que les pentes de passage à l'échelle λ des deux méthodes sont très similaires, indiquant des propriétés de lois à l'échelle étroitement comparables. Bien que la variable B diffère, cela reflète principalement des décalages verticaux dans les courbes perte-calcul sans modifier le passage à l'échelle directement. La limite asymptotique E est également comparable entre méthodes et tailles de modèles.

Malgré l'absence de gradients dans le réseau central du LLM, Ladder présente un exposant similaire à celui de QLoRA, avec un écart constant en décalage.

4.3 Mise à l'Échelle et Mémoire

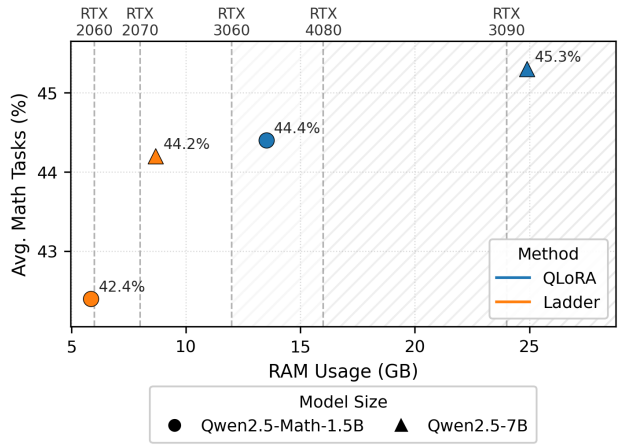
L'affinage de LLMs est gourmande en mémoire. La mémoire GPU (VRAM) nécessaire à l'entraînement peut être décomposée en trois termes principaux : $\mathbf{M}_1$: les poids du modèle, $\mathbf{M}_2$: les états de l'optimiseur, et $\mathbf{M}_3$: les activations intermédiaires. Les méthodes de PEFT visent à réduire le coût d'entraînement en diminuant un ou plusieurs de ces termes.

LoRA (Hu *et al.*, 2022) réduit $\mathbf{M}_2$ en ne mettant à jour qu'un petit ensemble de paramètres adaptateurs de rang faible, tandis que QLoRA (Dettmers *et al.*, 2023) réduit en plus $\mathbf{M}_1$ en quantifiant les poids (gelés) du backbone. Cependant, dans les deux cas, les activations intermédiaires du modèle $\mathbf{M}_3$ sont

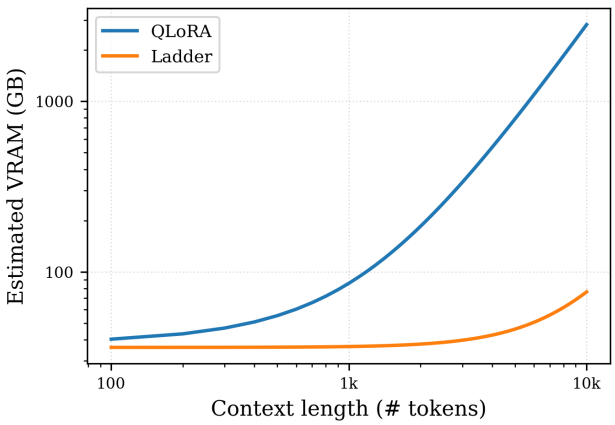
3. https://en.wikipedia.org/wiki/Huber_loss

toujours matérialisées en (quasi) pleine précision, ce qui domine l'empreinte mémoire. En pratique, $\mathbf{M}_3$ constitue souvent le terme le plus important lors de l'affinage.

Les approches Ladder telles que LST (Sung *et al.*, 2022) et QST (Zhang *et al.*, 2024b) utilisent un modèle latéral léger afin de réduire l'empreinte mémoire des états de l'optimiseur ($\mathbf{M}_2$) et des activations intermédiaires ($\mathbf{M}_3$). QST réduit en outre $\mathbf{M}_1$ en quantifiant les poids du backbone, tandis que LST conserve des poids en pleine précision.



(a) Frontière mémoire et performance sur une moyenne de 3 jeux de données en mathématiques. L'ensemble de ces entraînement a été réalisé avec une batch size de 1 et un contexte de 2k tokens. Les estimations d'usage de RAM ont été estimées sans sauvegarde des activations.



(b) Mémoire GPU requis par QLoRA et Ladder pour l'entraînement d'un Llama3.1-70B. Nous faisons l'hypothèse d'utiliser 8-bit AdamW, vanilla attention et pas de sauvegarde des activations.

FIGURE 3 – Analyse mémoire entre Ladder et QLoRA.

Afin d'assurer une comparaison équitable, nous évaluons QLoRA et Ladder sous la même configuration d'entraînement (taille de batch et longueur de séquence identiques). La Figure 3a montre qu'à taille de modèle comparable, QLoRA surpasse légèrement Ladder sur MATH-500, mais au prix d'un coût de calcul plus élevé (voir Section 4.2) et d'une consommation de VRAM substantiellement supérieure. En pratique, l'affinage d'un modèle 7B avec Ladder devient possible sur des GPU grand public d'entrée de gamme, alors qu'il peut être impossible avec QLoRA sur le même matériel.

Pour garantir l'équité des comparaisons, nous limitons notre analyse à l'attention « vanilla ». Il existe de nombreuses techniques permettant d'optimiser la mémoire (Sliding Window Attention, Grouped Query Attention (GQA) (Ainslie *et al.*, 2023), Flash Attention 1–3 (Dao *et al.*, 2022), PagedAttention (Kwon *et al.*, 2023), RadixAttention (Zheng *et al.*, 2024), FlexAttention (Dong *et al.*, 2025), mise en cache et déchargement NVMe [4]), mais elles reposent sur des hypothèses et impliquent des coûts différents ; une comparaison exhaustive dépasse donc le cadre de cette section.

