Ignore the KL Penalty! Boosting Exploration on Critical Tokens to Enhance RL Fine-Tuning

Jean Vassoyan[1,2] **Nathanaël Beau**[2,3] **Roman Plaud**[2,4]

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, France
[2]onepoint, France [3]Université de Paris, LLF, CNRS, France
[4] Institut Polytechnique de Paris

jean.vassoyan@ens-paris-saclay.fr
nathanael.beau.gs@gmail.com
roman.plaud@telecom-paris.fr

Abstract

The ability to achieve long-term goals is a key challenge in the current development of large language models (LLMs). To address this, pre-trained LLMs can be fine-tuned with reinforcement learning (RL) to explore solutions that optimize a given goal. However, exploration with LLMs is difficult, as a balance has to be struck between discovering new solutions and staying close enough to the pre-trained model, so as not to degrade basic capabilities. This is typically controlled with a Kullback-Leibler (KL) penalty. In this paper, we investigate the exploration dynamics of a small language model on a simple arithmetic task. We show how varying degrees of pre-training influence exploration and demonstrate the importance of "critical tokens" which have a dramatic impact on the final outcome. Consequently, we introduce a simple modification to the KL penalty that favors exploration on critical tokens, increasing the efficiency of the RL fine-tuning stage. [1]

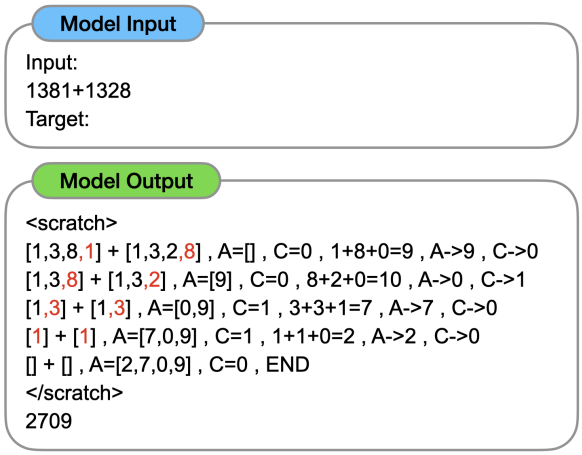


Figure 1: Illustration of the addition task with scratchpad, for a model pre-trained on numbers up to 3 digits. The highlighted critical tokens are decision points where the model tends to make mistakes, mainly because it is tempted to process the number as if it were shorter. This occurs when the model is faced with a number that is longer than those encountered during the pre-training stage (here, 4 digits instead of 3).

1 Introduction

In recent years, expectations on large language models (LLMs) have evolved, viewing them more and more as agents intended to achieve long-term goals (Wei et al., 2022; Bellos et al., 2024; Havrilla et al., 2024). In particular, a number of research studies have found that LLMs can learn to achieve long-term objectives when fine-tuned with Reinforcement Learning (RL), even with a sparse success/failure signal (Bakhtin et al., 2022; Zelikman et al., 2024; Havrilla et al., 2024; Guo et al., 2025). In such setting, a pre-trained language model is typically used as a policy to explore solutions within a text-generation task. Pre-training plays an ambivalent role in guiding exploration: on the one hand, the policy should not deviate too far from the pre-trained model in order to maintain basic capabilities (like language structure) – this is why a KL-divergence penalty is typically added to the loss (Ziegler et al., 2019). On the other hand, staying too close to the pre-trained model can significantly hinder its potential for exploration. On this matter (Havrilla et al., 2024) have demonstrated that LLM agents typically fail to explore beyond solutions produced by the pre-trained models. We hypothesize that more precisely balancing the trade-off between old and new policies can improve the model's exploration capabilities, especially as the distribution shift increases between pre-training and fine-tuning.

This article examines how varying levels of pre-training affect language model performance in a task requiring some level of exploration. We introduce an experimental setup where the model is first pre-trained on a simple arithmetic task, then fine-tuned with RL on a similar task with a small

[1]Our code and experiments are publicly available at: https://github.com/jvasso/llm-rl-arithmetic.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 6123–6133

distribution shift. We chose the arithmetic task for two main reasons. First, prior research highlights the value of studying language models on basic arithmetic problems (Liu and Low, 2023; Zhou et al., 2024), noting challenges in generalizing to novel digit lengths — though these difficulties vary by model type and use of scratchpads (Yuan et al., 2023; Lee et al., 2024). Second, this task closely mirrors real-world LLM applications while enabling fine-grained control over the distribution shift between pre-training and RL fine-tuning stages. Notably, we find that performance on this RL task is determined by a few critical tokens where the policy must diverge from the pre-trained model's predictions. This observation motivated a modification to the original KL penalty, making it more dependent on the pre-trained model's confidence.

