Inference Time Input Space Scaling with Untrained Filler Tokens

Anonymous ACL submission

Abstract

Additional text in the input context enables performance improvement in language generation, benefiting not only from its verbalized reasoning steps but also from the *extended computational space of longer inputs*. Prior work has trained filler, random or special tokens to serve as additional computation spaces for performance increase. In this study, we investigate whether language models can leverage artificially inserted filler tokens as expanded computation spaces at inference time without explicit training. We experiment with models ranging from 1.7B to 32B on open-domain QA and math tasks, showing that frequently occurring, semantically vacuous tokens such as whitespace and newlines help improve performance — with smaller models benefiting more, up to 12.372%p in SmolLM2-1.7B-Instruct. Logit lens and attention map analyses reveal that these tokens enable the model to use expanded spaces as a scratchpad for intermediate computation, whereas infrequent or random tokens introduce noise that disrupts this mechanism.

1 Introduction

While the effectiveness of additional context mostly stems from the solution steps provided (Hua and Zhang, 2022; Wei et al., 2022; Wang et al., 2023; Zelikman et al., 2022), prior studies have shown that training models with sequences of meaningless tokens — such as repeated filler characters or special tokens (e.g., <pause>) — can also improve performance (Herel and Mikolov, 2024; Goyal et al., 2024; Lanham et al., 2023). Extending this line of inquiry, we study whether language models can *exploit such expanded spaces even without training*, and find that providing tokens familiar from the pretraining corpus as an *expanded computation space* can enhance model performance at inference time.

Our study is broken down into three research questions: **RQ1**. What types and amounts of tokens are effective, and which parts of the input location benefit more from their insertion? **RQ2**. When during training do models start to effectively exploit the expanded computation spaces to support answer inference? **RQ3**. How do the extended token spaces interact with the original inputs and affect the answer prediction?

To answer the questions, we adopt six characters as filler tokens to extend the input space, and vary the amounts of filler tokens to be placed before and after the final ‘`Answer:`’ token of the input. We perform experiments with transformer-based causal decoder-only language models ranging from 1.7B to 32B parameters, and examine intermediate checkpoints to study when and how this expanded space becomes effective for problem-solving. To further investigate what actually occurs within this expanded space with the original inputs, we analyze attention maps.

We find that models can exploit filler tokens as expanded computation spaces even without explicit training. While the optimal token types and counts vary across models, some tokens consistently improve performance, with smaller models benefiting more, likely due to their limited computational capacity. Notably, through logit lens (Fiotto-Kaufman et al., 2024) analysis, the effectiveness of filler tokens depends critically on their frequency in the pretraining corpus: tokens that appear frequently, whether as non-content tokens (e.g., ‘ ’) or common words (e.g., ‘`the`’), enable the model to use these spaces as a scratchpad for intermediate computation, whereas infrequent or random tokens introduce noise that disperses attention and disrupts this mechanism. An analysis of intermediate PT and IT checkpoints indicates that models progressively learn to leverage these spaces during training. Attention map analysis further reveals that the expanded spaces serve as meaningful extensions of the original attention mechanism, sometimes attending directly to questions or answer options.

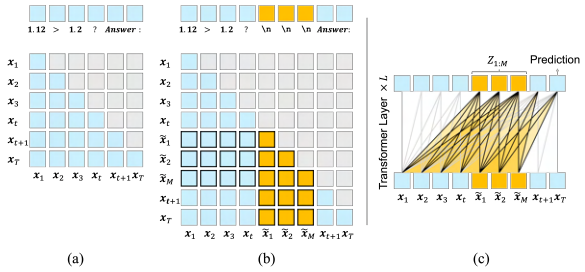

Figure 1: (a) The original prompt and its attention map. (b) The same prompt with filler tokens inserted (*yellow*), and its attention map. (c) An equivalent edge-wise depiction of the attention mapping. The yellow boxes in (b) and (c) mark the extended inputs, and black solid lines denote the additional attention operations induced by the filler tokens.

2 Input Space Scaling with Filler Tokens

Based on general next-token prediction (Appendix A.2), given an input sequence $\boldsymbol{x}_{1:T}$, we insert filler tokens $\tilde{\boldsymbol{x}}$ of length M (tokens in *yellow*), thereby extending the input to a sequence of length $T + M$ (Figure 1 (c)). This produces an additional set of M intermediate vectors, denoted as $\boldsymbol{Z}^l_{1:M} = [\boldsymbol{z}^l_1, \ldots, \boldsymbol{z}^l_M]$ for each l-th transformer layer in the model. The model then predicts the next token by jointly considering the original input and the inserted filler tokens. We hypothesize that these additional positions provide extra representational capacity beyond the original input length, functioning as *expanded computation spaces* (ECS) that support improved reasoning capabilities:

$$\begin{aligned} ECS_M &= f(x_{1:T+M}) \setminus f(x_{1:T}) \\ &= \boldsymbol{Z}_{1:M} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M]. \end{aligned} \tag{1}$$

Here, $f(\cdot)$ denotes the hidden representation function of the transformer-based decoder-only model given the input; the operator $\setminus$ denotes the set difference. For auto-regressive language modeling, non-lower-triangular scores in the attention map are masked out (grey regions in Figure 1 (a) and (b)) as it is not a bidirectional operation. With the original input of length T, attention computation involves T^2 scores, of which $T(T-1)/2$ are masked. When the input is augmented with M filler tokens, the attention complexity increases to $(T+M)^2$, with $(T+M)(T+M-1)/2$ scores masked. Consequently, the regions in black lines correspond to ECS.