4. Voir par exemple https://github.com/Mega4alik/ollm

5 Évaluation et Résultats

5.1 Tâche de Raisonement en Mathématiques

Configuration expérimentale. Nous entraînons un adaptateur Ladder sur les 1 000 échantillons d'entraînement du jeu de données s1K-1.1 (Muennighoff *et al.*, 2025), qui constitue une version nettoyée de 59 000 exemples de raisonnement mathématique provenant de sources multiples et utilisant les traces de modèles de raisonnement de plus grande taille comme réponses. La longueur de contexte sur ce jeu de données est limitée à 2 122 tokens, correspondant à l'exemple le plus long de l'ensemble d'entraînement.

Les hyperparamètres sont choisis sur la base de la validation et de l'exactitude mesurée sur un sous-ensemble du jeu de données EvoLM (Qi *et al.*, 2026). Le prompt utilisé pour l'entraînement et l'évaluation impose une réponse au format `\boxed`.

Nous évaluons sur le jeu de données MATH-500 (Hendrycks *et al.*, 2021), un benchmark composé de problèmes de mathématiques de niveau compétition et de difficulté variée (nous utilisons les mêmes 500 échantillons sélectionnés par OpenAI dans des travaux antérieurs (Lightman *et al.*, 2024)), ainsi que sur les jeux de données AIME'24 et AIME'25 (Mathematical Association of America, 2024), contenant chacun 30 problèmes de mathématiques de niveau lycée. Nous évaluons également sur le jeu de données AMC'23 (American Mathematics Competitions, 2023), qui comprend 40 exemples issus de compétitions mathématiques, sur le jeu de données Minerva Math (Lewkowycz *et al.*, 2022), contenant 272 problèmes de niveau licence, ainsi que sur le jeu de données OlympiadBench (He *et al.*, 2024), composé de 674 exemples.

Conformément à Hochlehnert *et al.* (2025) concernant les métriques d'évaluation, nous rapportons l'exactitude Pass@1, correspondant au pourcentage de problèmes résolus correctement dès la première tentative. Tous les jeux de données sont évalués sur 5 exécutions aléatoires afin d'assurer la robustesse des résultats, qui sont ensuite moyennés.

Nous évaluons en décodage glouton (soit, prendre le token ayant la plus haute probabilité) en fixant `max_new_tokens` à 2 048 tokens, en utilisant Math-Verify (HuggingFace, 2025), conformément aux travaux antérieurs indiquant que ce budget est suffisant pour ces tâches (Yang *et al.*, 2025).

Nous utilisons Qwen2.5-Math-1.5B (Yang *et al.*, 2024), et Qwen2.5-7B (Qwen *et al.*, 2025) comme modèles pour affinage.

Résultats. Le Tableau 2 compare nos modèles Ladder à QLoRA et à d'autres bases de référence publiées sur plusieurs benchmarks de raisonnement mathématique. Les scores pour s1.1-7B sont directement issus du classement Sober (Hochlehnert *et al.*, 2025), tandis que les résultats Full SFT proviennent de Wang *et al.* (2025), qui ne rapportent pas d'écarts-types. De plus, Qwen2.5-7B a été évalué après un apprentissage par renforcement. Les deux configurations expérimentales diffèrent légèrement de la nôtre, notamment par une longueur de génération de 32 768 tokens, nettement supérieure à nos 2 048 tokens.

Malgré ces différences : sur Qwen2.5-7B, Ladder améliore le Pass@1 moyen sur AIME25, AMC23 et OlympiadBench, mais est en retrait sur MATH-500 et Minerva ; sur Qwen2.5-Math-1.5B, il progresse sur AIME24/25 et AMC23/Minerva, mais sous-performe sur MATH-500 et OlympiadBench. Globalement, Ladder reste proche de QLoRA, tout en présentant des compromis dépendants du

Model	Pure SFT ?	MATH-500	AIME24	AIME25	AMC23	Minerva	Olympiad Bench
Based on Qwen2.5-7B							
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 with prior RL (∗) (Wang *et al.*, 2025)	✗	58.6	10.0	7.1	45.3	24.6	27.6
QLoRA	✓	**70.4** ± 1.4	**8.7** ± 1.8	6.0 ± 4.3	41.5 ± 1.4	**33.5** ± 1.3	32.0 ± 0.8
Ladder (ours)	✓	68.9 ± 1.6	8.0 ± 1.8	**8.0** ± 1.8	**47.5** ± 7.3	29.3 ± 1.2	**34.3** ± 0.5
Based on Qwen2.5-Math-1.5B							
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	✓	**70.4** ± 0.5	6.7 ±2.4	6.7 ± 3.3	45.0 ± 2.5	29.1 ± 0.6	**33.7** ± 1.6
Ladder (ours)	✓	67.2 ±1.4	**8.0** ± 5.1	**8.0** ± 3.0	**46.0** ± 4.9	**29.4** ± 1.8	32.7 ± 1.8

TABLE 2 – Résultats des expériences sur différents benchmarks de raisonnement mathématique. Nous rapportons la précision Pass@1 (moyenne ± écart-type) de toutes les méthodes sur 5 exécutions aléatoires. En (∗) les résultats issus de leurs articles originaux avec leur écart-type correspondant. Nous notons que certains modèles basés sur l'apprentissage par renforcement (RL) sont utilisés comme référence, tandis que le nôtre repose uniquement sur l'apprentissage supervisé (SFT).

benchmark : la plupart des écarts ne sont pas robustes selon les exécutions, tandis que nous observons des déficits constants sur MATH-500 et Minerva, ainsi qu'un gain constant sur OlympiadBench pour le modèle 7B.