Our contribution is two-fold: we first conduct an analysis of the influence of pre-training on a small language model's ability to explore out-of-distribution. More precisely, we investigate how pre-training with a broader range of operand lengths influences the model's performance on new operand lengths. Second, we introduce a simple trick that allows to adapt the KL penalty to the token-wise confidence of the pre-trained model. Our empirical results show that this modification to the KL penalty substantially enhances exploration efficiency.

2 Related Work

LLMs and Reasoning Recent state-of-the-art LLMs (Touvron et al., 2023; OpenAI, 2023) have shown strong performance on reasoning tasks across various benchmarks, including mathematics (Cobbe et al., 2021; Hendrycks et al., 2021) and code (Chen et al., 2021; Li et al., 2022). Combining LLMs with prompting strategies like chain-of-thought (Wei et al., 2022) has become a common approach for tackling complex reasoning tasks by guiding the model to break down problems into smaller subproblems.

LLMs and RL The integration of LLMs and RL has primarily been driven by Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2020; Stiennon et al., 2020), which aligns model outputs with human preferences. However we stress that learning from human preferences is a different framework from the more general one of RL, as the latter focuses on optimizing long-term objectives – possibly with high level of exploration – while learning from human preferences can be achieved solely with a fixed dataset. RL has also been applied to LLMs in this more general framework, in tasks such as grounding (Yao et al., 2020; Carta et al., 2023), code generation (Le et al., 2022), and mathematical reasoning (Havrilla et al., 2024). Training LLMs with RL presents challenges due to reward sparsity (Cao et al., 2024), credit assignment difficulties in identifying key actions that led to failure (Hwang et al., 2024), large state spaces requiring exploration, and unstable training processes. Havrilla et al. (2024) have raised concerns about RL algorithms, struggling to explore beyond solutions already produced by supervised fine-tuning (SFT) models.

LLMs and Addition The addition task remains challenging even for the latest LLMs, which struggle to accurately add large numbers and track digit positions (Wallace et al., 2019). Most related studies have focused on supervised learning approaches (Lee et al., 2024; McLeish et al., 2024) and improving positional encoding (Shen et al., 2023; Kazemnejad et al., 2023; McLeish et al., 2024; Zhou et al., 2024). Generalization to unseen lengths is a common evaluation criterion in these studies (Kazemnejad et al., 2023; Xiao and Liu, 2023; Zhou et al., 2024). Despite the addition task being a reasoning problem with a well-defined long-term reward, no research, to our knowledge, has addressed it using RL with a language model. The closest work is by Zhang and Parkes (2023), who incorporated a self-training loop after the supervised fine-tuning phase.

3 Problem formulation

3.1 Addition as a Markov Decision Process

We propose to study the performance of a language model on a simple arithmetic task. The model is prompted to perform the addition of two numbers whose lengths range from 1 and N. To do this, it has to break down the calculation step by step, following a predefined scratchpad. In practice, we opted for the scratchpad from (Lee et al., 2024) with minor modifications (see Figure 1).

This task can be simply expressed as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where the action space $\mathcal{A}$ is the vocabulary, each state $s_t \in \mathcal{S}$ is the text generated up to t steps, with s_0 the initial prompt and $\mathcal{T}$ the (deterministic) transition function that derives directly from the ac-

tions taken by the model. The reward function $\mathcal{R}$ is 0 all along the episode, and takes the value of 1 if the final result is correct (0 otherwise). As in most reinforcement learning problems, the goal is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected return over each episode: $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{T-1} \mathcal{R}(\mathbf{s}_t)\right]$. We directly take the language model, denoted π_θ, as the policy.

3.2 Experimental setting

Our experimental pipeline consists in pre-training the language model on number lengths ranging from 1 to N, then fine-tuning it with RL on number lengths $N+1$ or $N+2$.

In the pre-training phase, we followed the approach from Lee et al. (2024), training the language model from scratch using supervised learning on a scratchpad dataset. The dataset was balanced across number lengths from 1 to N, ensuring uniform representation. The resulting pre-trained model $\pi_{\theta_{\text{old}}}$ performs well on numbers up to length N. The evaluation was conducted on two setups: fixed digit addition, where both terms had exactly N digits, and varying digit addition, where one term had N digits and the other had fewer. More details on the evaluation methods are provided in Appendix A.

For the RL fine-tuning stage, we initialized the policy with $\pi_\theta = \pi_{\theta_{\text{old}}}$ and performed training on number lengths $N+1$ or $N+2$. This corresponds to an "out-of-distribution" scenario that the model cannot reliably handle without further training. As a result, the only way for the model to succeed in this new task is to explore, so as to identify the errors it makes in the scratchpad and correct them.