We examine whether ECS merely constitute redundant additions in the attention computation of transformer-based models, or whether they can actively contribute to model inference, and if so, in what manner. We hypothesize that such synthetic modifications of the input provide additional computational capacity for reasoning, thereby enhancing performance. However, since this intervention deviates from the way models are typically trained, we assume that its effectiveness may vary across models and settings, and in some cases may deteriorate the performance without additional training.

3 Experiments

3.1 Experimental Settings

We utilize various transformer-based models including SmolLM2 (Allal et al., 2025), Gemma-3 (Team et al., 2025b), Llama-3.1 (Grattafiori et al., 2024), Qwen2.5 (Qwen et al., 2025), OLMo-2 (OLMo et al., 2025) and experiment them on general domain QA datasets (MMLU (Hendrycks et al., 2021), ARC (Clark et al., 2018)) and math datasets (GSM8K (Cobbe et al., 2021) and HuggingFaceH4/MATH-500). Details regarding specific model types, datasets and evaluation methods are provided in Appendix A.2.1.

3.2 Effect on Question Answering Tasks

We first investigate the effect of adding filler tokens by evaluating the multiple-choice QA performance of models on MMLU, a general-domain QA benchmark, and ARC, a science-domain QA task. Through these experiments, we identify which types of filler tokens, and in what quantities, can positively influence model performance. See Appendix A.8 and A.9 for the full results; Appendix A.4 for results on *math* datasets.

3.2.1 Token Types

We consider six types of tokens that appear frequently in the pretraining corpus yet carry no meaningful semantic content: ‘ ’ (*space*), ‘\n’ (*enter*), ‘\t’ (*tab*), ‘.’ (*period*), ‘<pad>’ (*pad*), and ‘-’ (*dash*). We deliberately exclude semantically meaningful tokens, as inserting them as fillers without additional training may interfere with the model’s inference by introducing unintended semantic signals. Since our approach operates solely at inference time, we restrict our selection to tokens that are unlikely to carry task-relevant meaning. The proportion of each token in the pretraining corpus is reported in Appendix A.3. Experiments with random and infrequent filler tokens are provided in Appendix A.6 and A.5.

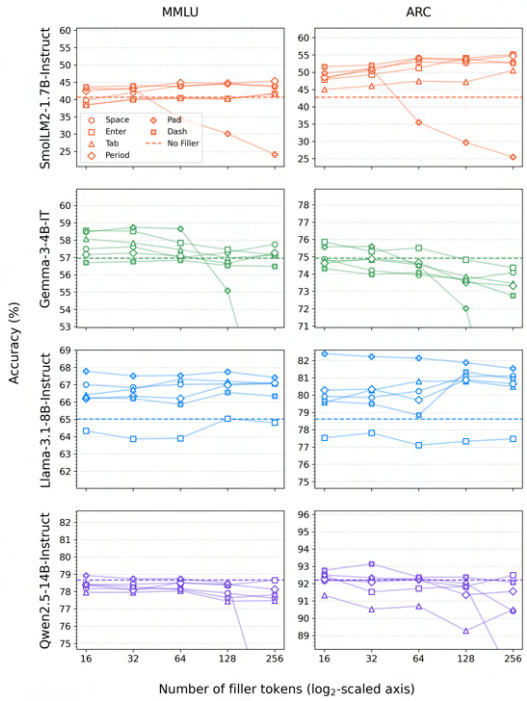

Figure 2: MMLU and ARC results of four different models. Smaller models tend to benefit from the additional space provided by almost any tokens, using them to improve performance. In contrast, models with sufficient capacity for solving the task gain relatively less from such extension.

Across models, we observe accuracy improvements of up to 12.372%p compared to the baseline without filler tokens (Figure 2). For SmolLM, *period* yields the best performance on MMLU while '\n' is most effective on ARC. The '<pad>' token severely degrades performance beyond 64 tokens for most models, except Llama-3.1-8B-IT, which achieves its highest performance with '<pad>' — attributable to Llama's use of '<eos>' in place of a dedicated pad token. Qwen2.5-14B-IT shows marginal gains, whereas Qwen2.5-32B-IT exhibits steady improvements without early degradation, though with a smaller magnitude than smaller models.

3.2.2 Token Amounts

Overall, performance tends to deteriorate as more tokens are added, with a sharp decline occurring for the '<pad>' token once the number exceeds 64. We attribute this behavior to the characteristics of the training corpus and preprocessing, where excessive repetitions of filler-like tokens are relatively rare. Since the models have not been trained on such inputs, an overly large number of added tokens cannot be effectively utilized and, instead, appear to hinder performance, resembling the lost-in-the-middle phenomenon (Liu et al., 2024; Wright et al., 2025). It was observed that once the number of added tokens surpassed 1024, the accuracy consistently declined to the 20% range across almost all cases, as shown in Table 2 in Appendix A.7.