Des travaux antérieurs suggèrent que les méthodes basées sur l'apprentissage par renforcement (RL) peuvent produire des gains plus importants que celles fondées sur le SFT pour les tâches de raisonnement (Chu *et al.*, 2025; Luo *et al.*, 2025; Trung *et al.*, 2024). Combiner Ladder avec le RL pourrait constituer une piste prometteuse pour démocratiser les modèles de raisonnement sur des GPU grand public à mémoire limitée. Toutefois, cela nécessiterait d'adapter des outils optimisés pour l'inférence, tels que SGLang (Zheng *et al.*, 2024) et vLLM (Kwon *et al.*, 2023), à l'architecture Ladder ; nous laissons cette direction de recherche à des travaux futurs.

Sur les benchmarks de raisonnement mathématique, Ladder peut être compétitif par rapport à QLoRA. Son principal avantage réside sur l'opportunité de pouvoir entraîner sur des GPU grand public.

5.2 Évaluation de xLadder

Nous comparons Ladder et xLadder sur le même benchmark de raisonnement mathématique en utilisant une profondeur l=4 avec Qwen2.5-Math-1.5B. Ladder@4 se connecte aux quatre premières couches du backbone ($\mathcal{C} = i \rightarrow i_{1 \leq i \leq 4}$), tandis que xLadder@4 ne se connecte qu'aux deux premières couches et ajoute deux couches supplémentaires dans le réseau latéral ($\mathcal{C} = i \rightarrow i_{1 \leq i \leq 2}$). Les deux variantes ont donc le même nombre de paramètres. Du fait de la proximité d'architecture entre les Ladder et xLadder, les équations de FLOPS (en Section 4.2) de Ladder restent vraies pour xLadder. Ainsi, nous nous attendons à ce que les tendances observées pour les lois d'échelle avec Ladder se généralisent à xLadder, à des constantes près. Cette hypothèse reste toutefois à confirmer empiriquement.

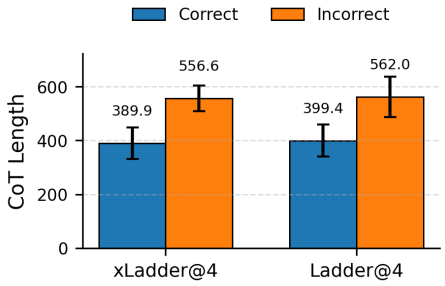


FIGURE 4 – Longueur moyenne des CoT sur 3 jeux de données mathématiques pour les réponses correctes et incorrectes.

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

TABLE 3 – Résultats de Ladder@4 et xLadder@4 sur un benchmark de raisonnement mathématique. Le modèle de base est Qwen2.5-Math-1.5B, et Ladder est connecté aux 4 premières couches du modèle de base, tandis que xLadder est connecté aux 2 premières couches avec 2 couches supplémentaires. Les résultats sont moyennés sur 5 entraînements.

Le Tableau 3 et la Figure 4 montrent que xLadder améliore l'exactitude sur la plupart des jeux de données et tend à produire des chaînes de pensée (CoT) plus courtes, aussi bien pour les réponses correctes qu'incorrectes. Cela est cohérent avec des observations antérieures selon lesquelles une CoT plus longue n'est pas toujours bénéfique et peut être corrélée à des erreurs (Wu *et al.*, 2025), suggérant que xLadder augmente la profondeur effective du raisonnement sans nécessiter davantage de tokens générés. Toutefois, il faudrait faire davantage d'expériences pour tirer des conclusions sûres sur cela.

Plus précisément, nous rapportons dans l'Appendice A des résultats comparatifs sur CoLA ainsi que sur un jeu de données original de critique par LLM. Le comportement de xLadder observé sur ces jeux de données est cohérent avec les observations précédentes.

À nombre de paramètres donné, ajuster les connexions inter-couches dans xLadder afin d'augmenter la profondeur du passage avant peut améliorer les performances et réduire le nombre de tokens générés dans la chaîne de pensée.

6 Conclusion

Nous revisitons les adaptateurs Ladder et mettons en évidence leurs principales propriétés : leur relation calcul–performance suit une tendance similaire à celle de QLoRA, souvent avec un décalage vertical.

Notre analyse mémoire et nos évaluations sur des benchmarks de raisonnement indiquent que Ladder est compétitif tout en étant plus efficace en termes de mémoire. Nous introduisons xLadder, qui approfondit le raisonnement à chaque pas de temps et tend ainsi à réduire la longueur des chaînes de pensée (CoT).

Bien qu'une validation plus large sur davantage de modèles et de tâches soit encore nécessaire, l'extension de profondeur au moment de l'affinage apparaît comme une nouvelle piste de recherche pour améliorer le raisonnement, en complément des stratégies de préentraînement et d'inférence (test-time).

Dans des travaux futurs, nous prévoyons d'explorer d'autres schémas de connexions et d'élargir l'ensemble des modèles et des tâches étudiés.

Remerciements

Ces travaux ont bénéficié d'un accès aux ressources de calcul en HPC et de stockage à l'IDRIS au travers de l'allocation de ressources 2025-011011668R5 attribuée par GENCI sur la partition A100 du calculateur Jean Zay.

Références

ABDIN M., ANEJA J., BEHL H., BUBECK S., ELDAN R., GUNASEKAR S., HARRISON M., HEWETT R. J., JAVAHERIPI M., KAUFFMANN P., LEE J. R., LEE Y. T., LI Y., LIU W., MENDES C. C. T., NGUYEN A., PRICE E., DE ROSA G., SAARIKIVI O., SALIM A., SHAH S., WANG X., WARD R., WU Y., YU D., ZHANG C. & ZHANG Y. (2024). *Phi-4 Technical Report*. Rapport interne, Microsoft Research.