3.3 Critical tokens

A notable finding from our experiments is the emergence of a small subset of tokens that significantly influence the final outcome. We refer to these as "critical tokens" and define them as follows. Within the output generated by a language model, a "critical token" is a token that satisfies both of these criteria:

- it is decisive for the rest of the answer: if the model is wrong about this token, the final answer will most likely be wrong (the model fails to correct itself);
- the pre-trained model shows substantially more uncertainty on these tokens than on the rest of the output.

	$\Delta\widehat{J}_{\theta_{\text{old}}}(s)$ critical	$\Delta\widehat{J}_{\theta_{\text{old}}}(s)$ non-critical (min.)
$N=3$	-0.33 $\pm$ 0.01	0.0012 $\pm$ 0.0001
$N=5$	-0.21 $\pm$ 0.18	0.0002 $\pm$ 0.0001
$N=7$	-0.13 $\pm$ 0.04	0.0004 $\pm$ 0.0001

Table 1: Comparison of the quantity $\Delta\widehat{J}_{\theta_{\text{old}}}(s)$ for critical and non-critical tokens, averaged over 50 generations. This shows the model's high level of uncertainty on critical tokens.

In our experiments, these tokens arise when the model has to act in a different way from that encountered during pre-training (out-of-distribution decision making). More precisely, if the model is pre-trained on numbers up to N digits, critical tokens occur in the decomposition stages that process the (N+1)-th or (N+2)-th digit (highlighted in Figure 1). Regarding the first criterion, we found that whenever these tokens are generated incorrectly, the model inevitably produces the wrong answer. As for the second criterion, we carried out a quantitative analysis comparing the model's certainty on these tokens against the others. More precisely, for each token, we measured the quantity $\Delta\widehat{J}_{\theta_{\text{old}}}(s)$, defined as the difference between the certainty on this token and the mean certainty on the others. The results, reported in Table 1, show a significant gap in certainty between the critical tokens and the rest of the output. More details on these critical tokens and their location in the scratchpad are provided in Appendix B.

4 Prioritized KL penalty

When fine-tuning a language model with RL, a Kullback-Leibler (KL) penalty term is usually added to the loss to avoid deviating too far from the pre-trained model: $\mathcal{L} = \mathcal{L}_{\text{RL}} + \alpha\mathcal{L}_{\text{KL}}$ where $\mathcal{L}_{\text{KL}} = \mathbb{E}_{s,a\sim\pi_\theta}\left[\log \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\right]$ and $\pi_{\theta_{\text{old}}}$ is the pre-trained model. As a result, the target policy π_θ is encouraged to approach the predictions of $\pi_{\theta_{\text{old}}}$ on each state-action pair. We argue that this penalty term could lead to more efficient exploration out of distribution if each state-action term was weighted by the certainty on the old policy predictions:

$$\widetilde{\mathcal{L}}_{\text{KL}} = \mathbb{E}_{s,a\sim\pi_\theta}\left[\widehat{J}_{\theta_{\text{old}}}(s)^\beta . \log \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\right] \quad (1)$$

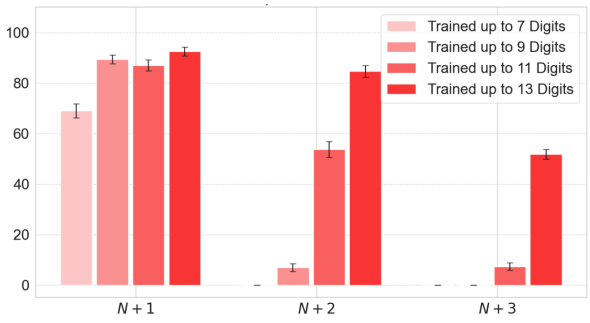


Figure 2: Model accuracy on addition tasks for models trained on numbers up to digit lengths $N = 7, 9, 11, 13$. Results are shown for varying digit evaluation. Error bars indicate 95% confidence intervals. Full detailed results are provided in Appendix D.1.