While all models show performance degradation beyond a certain input length, the SmolLM2-1.7B-IT model exhibits the opposite trend, with accuracy continuing to improve as the number of tokens increases up to 256. This suggests that, in SmolLM, the additional tokens are effectively utilized for reasoning, compensating for its limited parameter capacity by exploiting the synthetically expanded computation space. In contrast, larger models—already equipped with sufficient parameters to achieve strong performance—benefit relatively less from such extensions.

3.2.3 Insertion Position

We focus on the position of filler tokens: inserting tokens immediately before/after the '`Answer:`' token. As shown in Figure 5 (Appendix A.4), placing filler tokens before the `Answer:` token (colored) leads to better performance compared to placing filler tokens after it (empty). Even with a large number of filler tokens, the model can still generate an appropriate answer when '`Answer:`' appears at the very end, since the final input naturally prompts next-token prediction to produce an answer. However, if many filler tokens follow '`Answer:`', the model is more likely to predict another filler token as the next token, which provides little benefit for answer generation. Therefore, when a large amount of artificial tokens is added to the input, positioning the answer-prompting token at the very end of the sequence appears crucial for effectively utilizing the additional computational space.

3.3 Development of Expanded Input Space

We examine OLMo-2-1124-13B checkpoints across the full PT and IT stages on ARC to identify when the capability to exploit filler tokens emerges, leveraging the fact that all checkpoints are available. PT Stage 1 spans 590K steps (5T tokens), and we sample every 200B tokens; Stage 2 covers 35K steps (300B tokens) using ingredient 4. As shown in Figure 3, adding 64 '\n' tokens initially degrades performance, with gains only emerging from mid-to-late Stage 2, suggesting that sufficient data and language proficiency are prerequisites for leveraging fillers as additional computational space.

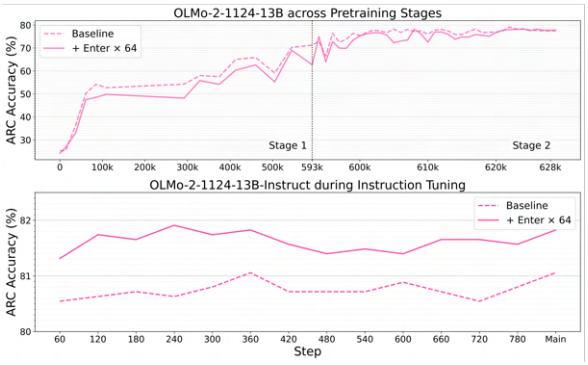

Figure 3: (Top) ARC results across PT checkpoints of OLMo-2-1124-13B. PT Stage 1 is compressed; Stage 2 expanded for readability. (Bottom) ARC results across IT checkpoints of OLMo-2-1124-13B-Instruct.

For OLMo-2 IT, the phase spans 14 checkpoints sampled at 60-step intervals. Performance varies only slightly above 80.5, peaking at the final checkpoint. With filler tokens, the model follows a similar trend but maintains approximately 1%p higher scores throughout. The consistent gains across all IT checkpoints suggest that once the model has acquired sufficient language ability, it can reliably leverage filler tokens as additional computational space.

3.4 Interpretation of Inner Workings

Logit Lens To examine how filler tokens influence layer-wise decision making, we conduct a logit lens analysis comparing settings with and without filler tokens, using ‘ ’, ‘\n’, ‘of’, the infrequent token ‘갉’, and random tokens of the same length (Appendix A.5). ‘ ’ and ‘\n’ enable the model to confidently predict the correct answer from around the 2/3 layer mark, whereas without fillers, the model only arrives at the correct answer at the final layers with low confidence. Interestingly, ‘of’ exhibits a similar effect despite being semantically meaningful, suggesting that PT corpus frequency is the key factor. In contrast, ‘갉’ provides no benefit, and random tokens actively disrupt prediction. These results indicate that filler tokens act not as mere length extensions, but as a scratchpad for intermediate computation when the tokens are familiar from pretraining.

Attention Map To assess the relationship between the original input and the extended spaces, we inspect attention maps (Appendix A.6), examining a success case with filler tokens and a failure case with random tokens.

As shown in Figure 8, in the early layers (a), attention scores within filler tokens are higher on average than in other tokens. In the middle layers (b), filler tokens attend uniformly to the original input regardless of position, suggesting even contribution across the filler space. Notably, certain filler tokens attend strongly to specific question words (e.g., *kinase*) and answer options (c), indicating active participation in interpreting the input. By layer 23 of 34 (d), the filler space shows strong attention to the correct answer option D, suggesting that the decision emerges in the mid-to-late layers. In the final layer (e), filler tokens shift attention to <EOT> and answer tokens, indicating preparation for output generation. In contrast, Figure 9 shows that when random tokens are used as fillers, attention is dispersed and sporadic, lacking the focused attention on key parts of the input that is observed with ‘ ’. These examples demonstrate that effective filler tokens do attend to important information in the question and options and influence the process of answer selection, whereas random tokens instead interfere with decision making. This confirms that it is not the extended input length itself that benefits the model, but rather the use of tokens familiar from pretraining that enables meaningful exploitation of the expanded computation space. Additional examples from other models and samples are provided in Appendix A.10.