AINSLIE J., LEE-THORP J., DE JONG M., ZEMLYANSKIY Y., LEBRON F. & SANGHAI S. (2023). GQA : Training generalized multi-query transformer models from multi-head checkpoints. In H. BOUAMOR, J. PINO & K. BALI, Éds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 4895–4901, Singapore : Association for Computational Linguistics. DOI : 10.18653/v1/2023.emnlp-main.298.

AMERICAN MATHEMATICS COMPETITIONS (2023). American mathematics contest 12. https://huggingface.co/datasets/AI-MO/aimo-validation-amc. Accessed : 2025-06-25.

BAI Y., KADAVATH S., KUNDU S., ASKELL A., KERNION J., JONES A., CHEN A., GOLDIE A., MIRHOSEINI A., MCKINNON C., CHEN C., OLSSON C., OLAH C., HERNANDEZ D., DRAIN D., GANGULI D., LI D., TRAN-JOHNSON E., PEREZ E., KERR J., MUELLER J., LADISH J., LANDAU J., NDOUSSE K., LUKOSUITE K., LOVITT L., SELLITTO M., ELHAGE N., SCHIEFER N., MERCADO N., DASSARMA N., LASENBY R., LARSON R., RINGER S., JOHNSTON S., KRAVEC S., SHOWK S. E., FORT S., LANHAM T., TELLEEN-LAWTON T., CONERLY T., HENIGHAN T., HUME T., BOWMAN S. R., HATFIELD-DODDS Z., MANN B., AMODEI D., JOSEPH N., MCCANDLISH S., BROWN T. & KAPLAN J. (2022). Constitutional ai : Harmlessness from ai feedback.

CHEN F. (2024). Cst : Calibration side-tuning for parameter and memory efficient transfer learning.

CHEN T., XU B., ZHANG C. & GUESTRIN C. (2016). Training deep nets with sublinear memory cost.

CHEN Z., DENG Y., YUAN H., JI K. & GU Q. (2024). Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning*.

CHU T., ZHAI Y., YANG J., TONG S., XIE S., SCHUURMANS D., LE Q. V., LEVINE S. & MA Y. (2025). SFT memorizes, RL generalizes : A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*.

COBBE K., KOSARAJU V., BAVARIAN M., CHEN M., JUN H., KAISER L., PLAPPERT M., TWOREK J., HILTON J., NAKANO R., HESSE C. & SCHULMAN J. (2021). Training verifiers to solve math word problems.

DAO T., FU D., ERMON S., RUDRA A. & RÉ C. (2022). Flashattention : Fast and memory-efficient exact attention with io-awareness. In S. KOYEJO, S. MOHAMED, A. AGARWAL, D. BELGRAVE, K. CHO & A. OH, Éds., *Advances in Neural Information Processing Systems*, volume 35, p. 16344–16359 : Curran Associates, Inc.

DEEPSEEK-AI, GUO D. *et al.* (2025). *DeepSeek-R1 : Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. Rapport interne, DeepSeek-AI.

DETTMERS T., PAGNONI A., HOLTZMAN A. & ZETTLEMOYER L. (2023). QLoRA : Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

DONG J., FENG B., GUESSOUS D., LIANG Y. & HE H. (2025). Flexattention : A programming model for generating fused attention variants. In *Eighth Conference on Machine Learning and Systems*.

FRANKLE J., DZIUGAITE G. K., ROY D. & CARBIN M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In H. D. III & A. SINGH, Éds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 de *Proceedings of Machine Learning Research*, p. 3259–3269 : PMLR.

GOMEZ A. N., REN M., URTASUN R. & GROSSE R. B. (2017). The reversible residual network : Backpropagation without storing activations. In I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN & R. GARNETT, Éds., *Advances in Neural Information Processing Systems*, volume 30 : Curran Associates, Inc.

GRATTAFIORI A., DUBEY A. *et al.* (2024). *The Llama 3 Herd of Models*. Rapport interne, Meta AI.

GUHA E. K., MARTEN R., KEH S., RAOOF N., SMYRNIS G., BANSAL H., NEZHURINA M., MERCAT J., VU T., SPRAGUE Z. R., SUVARNA A., FEUER B., CHEN L. L., KHAN Z., FRANKEL E., GROVER S., CHOI C., MUENNIGHOFF N., SU S., ZHAO W., YANG J., PIMPALGAONKAR S., SHARMA K., JI C. C.-J., DENG Y., PRATT S. M., RAMANUJAN V., SAAD-FALCON J., ACHARYA S., LI J., DAVE A., ALBALAK A., ARORA K., WULFE B., HEGDE C., DURRETT G., OH S., BANSAL M., GABRIEL S., GROVER A., CHANG K.-W., SHANKAR V., GOKASLAN A., MERRILL M. A., HASHIMOTO T., CHOI Y., JITSEV J., HECKEL R., SATHIAMOORTHY M., DIMAKIS A. & SCHMIDT L. (2026). Openthoughts : Data recipes for reasoning models. In *The Fourteenth International Conference on Learning Representations*.

HE C., LUO R., BAI Y., HU S., THAI Z., SHEN J., HU J., HAN X., HUANG Y., ZHANG Y., LIU J., QI L., LIU Z. & SUN M. (2024). OlympiadBench : A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In L.-W. KU, A. MARTINS & V. SRIKUMAR, Éds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 3828–3850, Bangkok, Thailand : Association for Computational Linguistics. DOI : 10.18653/v1/2024.acl-long.211.