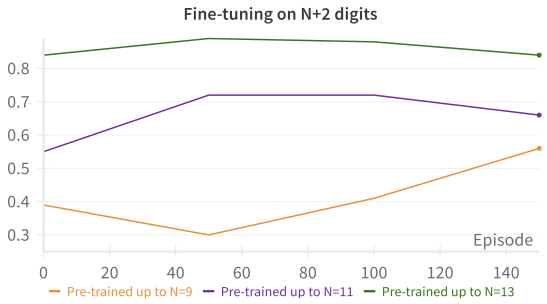


Figure 3: Learning curves of multiple models pre-trained up to N, fine-tuned with RL on $N + 2$.

where $\widehat{J}_{\theta_{\text{old}}}(s)$ estimates the certainty of the pre-trained model in state s and β is a hyperparameter. This quantity can be taken as the normalized negentropy (Brillouin, 1953), which is negatively correlated with entropy: $J = \frac{H_{\max} - H}{H_{\max}}$. In an ideal scenario, one would not only account for the data uncertainty but also for the model uncertainty, for example leveraging a bayesian approach[2]. However, since our framework falls within a context where the pre-trained model is given and fixed, we deliberately settle for an approximation that does not take into account this type of uncertainty. Our final estimate is as follows:

$$\widehat{J}_{\theta_{\text{old}}}(s) = \frac{H_{\max} - H(\pi_{\theta_{\text{old}}}(\cdot|s))}{H_{\max}} \qquad (2)$$

Our results in the next section show that, although the penalty term from Equation 1 does not address crucial aspects such as model overconfidence, it outperforms the standard KL penalty in our experimental setting.

5 Experimental results

5.1 Training Details

All experiments were carried out with the GPT-2 language model (Radford et al., 2019). A character-level tokenizer was used to ensure proper representation of digits, facilitating addition tasks (Wallace et al., 2019). The resulting model had 85M parameters. The reinforcement learning experiments were carried out with A2C (Mnih et al., 2016). We chose this algorithm because it is both simple and efficient, with few hyperparameters, making it more suitable for our comparison purposes. When applicable, the computation of the KL divergence was approximated with the estimator from Schulman (2020): $\text{KL}[q, p] \approx \frac{1}{2}(\log p(x) - \log q(x))^2$. The hyperparameters used for each experiment are provided in Appendix D.

5.2 Comparison of varying levels of pre-training

Before the application of any fine-tuning with RL, we show in Figure 2 that increasing the number N of digits during the pre-training stage improves generalization on addition tasks with larger numbers of digits. The same trend holds for equal-length addition evaluations, where models trained on larger N demonstrate better generalization. Detailed results on each task are provided in Appendix D.1.

In another experiment, we fine-tuned each pre-trained model with RL and examined their performance on additions with $N + 1$ digits. The results are reported in Figure 3. Interestingly, the models pre-trained on more digits — despite being initially more effective — tend to plateau during the exploration phase. One possible explanation is that making fewer early mistakes reduces the incentive to explore. Moreover, a qualitative analysis of the scratchpads generated by these models revealed that the errors they make (mostly copying or token-duplication issues) are less generic than those related to critical tokens. Correcting such errors may require substantially more training steps.

5.3 Impact of the prioritized KL penalty

To assess the effectiveness of the prioritized KL penalty, we conducted an experiment where a pre-trained model was fine-tuned with RL using this trick and compared it against a fine-tuning with the standard KL penalty. We chose to run this experiment on $N = 7$ digits as this is the first value

[2] In a bayesian approach, one would provide an estimate of J not only based on data uncertainty but also on model uncertainty: $J(s) = J\left[\int_{\theta_{\text{old}}} \pi_{\theta_{\text{old}}}(\cdot|s) p(\theta_{\text{old}}|\mathcal{D}_{\text{pretrain}}) d\theta_{\text{old}}\right]$.

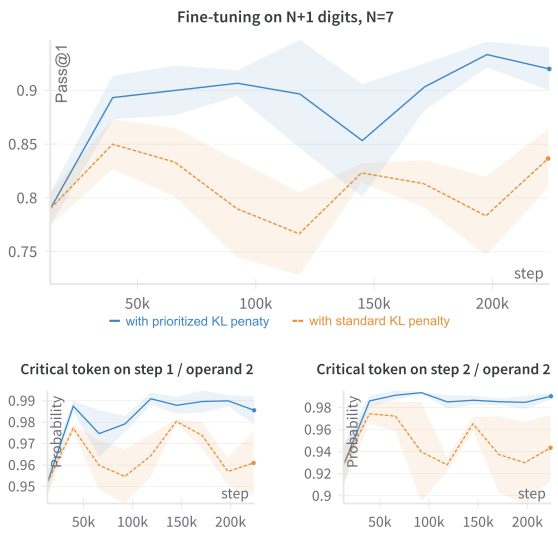


Figure 4: Top: Learning curves of a model fine-tuned with RL on N+1=8 digits. Bottom: Probability of making the right prediction on two critical tokens. Results on more critical tokens are provided in Appendix D.2.

of N for which generalization capabilities emerge after pre-training. The resulting learning curves are provided in Figure 4. From these results, one can notice that the model that benefited from the prioritized KL penalty significantly outperformed the other one. We also provide, on the same figure, some curves depicting the probability of making the right prediction on two critical tokens. Notably, the first model consistently increased and maintained a high probability of correct predictions over the long term, whereas the other one frequently reverted to its initial probability levels, likely due to the effects of the standard KL divergence. In Appendix C, we test multiple orders of magnitude for the value of the exponent β and show that the performance gain provided by the prioritized KL penalty is robust over a wide range of values.