4 Conclusion

In this work, we introduced an intriguing phenomenon where inserting filler tokens at inference time improves language model performance without explicit training. Across models of varying sizes, frequently occurring, semantically vacuous tokens consistently yield gains on QA and math tasks, with smaller models benefiting more, suggesting that expanded spaces compensate for limited parameter capacity. Through logit lens analysis, we find that token familiarity from the pretraining corpus is key: ‘ ’ and ‘\n’ tokens enable the model to use these spaces as a scratchpad for intermediate computation, whereas infrequent or random tokens introduce noise that disrupts this mechanism. Attention map analysis further confirms that these spaces are not redundant extensions but sites of meaningful computation for answer inference, and experiments with intermediate checkpoints show that this ability is progressively acquired during pretraining.

Limitations

Building on prior studies suggesting that the additional token space itself contributes to problem-solving performance, our work investigates whether such tokens can play a similar role even without training. However, since the tokens were introduced without any training process, we did not use completely random token combinations, otherwise they would increase the perplexity. Instead, we restricted our experiments to six types of characters that can be commonly observed in a natural training corpus. We further limited the number of additional tokens to a range from 16 to 8194.

In addition, we did not artificially insert these tokens in the middle of the question and instruction tokens for the stable perplexity. Rather, we only added filler tokens only before and after the answer tokens. We also observed that this capability emerges during the middle stages of pre-training (PT). However, as the training data recipes for all models are not publicly available, we can only hypothesize that there is a correlation between the statistics of the pre-training corpus and the effectiveness of the filler tokens, without being able to draw definitive conclusions.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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.

Nathaniel D Daw, Yael Niv, and Peter Dayan. 2005. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Jonathan St BT Evans. 1984. Heuristic and analytic processes in reasoning. *British Journal of Psychology*, 75(4):451–468.

Siqi Fan, Bowen Qin, Peng Han, Shuo Shang, Yequan Wang, and Aixin Sun. 2025. The price of a second thought: On the evaluation of reasoning efficiency in large language models. *Preprint*, arXiv:2505.22017.

Jaden Fiotto-Kaufman, Alexander R Loftus, Eric Todd, Jannik Brinkmann, Caden Juang, Koyena Pal, Can Rager, Aaron Mueller, Samuel Marks, Arnab Sen Sharma, Francesca Lucchetti, Michael Ripa, Adam Belfki, Nikhil Prakash, Sumeet Multani, Carla Brodley, Arjun Guha, Jonathan Bell, Byron Wallace, and David Bau. 2024. Nnsight and ndif: Democratizing access to foundation model internals.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

David Herel and Tomas Mikolov. 2024. Thinking tokens for language modeling. *Preprint*, arXiv:2405.08644.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wenyue Hua and Yongfeng Zhang. 2022. System 1 + system 2 = better world: Neural-symbolic chain of logic reasoning. In *Findings of the Association for*

Computational Linguistics: EMNLP 2022, pages 601–612, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Daniel Kahneman. 2003. Maps of bounded rationality: Psychology for behavioral economics. *American economic review*, 93(5):1449–1475.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhiwei Li, Bao-Long Bi, Ling-Rui Mei, Junfeng Fang, Xiao Liang, Zhijiang Guo, and 2 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *Preprint*, arXiv:2502.17419.

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025a. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025b. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed 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*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too). *Preprint*, arXiv:2311.11829.

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, and 3 others. 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.

Dustin Wright, Zain Muhammad Mujahid, Lu Wang, Isabelle Augenstein, and David Jurgens. 2025. Unstructured Evidence Attribution for Long Context Query Focused Summarization. *Preprint*, arXiv:2502.14409.

Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. 2025. Reasoning models better express their confidence. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. 2024.

Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

A Appendix

A.1 Related work

Slow thinking for improved reasoning Dual-system theory (Daw et al., 2005), which posits that human cognition operates through two distinct modes, has also been invoked in prior machine learning research. System 1 processes problems quickly in an immediate and automatic manner, while System 2 is slower but more analytical, goes step by step, leading to more accurate and higher-level decision-making (Evans, 1984; Kahneman, 2003). Transformer-based language models have been known to be proficient at fast decision-making but have shown weaknesses in handling complex reasoning. However, recent models exhibit sophisticated reasoning capabilities, in some cases comparable to those of human experts (OpenAI et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025a). Recent developments show that large language models (LLMs) are evolving from fast, intuitive System 1 processing toward slower but more deliberate System 2 reasoning (slow thinking) (Li et al., 2025; Weston and Sukhbaatar, 2023). LLMs can infer solutions to unseen problems by leveraging patterns from a few given samples or shots. In particular, they are developing sophisticated reasoning skills that mimic human abilities, such as decomposing problems step by step (e.g., chain-of-thought, CoT) and solving them in a structured, sequential manner (Wei et al., 2022; Hua and Zhang, 2022; Wang et al., 2023; Zelikman et al., 2024).