HENDRYCKS D., BURNS C., KADAVATH S., ARORA A., BASART S., TANG E., SONG D. & STEINHARDT J. (2021). Measuring mathematical problem solving with the math dataset.

HINTON G., VINYALS O. & DEAN J. (2015). Distilling the knowledge in a neural network.

HOCHLEHNERT A., BHATNAGAR H., UDANDARAO V., ALBANIE S., PRABHU A. & BETHGE M. (2025). A sober look at progress in language model reasoning : Pitfalls and paths to reproducibility.

HOULSBY N., GIURGIU A., JASTRZEBSKI S., MORRONE B., DE LAROUSSILHE Q., GESMUNDO A., ATTARIYAN M. & GELLY S. (2019). Parameter-efficient transfer learning for NLP. In K. CHAUDHURI & R. SALAKHUTDINOV, Éds., *Proceedings of the 36th International Conference*

on Machine Learning, volume 97 de *Proceedings of Machine Learning Research*, p. 2790–2799 : PMLR.

HU E. J., YELONG SHEN, WALLIS P., ALLEN-ZHU Z., LI Y., WANG S., WANG L. & CHEN W. (2022). LoRA : Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

HUGGINGFACE (2025). Math-verify. Accessed : 2025-06-25.

JAWAHAR G., SAGOT B. & SEDDAH D. (2019). What does BERT learn about the structure of language ? In A. KORHONEN, D. TRAUM & L. MÀRQUEZ, Éds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 3651–3657, Florence, Italy : Association for Computational Linguistics. DOI : 10.18653/v1/P19-1356.

KAPLAN J., MCCANDLISH S., HENIGHAN T., BROWN T. B., CHESS B., CHILD R., GRAY S., RADFORD A., WU J. & AMODEI D. (2020). Scaling laws for neural language models. arXiv : 2001.08361.

KE P., WEN B., FENG A., LIU X., LEI X., CHENG J., WANG S., ZENG A., DONG Y., WANG H., TANG J. & HUANG M. (2024). CritiqueLLM : Towards an informative critique generation model for evaluation of large language model generation. In L.-W. KU, A. MARTINS & V. SRIKUMAR, Éds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 13034–13054, Bangkok, Thailand : Association for Computational Linguistics. DOI : 10.18653/v1/2024.acl-long.704.

KITAEV N., KAISER L. & LEVSKAYA A. (2020). Reformer : The efficient transformer. In *International Conference on Learning Representations*.

KWON W., LI Z., ZHUANG S., SHENG Y., ZHENG L., YU C. H., GONZALEZ J. E., ZHANG H. & STOICA I. (2023). Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

LEE H., PHATALE S., MANSOOR H., MESNARD T., FERRET J., LU K. R., BISHOP C., HALL E., CARBUNE V., RASTOGI A. & PRAKASH S. (2024). RLAIF vs. RLHF : Scaling reinforcement learning from human feedback with AI feedback. In R. SALAKHUTDINOV, Z. KOLTER, K. HELLER, A. WELLER, N. OLIVER, J. SCARLETT & F. BERKENKAMP, Éds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 de *Proceedings of Machine Learning Research*, p. 26874–26901 : PMLR.

LESTER B., AL-RFOU R. & CONSTANT N. (2021). The power of scale for parameter-efficient prompt tuning. In M.-F. MOENS, X. HUANG, L. SPECIA & S. W.-T. YIH, Éds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 3045–3059, Online and Punta Cana, Dominican Republic : Association for Computational Linguistics. DOI : 10.18653/v1/2021.emnlp-main.243.

LEWKOWYCZ A., ANDREASSEN A. J., DOHAN D., DYER E., MICHALEWSKI H., RAMASESH V. V., SLONE A., ANIL C., SCHLAG I., GUTMAN-SOLO T., WU Y., NEYSHABUR B., GUR-ARI G. & MISRA V. (2022). Solving quantitative reasoning problems with language models. In A. H. OH, A. AGARWAL, D. BELGRAVE & K. CHO, Éds., *Advances in Neural Information Processing Systems*.

LI J., BEECHING E., TUNSTALL L., LIPKIN B., SOLETSKYI R., HUANG S. C., RASUL K., YU L., JIANG A., SHEN Z., QIN Z., DONG B., ZHOU L., FLEUREAU Y., LAMPLE G. & POLU S. (2024). Numinamath.

LI X. L. & LIANG P. (2021). Prefix-tuning : Optimizing continuous prompts for generation. In C. ZONG, F. XIA, W. LI & R. NAVIGLI, Éds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, p. 4582–4597, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.acl-long.353.

LIGHTMAN H., KOSARAJU V., BURDA Y., EDWARDS H., BAKER B., LEE T., LEIKE J., SCHULMAN J., SUTSKEVER I. & COBBE K. (2024). Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

LIU H., TAM D., MOHAMMED M., MOHTA J., HUANG T., BANSAL M. & RAFFEL C. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In A. H. OH, A. AGARWAL, D. BELGRAVE & K. CHO, Éds., *Advances in Neural Information Processing Systems*.

LUO H., SUN Q., XU C., ZHAO P., LOU J.-G., TAO C., GENG X., LIN Q., CHEN S., TANG Y. & ZHANG D. (2025). Wizardmath : Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *The Thirteenth International Conference on Learning Representations*.

MALLADI S., GAO T., NICHANI E., DAMIAN A., LEE J. D., CHEN D. & ARORA S. (2023). Fine-tuning language models with just forward passes. In A. OH, T. NAUMANN, A. GLOBERSON, K. SAENKO, M. HARDT & S. LEVINE, Éds., *Advances in Neural Information Processing Systems*, volume 36, p. 53038–53075 : Curran Associates, Inc.