6 Conclusion

In this paper, we studied the performance of a language model pre-trained with supervised learning and fine-tuned with RL on a simple arithmetic task. We showed that this experimental setting allowed to identify a new error mode – critical tokens – featuring decisions out of the pre-training data distribution. Therefore, we proposed a simple trick – the prioritized KL penalty – allowing to boost exploration on these tokens during the RL fine-tuning stage. In future work, we will try to extend the analysis of critical tokens to broader domains and examine the possible application of the prioritized KL penalty to more standard LLM problems.

7 Limitations

The main limitation of our study relates to the restricted experimental setup, which limits the scope of the results. Our experiments were carried out with a small language model, GPT-2, with much less capabilities than the newer, bigger models. As a result, the task is far less challenging than the benchmarks usually used to evaluate LLMs. However, this simplicity is also a strength as it allows to study the behavior of the model in a very flexible environment, with more control over the distribution shift. Moreover, the use of a formatted scratchpad for each answer allowed to easily run statistics about the model behaviour on critical tokens.

Acknowledgment

We warmly thank Matthieu Labeau for reviewing an earlier version of this paper and offering valuable feedback. We also thank Nicolas Vayatis, Pirmin Lemberger, Antoine Saillenfest and Ben Kabongo for insightful discussions about this work. This work was granted access to the HPC resources of IDRIS under the allocation 2024-TMP32592 made by GENCI.

References

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. 2022. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Filippos Bellos, Yayuan Li, Wuao Liu, and Jason Corso. 2024. Can large language models reason about goal-oriented tasks? In *Proceedings of the First edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*, pages 24–34, St. Julian's, Malta. Association for Computational Linguistics.

Leon Brillouin. 1953. The negentropy principle of information. *Journal of Applied Physics*, 24(9):1152–1163.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. 2024. Beyond sparse rewards: Enhancing reinforcement learning with language model critique in text generation. *Preprint*, arXiv:2401.07382.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer.

2023. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3676–3713. PMLR.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, and et al. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. 2024. Teaching large language models to reason with reinforcement learning. *Preprint*, arXiv:2403.04642.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Hyeonbin Hwang, Doyoung Kim, Seungone Kim, Seonghyeon Ye, and Minjoon Seo. 2024. Self-explore: Enhancing mathematical reasoning in language models with fine-grained rewards. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 1444–1466. Association for Computational Linguistics.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 21314–21328. Curran Associates, Inc.

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Teaching arithmetic to small transformers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, and et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms GPT-4 on arithmetic tasks. *CoRR*, abs/2305.14201.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024. Transformers can do arithmetic with the right embeddings. *Preprint*, arXiv:2405.17399.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

OpenAI. 2023. Gpt-4: Generative pre-trained transformer 4. `https://openai.com`. Accessed: 2024-02-06.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

John Schulman. 2020. Approximating kl divergence, 2020. *URL http://joschu. net/blog/kl-approx. html*.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. Positional description matters for transformers arithmetic. *Preprint*, arXiv:2311.14737.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. 2022. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Changnan Xiao and Bing Liu. 2023. Conditions for length generalization in learning reasoning skills. *Preprint*, arXiv:2311.16173.

Shunyu Yao, Rohan Rao, Matthew J. Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8736–8754. Association for Computational Linguistics.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *CoRR*, abs/2304.02015.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Hugh Zhang and David C. Parkes. 2023. Chain-of-thought reasoning is a policy improvement operator. *Preprint*, arXiv:2309.08589.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024. Transformers can achieve length generalization but not robustly. *Preprint*, arXiv:2402.09371.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences. *Preprint*, arXiv:1909.08593.

A Evaluation Methodology

The evaluation methodology assesses the performance of models pre-trained on digit addition tasks, following the framework of Lee et al. (2024). Each model, denoted as $\pi_{\theta_{\text{old}}}$, is pre-trained using supervised learning on addition tasks involving up to N digits. The evaluation is conducted under two scenarios:

1. **Identical Digit Addition:** Both terms in the addition consist of exactly N digits (i.e., $N + N$ digit addition).