Tokens for thinking Previous studies have explored methodologies that incorporate dummy tokens or special tokens during the pre-training (PT) or fine-tuning (FT) stages, aiming to improve the reasoning of the models. First, in the DeepSeek-R1 (Guo et al., 2025) model, the tokens <think> and </think> are utilized to guide the model to perform a thinking process for answer inference within the enclosed span. Similarly, Zelikman et al. (2024) proposed a method in which, at each token generation step, the tokens <|startofthought|> and <|endofthought|> were used to prompt the model to generate multiple rationales explaining the future text, thereby improving prediction. Herel and Mikolov (2024) demonstrated that, within a recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997) architecture, inserting thinking tokens <T> between input tokens reduces the perplexity of correct answers in complex mathematical computations. Additionally, ongoing research has explored the use of special thinking tokens to prompt models to engage in improved reasoning (Fan et al., 2025; Yoon et al., 2025).

The works most closely aligned with our study are Pfau et al. (2024) and Goyal et al. (2024). Pfau et al. (2024) shows that transformers can use meaningless filler tokens in place of chain-of-thought for problem-solving when they are properly trained. They introduce a method for generating the synthetic data with filler tokens and training to converge the models. In Goyal et al. (2024), multiple <PAUSE> tokens are inserted into the model input to delay the answer, rather than producing it immediately. To this end, the input is deliberately modified during the PT and FT stages to include <PAUSE> tokens, making the model learn them. They report that delaying the responses using these special tokens leads to performance improvements across several benchmarks. In Lanham et al. (2023), filler tokens (e.g., '.....') are replaced with chain-of-thought (CoT) sentences to study what contributes to the performance, but the performance decreases without CoT sentences, suggesting that training should be executed to use filler tokens for reasoning. Whereas prior works rely on training such tokens to elicit reasoning or thinking abilities from models, our approach explores the provision of additional extended space in the model input without training the tokens, instead utilizing tokens that the model has seen during training, to evaluate their effect.

A.2 Next-token prediction

Causal decoder-only language models, consisting of a vocabulary $\mathcal{V}$ and L transformer layers, are given an input $\boldsymbol{x}_{1:T} \in \mathcal{V}^T$ with T length of tokens. For each layer $l \in [1, L]$, an intermediate vector $\boldsymbol{z}_t^{(l)}$, for each input token $\boldsymbol{x}_t$, is obtained. Based on the last token vector of the last layer $\boldsymbol{z}_T^{(L)}$, the models predict the most likely next token $\boldsymbol{x}_{T+1}$. In each transformer block, it takes a matrix of T vectors of D hidden state dimension size, $\boldsymbol{Z}_{1:T}^l = [\boldsymbol{z}_1, ..., \boldsymbol{z}_T] \in \mathbb{R}^{D \times T}$, as its input, and transform it into the output matrix $\boldsymbol{Z}_{1:T}^{l+1} \in \mathbb{R}^{D \times T}$, with its internal attention, feedforward and layer-norm modules. The attention module takes query vector $\boldsymbol{Q}_t \in \mathbb{R}^D$ for each input position t, and key and value matrices $\boldsymbol{K}_{1:t}, \boldsymbol{V}_{1:t} \in \mathbb{R}^{t \times D}$ of all previous positions. Then the attention output passes

feed forward network and layer-norm modules as follows:

$$\begin{aligned} \boldsymbol{a}_t^l &= LN(Attn(\boldsymbol{Q}_t, \boldsymbol{K}_{1:t}, \boldsymbol{V}_{1:t}) + \boldsymbol{z}_t^l) \\ \boldsymbol{z}_t^{l+1} &= LN(FFN(\boldsymbol{a}_t^l) + \boldsymbol{a}_t^l). \end{aligned} \tag{2}$$

Note that, depending on the model, layer normalization can be applied either before or after the attention operation.

A.2.1 Experimental Settings

Models We experiment with models with the number of parameters from 1.7B to 32B that are publicly available on HuggingFace's transformers libraries (Wolf et al., 2020). We adopt HuggingFace's `SmolLM2-1.7B-Instruct` (Allal et al., 2025) model, which is a 1.7B-sized model. For the 4B size model, we use Google Gemma team's `Gemma-3-4B-it` (Team et al., 2025b), which has 1024 context length. For the 8B model, we adopt Meta's `Llama-3.1-8B-Instruct` (Grattafiori et al., 2024) with 131072 context length. For 14B model, we adopt Qwen team's `Qwen2.5-14B-IT` (Qwen et al., 2025) (context length of 131072), AllenAI's OLMo-2 (OLMo et al., 2025) (context length of 4096). OLMo-2 has released its intermediate checkpoints for pretraining (`OLMo-2-1124-13B`) and instruction-tuning (`OLMo-2-1124-13B-Instruct`), so we exploit these checkpoints for tracking with the training steps. For the biggest size model in our experiments, we use the `Qwen2.5-32B-Instruct` model.