MATHEMATICAL ASSOCIATION OF AMERICA (2024). American invitational mathematics examination. https://artofproblemsolving.com/wiki/index.php?title=AIME_Problems_and_Solutions. Accessed : 2025-06-25.

MUENNIGHOFF N., YANG Z., SHI W., LI X. L., FEI-FEI L., HAJISHIRZI H., ZETTLEMOYER L., LIANG P., CANDÈS E. & HASHIMOTO T. (2025). s1 : Simple test-time scaling. In C. CHRISTODOULOPOULOS, T. CHAKRABORTY, C. ROSE & V. PENG, Éds., *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, p. 20275–20321, Suzhou, China : Association for Computational Linguistics. DOI : 10.18653/v1/2025.emnlp-main.1025.

QI Z., NIE F., ALAHI A., ZOU J., LAKKARAJU H., DU Y., XING E. P., KAKADE S. M. & ZHANG H. (2026). EvoLM : In search of lost training dynamics for language model reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

QWEN, :, YANG A., YANG B., ZHANG B., HUI B., ZHENG B., YU B., LI C., LIU D., HUANG F., WEI H., LIN H., YANG J., TU J., ZHANG J., YANG J., YANG J., ZHOU J., LIN J., DANG K., LU K., BAO K., YANG K., YU L., LI M., XUE M., ZHANG P., ZHU Q., MEN R., LIN R., LI T., TANG T., XIA T., REN X., REN X., FAN Y., SU Y., ZHANG Y., WAN Y., LIU Y., CUI Z., ZHANG Z. & QIU Z. (2025). *Qwen2.5 Technical Report*. Rapport interne, Qwen Team.

REN J., RAJBHANDARI S., AMINABADI R. Y., RUWASE O., YANG S., ZHANG M., LI D. & HE Y. (2021). ZeRO-Offload : Democratizing Billion-Scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, p. 551–564 : USENIX Association.

SANH V., DEBUT L., CHAUMOND J. & WOLF T. (2020). Distilbert, a distilled version of bert : smaller, faster, cheaper and lighter.

SUNG Y.-L., CHO J. & BANSAL M. (2022). LST : Ladder side-tuning for parameter and memory efficient transfer learning. In A. H. OH, A. AGARWAL, D. BELGRAVE & K. CHO, Éds., *Advances in Neural Information Processing Systems*.

TANG N., FU M., ZHU K. & WU J. (2024). Low-rank attention side-tuning for parameter-efficient fine-tuning.

TOSHNIWAL S., DU W., MOSHKOV I., KISACANIN B., AYRAPETYAN A. & GITMAN I. (2025). Openmathinstruct-2 : Accelerating AI for math with massive open-source instruction data. In *The Thirteenth International Conference on Learning Representations*.

TRUNG L., ZHANG X., JIE Z., SUN P., JIN X. & LI H. (2024). ReFT : Reasoning with reinforced fine-tuning. In L.-W. KU, A. MARTINS & V. SRIKUMAR, Éds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 7601–7614, Bangkok, Thailand : Association for Computational Linguistics. DOI : 10.18653/v1/2024.acl-long.410.

VOITA E., SENNRICH R. & TITOV I. (2019). The bottom-up evolution of representations in the transformer : A study with machine translation and language modeling objectives. In K. INUI, J. JIANG, V. NG & X. WAN, Éds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 4396–4406, Hong Kong, China : Association for Computational Linguistics. DOI : 10.18653/v1/D19-1448.

WANG Y., NIE P., ZOU K., WU L. & CHEN W. (2025). Unleashing the reasoning potential of LLMs by critique fine-tuning on one problem. In C. CHRISTODOULOPOULOS, T. CHAKRABORTY, C. ROSE & V. PENG, Éds., *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, p. 3017–3027, Suzhou, China : Association for Computational Linguistics. DOI : 10.18653/v1/2025.emnlp-main.149.

WU Y., WANG Y., YE Z., DU T., JEGELKA S. & WANG Y. (2025). When more is less : Understanding chain-of-thought length in llms.

YANG A., ZHANG B., HUI B., GAO B., YU B., LI C., LIU D., TU J., ZHOU J., LIN J., LU K., XUE M., LIN R., LIU T., REN X. & ZHANG Z. (2024). *Qwen2.5-Math Technical Report : Toward Mathematical Expert Model via Self-Improvement*. Rapport interne, Qwen Team.

YANG W., LIAO M. & FAN K. (2025). Markov chain of thought for efficient mathematical reasoning. In L. CHIRUZZO, A. RITTER & L. WANG, Éds., *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)*, p. 7132–7157, Albuquerque, New Mexico : Association for Computational Linguistics. DOI : 10.18653/v1/2025.naacl-long.365.

YE Y., HUANG Z., XIAO Y., CHERN E., XIA S. & LIU P. (2025). LIMO : Less is more for reasoning. In *Second Conference on Language Modeling*.

YU L., JIANG W., SHI H., YU J., LIU Z., ZHANG Y., KWOK J., LI Z., WELLER A. & LIU W. (2024). Metamath : Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

YUAN L., LI W., CHEN H., CUI G., DING N., ZHANG K., ZHOU B., LIU Z. & PENG H. (2025). Free process rewards without process labels. In *Forty-second International Conference on Machine Learning*.

ZHANG B., LIU Z., CHERRY C. & FIRAT O. (2024a). When scaling meets LLM finetuning : The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*.

ZHANG S., ROLLER S., GOYAL N., ARTETXE M., CHEN M., CHEN S., DEWAN C., DIAB M., LI X., LIN X. V., MIHAYLOV T., OTT M., SHLEIFER S., SHUSTER K., SIMIG D., KOURA P. S., SRIDHAR A., WANG T. & ZETTLEMOYER L. (2022). *OPT : Open Pre-trained Transformer Language Models*. Rapport interne, Meta AI.