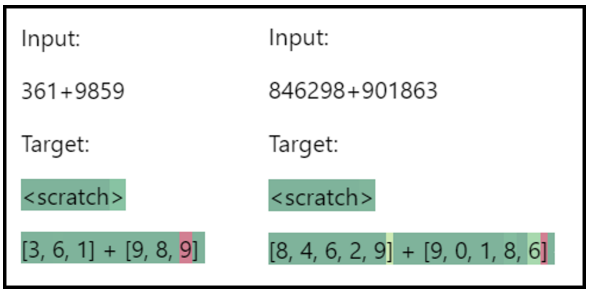
2. **Varying Digit Addition:** The model is tested on addition tasks where the number of digits in the two terms varies (i.e., $N + M$ digit addition, where $M \leq N$). The pairs of numbers with different digit counts are sampled to ensure a broader range of difficulty.

Model outputs are evaluated by comparing the predicted results to the ground truth for each addition. Accuracy is computed as the proportion of correct predictions over the total number of examples.

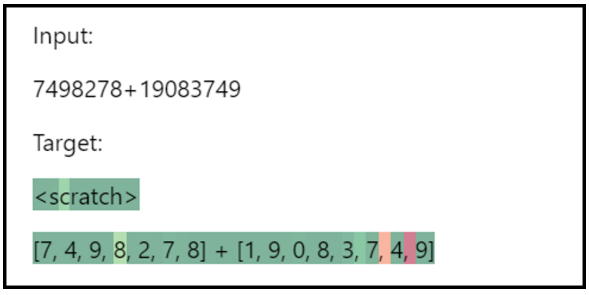
The evaluation is performed on 1,000 test examples. To account for variability in performance, results include confidence intervals obtained via resampling.

B Critical tokens

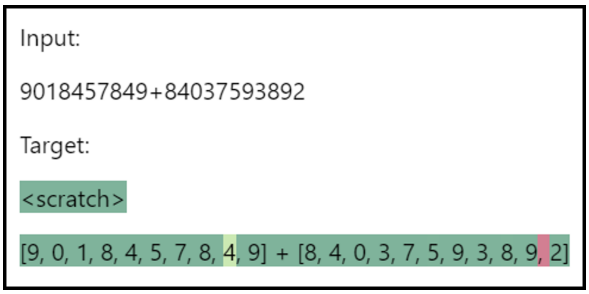
In this section, we provide insight into critical tokens, that play a crucial role in determining the success of the addition task. Consider a pre-trained model on additions of numbers up to N digits. Now, consider a generalization test in which the model is prompted to add numbers with $N + 1$ digits. Our experiments reveal that when the model fails at this task, the failure can typically be traced back to errors made on critical tokens. We observe that these critical tokens arise at the stage of the generation where the model must choose whether to treat the problem as an addition of N-digit numbers – leading to failure – or correctly addressing the task of adding $(N + 1)$-digit numbers. More precisely, this error is caused by the omission of digits when copying the numbers from the previous step. Figure 5a shows two examples of failed generation caused by errors on the critical tokens. In the first case, the model pre-trained on numbers up to 3 digits mistakenly recopies the last digit instead of the penultimate digit, leading to an incorrect outcome. In the second example, where the model is pre-trained on numbers up to 5 digits, it incorrectly closes the bracket in both cases instead of inserting a comma (the stage preceding the copying of the sixth digit). These examples illustrate two types of critical tokens. We only show them on the first decomposition line, but they can be found on the subsequent lines as well.



(a)



(b)



(c)

Figure 5: Output examples for addition tasks on $N + 1$ digit lengths (the model is faced with numbers one notch longer than those encountered in pre-training). Each generated token is colored according to its certainty. A green color is a maximal certainty, while a red color is a minimal certainty.

As explained in Section 4, we quantify the certainty of model being in state s through the quantity $\widehat{J}_{\theta_{\text{old}}}(s)$. To provide more visual understanding of this quantity, we display in Figure 5 a few output examples with the colors as indication of the model certainty (green: high certainty, red: low certainty).

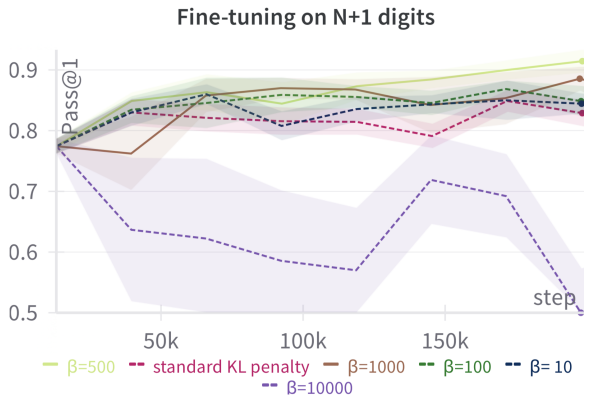


Figure 6: Fine-tuning results with various values of β (averaged over 9 random seeds)