Datasets We experiment with two open-domain question answering (QA) datasets: Measuring Massive Multitask Language Understanding (MMLU, Hendrycks et al. (2021)) and AI2 Reasoning Challenge (ARC, Clark et al. (2018)) for measuring the model's ability in general domain question answering. MMLU is composed of 57 tasks, a total of 14079 samples that can be categorized into (1) Humanities, (2) Social Science, (3) Science, Technology, Engineering, and Mathematics (STEM), and (4) Other. We adopt the ARC *challenge* test subset, which has 1172 samples in science questions with mostly 4 options, but also with 3 and 5 options. We additionally evaluate on the mathematics datasets GSM8K and MATH-500 [1] to measure the deeper reasoning ability of models. GSM8K comprises 1819 test set examples of school mathematics. MATH-500 consists of 500 problems from the OpenAI MATH benchmark (Lightman et al., 2024).

Evaluation We evaluate model performance using the last token logits of the final layer. For the MMLU and ARC datasets, which consist of multiple-choice questions with mostly four options, we provide the model with the input sequence: `chat template` (for IT models), `task instruction`, `context` (if exists), `question`, `answer options`, and the prompt '`Answer:`'. From the final logits, we extract only those corresponding to the option tokens (`A`–`D`), apply a softmax over them, and select the option with the highest probability as the predicted answer. For the math tasks, the model is required to generate an answer in free-form text, and the final prediction is extracted from the output using a predefined rule to match the exact value, following the existing implementation.[2]

For our experiments, we use a zero-shot setting, providing the models only with the dataset samples without any additional task-related examples, and we report the average result of three runs with different seeds (42, 64, and 128). The inference time for each experiment varies from 3 minutes to over 3 hours on one H100 GPU depending on the model size, input length, and batch size.

A.3 Token Proportion in PT dataset

To examine the proportion of each token in the pretraining (PT) corpus, we count token frequencies over the `allenai/olmo-mix-1124` dataset, presented in Table 1. Since the full PT dataset is prohibitively large (∼621M rows), we performed stratified random sampling of 4.3M rows (∼0.69% of the total) with subset proportions preserved. Interestingly, '\n' is the most frequent token in the OLMo pretraining corpus and is also found to be the most effective filler token in the OLMo-2 experiments, consistent across both PT and IT settings.

We further examine the top 10 most frequent tokens in the corpus. The most frequent token is '\n' (4.041%), followed by '.' (3.528%) and ',' (3.129%). Common function words with a preceding space also rank highly, including 'Ġthe' (2.892%), 'Ġof' (1.573%), ' '(Ġ) (1.536%), 'Ġto' (1.522%), 'Ġand' (1.437%), 'Ġa' (1.262%),

[1] https://huggingface.co/datasets/HuggingFaceH4/MATH-500

[2] https://github.com/EleutherAI/lm-evaluation-harness/blob/main/lm_eval/tasks/hendrycks_math/utils.py

Token	Count	(%)
‘ ’ (*space*)	88,891,374	1.536%
‘\n’ (*newline*)	233,800,111	4.041%
‘\t’ (*tab*)	8,415,928	0.145%
‘.’ (*period*)	206,543,435	3.570%
‘<pad>’ (*pad*)	5	0.000%
‘-’ (*dash*	51,300,683	0.887%

Table 1: Proportion of six tokens in the `allenai/olmo-mix-1124` corpus (PT dataset), estimated via stratified random sampling of 4.3M rows (∼0.69% of ∼621M total rows) with subset proportions preserved. `\n` (*newline*) is the most frequent token, accounting for 4.041% of all tokens.

and ’Ġin’ (0.933%). Overall, punctuation and whitespace-related tokens constitute a substantial portion of the corpus.

A.4 Effect on mathematics tasks

To evaluate the mathematical reasoning ability of the models, we assess their generation performance on math tasks. Figure 4 and 5 reports results on MATH-500 and GSM8K. The charts show the models’ ability to generate correct answers when given CoT rationales included in the dataset.

A.4.1 Math reasoning

The result trends differ slightly between the two benchmarks as in Figure 4 and 5. Similar to the QA experiments, the smallest model exhibits the most pronounced performance gains. For Gemma-3-4B-IT, performance peaks when 256 tokens are added in MATH-500, suggesting once again that the expanded space compensates for the limitations imposed by its smaller parameter size. We also observe that as the number of inserted tokens increases, performance suddenly drops beyond a certain point. In MATH-500, Llama-3.1-8B-IT initially exhibits a decline in performance, but its accuracy improves markedly when 256 filler tokens are added. In contrast, on GSM8K, Llama shows an initial performance gain, but its accuracy gradually decreases as more tokens are introduced. Qwen2.5-14B-IT, on the other hand, does not appear to benefit from the expanded space in either task: the space token yields only a minor improvement in the very early stages, while the <pad> token provides no observable effect.