ZHANG Z., ZHAO D., MIAO X., OLIARO G., ZHANG Z., LI Q., JIANG Y. & JIA Z. (2024b). Quantized side tuning : Fast and memory-efficient tuning of quantized large language models. In L.-W. KU, A. MARTINS & V. SRIKUMAR, Éds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1–17, Bangkok, Thailand : Association for Computational Linguistics. DOI : 10.18653/v1/2024.acl-long.1.

ZHENG L., CHIANG W.-L., SHENG Y., ZHUANG S., WU Z., ZHUANG Y., LIN Z., LI Z., LI D., XING E., ZHANG H., GONZALEZ J. E. & STOICA I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. In A. OH, T. NAUMANN, A. GLOBERSON, K. SAENKO, M. HARDT & S. LEVINE, Éds., *Advances in Neural Information Processing Systems*, volume 36, p. 46595–46623 : Curran Associates, Inc.

ZHENG L., YIN L., XIE Z., SUN C., HUANG J., YU C. H., CAO S., KOZYRAKIS C., STOICA I., GONZALEZ J. E., BARRETT C. & SHENG Y. (2024). Sglang : Efficient execution of structured language model programs. In A. GLOBERSON, L. MACKEY, D. BELGRAVE, A. FAN, U. PAQUET, J. TOMCZAK & C. ZHANG, Éds., *Advances in Neural Information Processing Systems*, volume 37, p. 62557–62583 : Curran Associates, Inc.

A Expériences supplémentaires

A.1 Tâche de classification : LLM critique

Configuration. Nous construisons un jeu de données pour un critique LLM en échantillonnant à partir du jeu de données MATH-500 (Hendrycks *et al.*, 2021). Tout d'abord, plusieurs solveurs préentraînés, c'est-à-dire des LLM, sont invités à produire une chaîne de pensée (CoT) ainsi qu'une réponse pour chaque question du jeu de données MATH-500. Nous ne conservons que les générations dans lesquelles la réponse `\boxed{}` apparaît dans les 2 048 premiers tokens, puis nous vérifions la correction des réponses à l'aide de Math-Verify (HuggingFace, 2025). Le corpus obtenu est équilibré, avec une répartition 50/50 entre réponses correctes et incorrectes, ce qui conduit à un jeu de données de 380 échantillons d'entraînement et 396 échantillons de test, avec une longueur moyenne de 600 tokens. Ce jeu de données sera librement disponible sur HuggingFace.

Pour le critique lui-même, nous affinons les moèles Qwen2.5-0.5B et Qwen2.5-3B avec les méthodes Ladder et QLoRA pendant 3 époques, avec un taux d'apprentissage de 5E-4. Ces hyperparamètres ont été sélectionnés sur la base de la convergence de la courbe d'entraînement. La tâche est binaire : prédire si la CoT en entrée est correcte ou incorrecte.

Résultats. Le Tableau 4 présente les résultats de notre critique LLM sur l'ensemble de test. Les deux méthodes d'affinage améliorent significativement les performances quasi aléatoires des modèles de base, pour atteindre plus de 80% de précision. Comme les deux méthodes obtiennent des résultats similaires, nous pouvons conclure que la qualité des données d'entraînement et les contraintes d'implémentation, c'est-à-dire le nombre limité de GPU, sont plus importantes que la méthode d'affinage elle-même.

Model	Method	% Précision
Qwen2.5-0.5B	Base	59.0
Qwen2.5-0.5B	Ladder	**81.8**
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	**82.8**

TABLE 4 – Résultats du critique LLM sur 396 échantillons de test. L'intervalle de confiance de Wald à 95% est de $\pm 3.9\%$.

Les LLM sont de plus en plus utilisés pour évaluer la qualité des générations d'autres LLM, par exemple dans les tâches LLM-as-a-Judge (Zheng *et al.*, 2023), la modélisation de récompense pour l'affinage par RL (Lee *et al.*, 2024; Bai *et al.*, 2022), et les critiques de notation (Ke *et al.*, 2024). Notre critique s'inscrit dans cette tendance : il fournit un filtre rapide, précis à 82%, qui écarte la plupart des CoT erronées avant de transmettre celles dont la confiance est plus élevée à un processus plus coûteux, comme une évaluation humaine ou un affinage par RL.

Le critique présente une erreur résiduelle de 18%, et la taille modeste du jeu de données d'entraînement laisse une marge d'amélioration de la robustesse en cas de décalage de distribution, par exemple grâce à l'utilisation de LLM préentraînés plus diversifiés ou de prompts plus complexes.

A.2 Tâche de classification : CoLA

Configuration. Dans le benchmark GLUE, nous nous concentrons sur le jeu de données CoLA, qui est une tâche de classification binaire portant sur l'acceptabilité grammaticale des phrases. Nous utilisons les mêmes paramètres que ceux fournis dans les travaux antérieurs sur LST (Sung *et al.*, 2022).

Nous réimplémentons l'architecture, car le code original n'était pas compatible avec les versions récentes des bibliothèques Python. L'architecture LST originale est implémentée avec un réseau latéral connecté à chaque couche du modèle backbone, et elle est initialisée en élaguant les poids du modèle backbone. Nous entraînons pendant plusieurs époques sur deux jeux de données : 7 époques sur CoLA et 4 époques sur QQP. Le meilleur modèle est sélectionné sur la base de la perte de validation calculée à chaque époque. Ladder utilise une taille de batch de 4, tandis que QLoRA ne parvient pas à apprendre correctement avec une petite taille de batch ; nous utilisons donc une taille de batch de 32. Leur taux d'apprentissage est fixé à 1E-5 avec un scheduler cosinus. L'optimiseur utilisé est AdamW standard.