C Assessing the impact of the certainty exponent β

In order to better assess the robustness of our prioritized KL penalty, we have carried out an experiment testing multiple orders of magnitude for the value of the β exponent in Equation 1. The corresponding learning curves are reported in Figure 6. Despite important error margins, these results show that the prioritized KL penalty slightly outperforms the standard KL penalty for values of β ranging from 10 to 500, reaching its maximum at $\beta = 500$ and starting to decline from $\beta = 1000$ (which shows early signs of instability). The performance drops drastically at $\beta = 10000$. The good performance over such a wide range of beta values can be explained by the fact that after our pre-training, the confidence of the model is extremely high (except on critical tokens), which is why it takes large values of β to drastically reduce the weight $\widehat{J}_{\theta_{\text{old}}}(s)^{\beta}$ in the prioritized KL penalty. Therefore, we believe that this range (10-500) of acceptable β values might be quite different in another problem.

D Experiments Details

The hyperparameters used in the experiment from Section 5.2 are provided in Table 2. The hyperparameters used in the experiment from Section 5.3 are provided in Table 3.

D.1 Detailed Pretraining Results

Tables 4 and 5 display the model's performance on addition tasks for different digit lengths that the model was pretrained on. These digit lengths refer to the number of digits used during pretraining (7, 9, 11, and 13 digits), with accuracy then measured on

Hyperparameter	Value
Learning rate	10^{-6}
Discount factor	1
Value function coefficient	0.1
Entropy coefficient	0.0005
KL penalty coefficient	10
Repeat per collect	1
Episodes per collect	50
Episodes per test	100

Table 2: Hyperparameters used in the RL experiment comparing multiple levels of pre-training

Hyperparameter	Value
Learning rate	10^{-6}
Discount factor	1
Value function coefficient	0.1
Entropy coefficient	0.0005
KL penalty coefficient	5
KL penalty exponent (β)	150
Repeat per collect	1
Episodes per collect	50
Episodes per test	100

Table 3: Hyperparameters used in the RL experiment evaluating the impact of the prioritized KL penalty

tasks involving identical digit lengths and varying digit lengths. The model is subsequently evaluated on its ability to generalize to more complex tasks, i.e., $N + 1$, $N + 2$, and $N + 3$ digits, where the total number of digits exceeds the training range.

Across both tables, the general trend indicates that the model is more adept at solving tasks within its training range, and it exhibits improved generalization with larger digit lengths training. However, in both identical and varying digit tasks, the model's ability to handle tasks involving $N + 2$ and $N + 3$ is limited, particularly for smaller digit lengths. This suggests that while pretraining enables the model to generalize to some extent, there are clear limitations when the task complexity surpasses the data on which the model was trained.

D.2 Details on the fine-tuning with RL experiments

In Figure 7 we expose the evolution of the right prediction probability for 6 different critical tokens. These critical tokens are selected as the commas on the $(N + 1)$-th token for each operand list, which

Nb. of Digits	N Accuracy	$N+1$ Accuracy	$N+2$ Accuracy	$N+3$ Accuracy
7	$98.9\% \pm 0.7\%$	$48.8\% \pm 3.0\%$	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$
9	$96.4\% \pm 0.6\%$	$78.9\% \pm 2.4\%$	$0.5\% \pm 0.5\%$	$0.0\% \pm 0.0\%$
11	$91.2\% \pm 1.3\%$	$75.1\% \pm 2.7\%$	$30.7\% \pm 2.4\%$	$0.2\% \pm 0.3\%$
13	$93.0\% \pm 1.6\%$	$88.9\% \pm 2.1\%$	$67.7\% \pm 3.1\%$	$20.4\% \pm 2.4\%$

Table 4: Model accuracy on addition tasks with identical digit lengths.

Nb. of Digits	N Accuracy	$N+1$ Accuracy	$N+2$ Accuracy	$N+3$ Accuracy
7	$100.0\% \pm 0.0\%$	$69.0\% \pm 2.4\%$	$0.0\% \pm 0.0\%$	$0.0\% \pm 0.0\%$
9	$97.0\% \pm 0.6\%$	$89.4\% \pm 1.8\%$	$6.9\% \pm 1.3\%$	$0.0\% \pm 0.0\%$
11	$94.4\% \pm 1.4\%$	$87.0\% \pm 2.1\%$	$53.7\% \pm 3.2\%$	$7.3\% \pm 1.6\%$
13	$95.6\% \pm 1.4\%$	$92.5\% \pm 1.9\%$	$84.7\% \pm 2.4\%$	$51.8\% \pm 3.2\%$

Table 5: Model accuracy on addition tasks with varying digit lengths.

is a frequent source of errors. One can observe that in each situation (despite important error margins), the probabilities outputted by the model trained with prioritized KL penalty are higher than the other. Note that this effect is more pronounced on tokens "step 1 / operand 2" and "step 2 / operand 2" as on the others, the probability of success is already very high from the start.