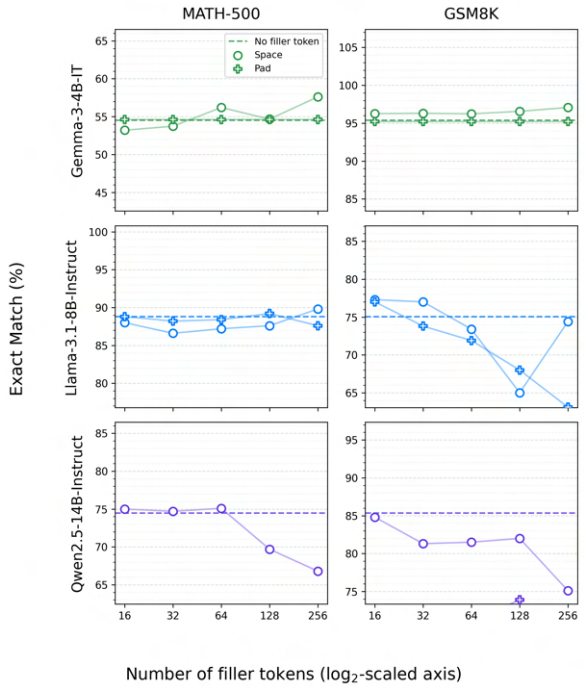

Figure 4: MATH-500 and GSM8K exact match (%) scores. Models perform better with ‘ ’ tokens than with <pad> tokens.

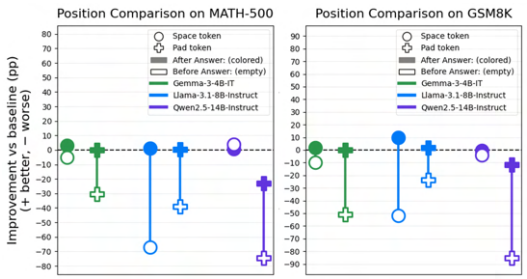

Figure 5: Position comparison between the filler tokens placed before and after the ‘`Answer:`’ token. When the filler tokens are placed *after* ‘`Answer:`’ token, the performance significantly degrades.

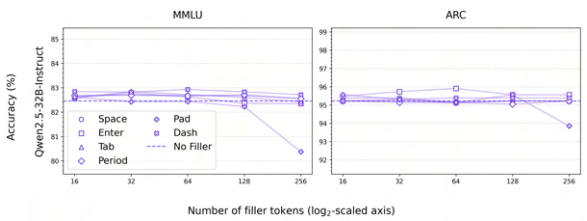

Figure 6: MMLU and ARC accuracy (%) results of Qwen2.5-32B-IT model.