La configuration QLoRA est la même que précédemment, avec une quantification 4-bit et $r = 16$ et $\alpha = 32$ sur toutes les couches linéaires. L'architecture Ladder est un ladder entièrement connecté à chaque couche du modèle backbone, avec une initialisation uniforme du réseau latéral et de sa projection linéaire

Model	Method	Reproduction	Original
OPT1.3B	QLoRA	0.630	*0.621 ± 0.023*
OPT1.3B	LST	0.579	*0.595 ± 0.031*
OPT2.7B	QLoRA	0.656	*0.637 ± 0.026*
OPT2.7B	LST	0.592	*0.607 ± 0.035*
OPT6.7B	QLoRA	0.665	*0.643 ± 0.028*
OPT6.7B	LST	0.596	*non rapporté*

TABLE 5 – Reproduction des résultats rapportés dans (Sung *et al.*, 2022) sur le benchmark CoLA avec notre base de code Ladder. La métrique utilisée est le MCC.

À des fins de reproduction, les modèles utilisés appartiennent à la famille OPT (Zhang *et al.*, 2022), avec 1.3B, 2.7B et 6.7B paramètres. Des modèles plus récents, tels que Llama3.2-3B et Qwen2.5-1.5B, sont également utilisés pour comparer l'architecture Ladder à la méthode QLoRA.

Nous entraînons les modèles pendant 7 époques avec arrêt anticipé, un taux d'apprentissage de 1E-5, l'optimiseur AdamW et un scheduler cosinus. Tous les résultats sont des moyennes sur 4 entraînements. L'entraînement est effectué sur un seul GPU ADA RTX5000 32GB.

Résultats. En relançant les expériences LST originales sur le benchmark CoLA, nous observons une différence dans les résultats par rapport à l'article original, comme indiqué dans le Tableau 5. Cette différence est probablement due à l'utilisation de versions différentes des bibliothèques et des noyaux GPU, ce qui peut entraîner de légères variations dans les résultats.

Par ailleurs, en comparant Ladder à xLadder, nous observons que l'architecture xLadder semble surpasser systématiquement l'architecture Ladder, comme le montre le Tableau 6. Cela suggère que les couches et connexions supplémentaires de l'architecture xLadder apportent un gain de performance

Modèle	Méthode	MCC
Llama3.2-3B	QLoRA	**0.69**
Llama3.2-3B	Ladder	0.61
Llama3.2-3B	xLadder	0.64
Qwen2.5-1.5B	QLoRA	**0.62**
Qwen2.5-1.5B	Ladder	0.58
Qwen2.5-1.5B	xLadder	0.59

TABLE 6 – Résultats supplémentaires sur CoLA avec des modèles plus récents et un ladder plus simple : initialisation uniforme, portes de somme entre le ladder et le backbone avec $\alpha = 0.5$. L'intervalle de confiance de Wald à 95% est de ± 0.03.

par rapport à l'architecture Ladder standard. Enfin, lorsque nous comparons la méthode Ladder à QLoRA, nous observons que les méthodes fondées sur ladder obtiennent des résultats légèrement inférieurs à QLoRA sur la tâche de classification CoLA. Cela suggère que l'initialisation par élagage tend à donner des résultats légèrement meilleurs.

B Détails des Hyperparamètres

B.1 Hyperparamètres pour les lois à l'échelle

Paramètre	Valeur
Échantillons d'entraînement	$D_f = \{50k, 100k, 200k, 300k, 400k\}$
Nombre d'époques	1
Sélection du meilleur modèle	Perte de validation
Jeu de validation	Sous-ensemble d'EvoLM (Qi *et al.*, 2026)
Longueur de séquence	1 839 tokens
Taux d'apprentissage	2E-5
Scheduler	Cosinus, warmup ratio $= 0.1$
Optimiseur	AdamW 8-bit
Weight decay	0.01
Quantification QLoRA	4-bit
Configuration QLoRA	$r = 16$, $\alpha = 32$ sur toutes les couches linéaires
GPU	1 NVIDIA A100 80GB
Temps d'entraînement, 100k	$\sim$1h pour Ladder, $\sim$2h pour QLoRA
Modèle / méthode	**Micro-batch / Acc. gradient / Batch effectif**
Qwen2.5-Math-1.5B / Ladder	8 / 3 / 24
Qwen2.5-Math-1.5B / QLoRA	4 / 5 / 20
Qwen2.5-7B / Ladder	4 / 3 / 12
Qwen2.5-7B / QLoRA	2 / 5 / 10

TABLE 7 – Hyperparamètres des expériences de mise à l'échelle de la perte de test.

B.2 Hyperparamètres pour les tâches de raisonnement mathématique

Paramètre	**Valeur**
Nombre d'époques	3
Évaluation	Après chaque époque sur l'ensemble de validation
Longueur de séquence	2 122 tokens
Quantification du backbone	8-bit
Architecture Ladder	Ladder entièrement connecté à chaque couche du backbone
Initialisation Ladder	Uniforme
Somme à porte	Non pondérée
GPU	1 NVIDIA A100 80GB
Modèle / méthode	**Batch size / Taux d'apprentissage**
Qwen2.5-Math-1.5B / Ladder	8 / 2E-4
Qwen2.57B / Ladder	4 / 2E-4
Qwen2.5-Math-1.5B / QLoRA	4 / 2E-5
Qwen2.57B / QLoRA	2 / 2E-5

TABLE 8 – Hyperparamètres d'entraînement pour les expériences de raisonnement mathématique.