E Softwares

We carried out our experiments using the Python packages Transformers (Wolf et al., 2020) and Tianshou (Weng et al., 2022).

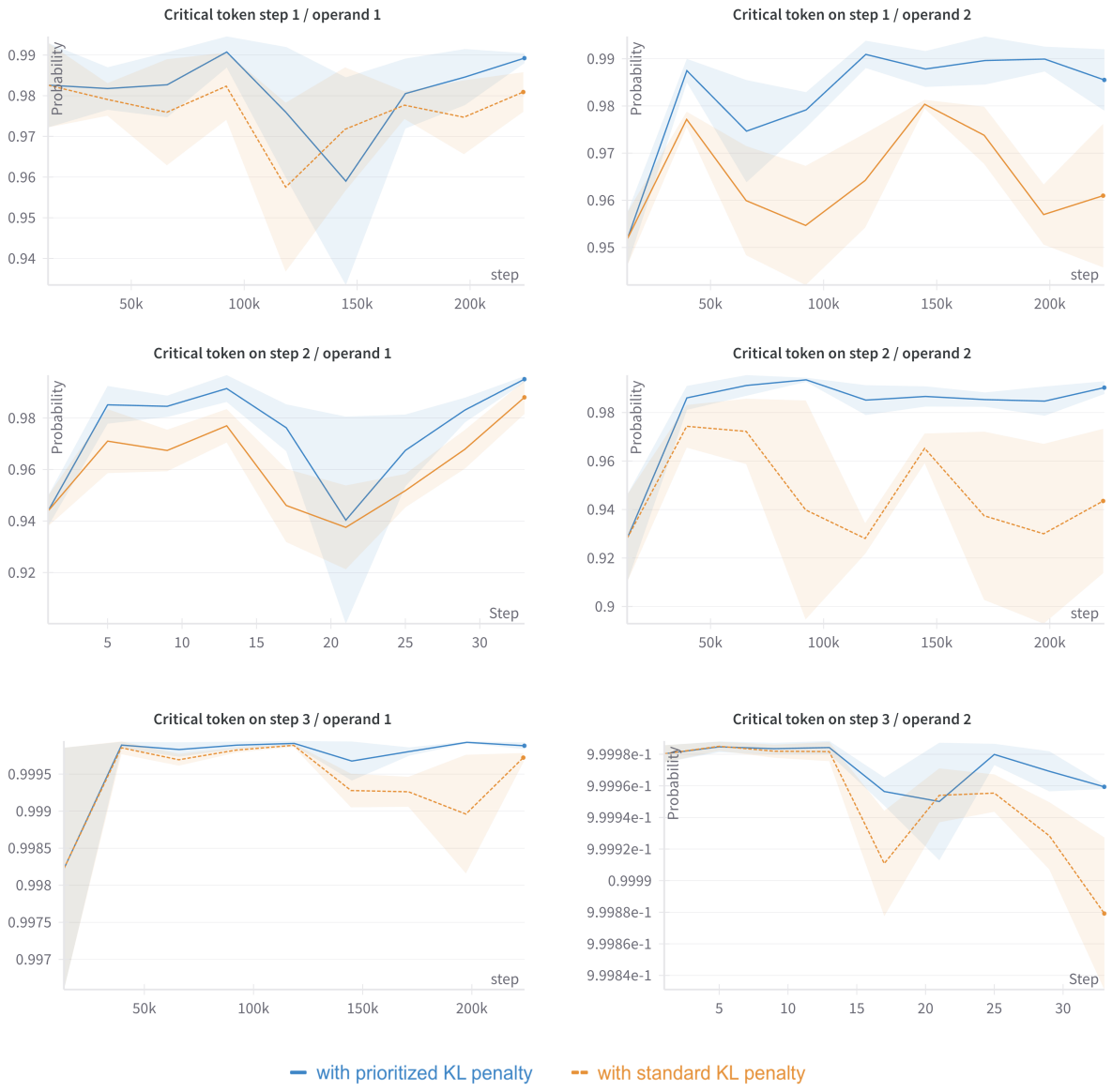


Figure 7: Evolution of the right prediction probability on multiple critical tokens, during the RL fine-tuning on number length $N + 1 = 8$.