A.5 Layer-wise answer prediction

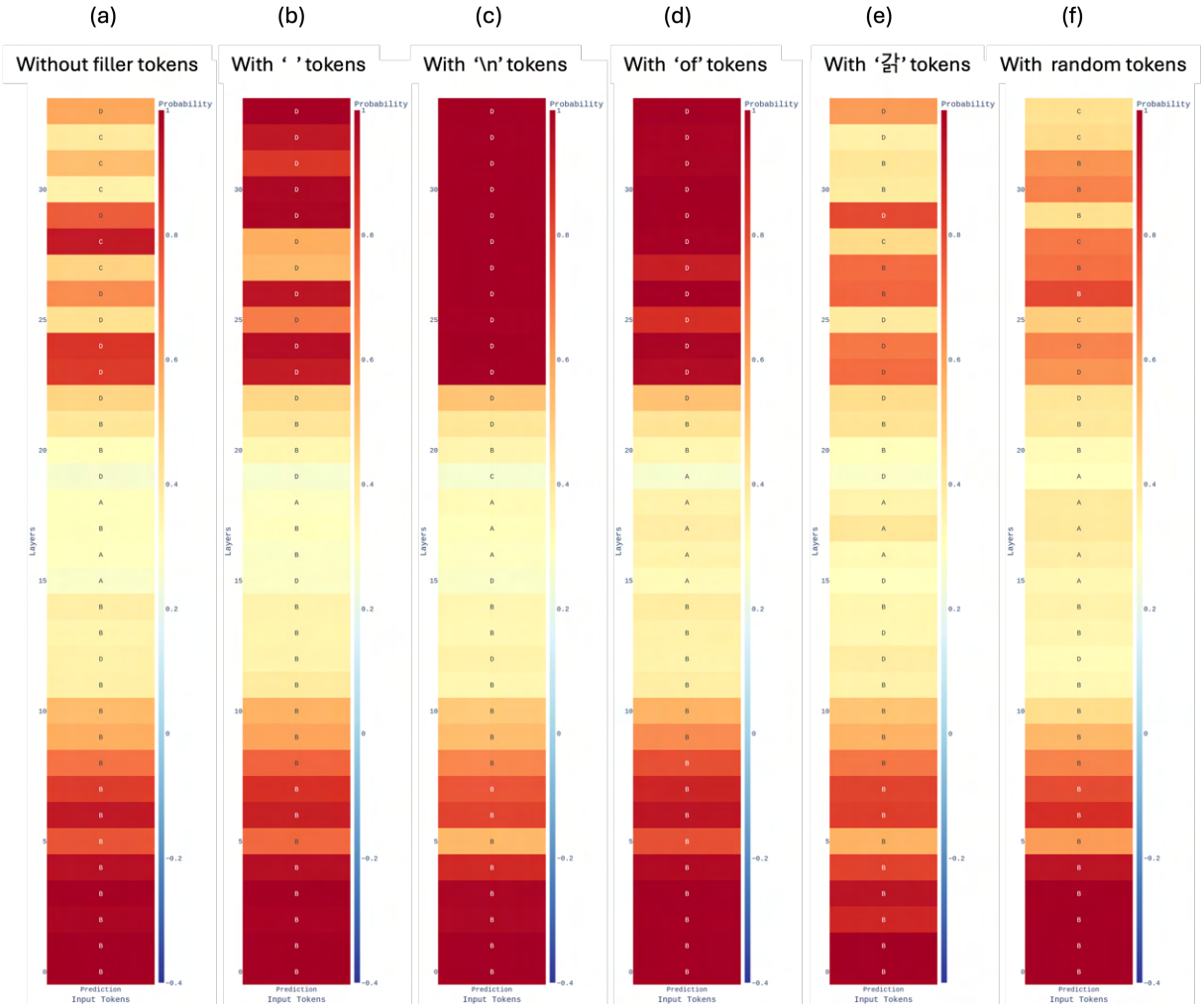

Figure 7: Layer-wise answer prediction without (a), with 16 ‘ ’ (b), ‘\n’ (c), ‘of’ (d), ‘갉’ (e) and random filler tokens (f) of `Gemma-3-4B-IT`. The same sample as Figure 8 and Figure 9 is used for the layer-wise prediction, where the gold answer is ‘D’. When ‘ ’, ‘\n’, or ‘`of`’ are used as fillers, the model confidently predicts the correct answer D from around the 2/3 layer mark. Without fillers, or when using an infrequent token (‘갉’), the model only settles on the correct answer at the final layers with low confidence, sometimes yielding incorrect outputs. Random tokens actively interfere with correct prediction.

A.6 Attention map: Success and failure cases

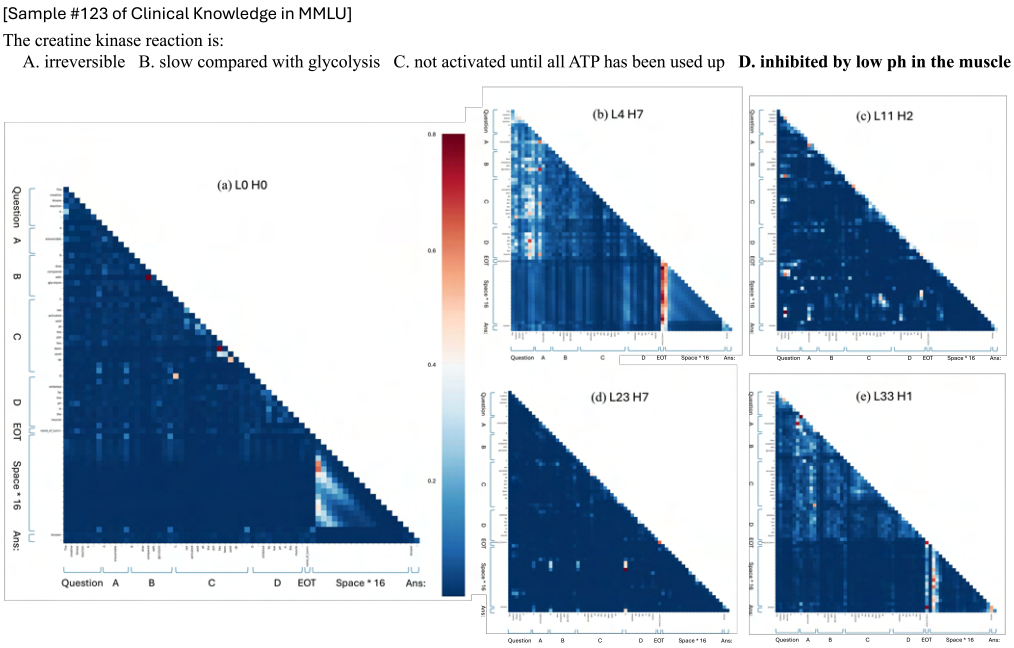

Figure 8: Attention maps of `Gemma-3-4B-IT` model with 16 of ‘ ’ tokens. This is the sample #123 of clinical knowledge in MMLU dataset. For simplicity, we do not include the texts in chat templates (e.g., ‘You are a helpful assistant...’) and the task explanations (e.g., ‘This is a multiple-choice...’) in this example. ‘L’ and ‘H’ in the figures indicate *layer* and *head* index, respectively. The ‘ ’ tokens tend to copy the attention pattern of the immediately preceding token (`<EOT>`), forming a consistent pattern. As shown in (c), they subsequently attend to the question and answer options, and as in (d), exhibit strong attention scores toward the correct answer (dark red square). Toward the final layers, attention shifts to the last tokens in preparation for generation.

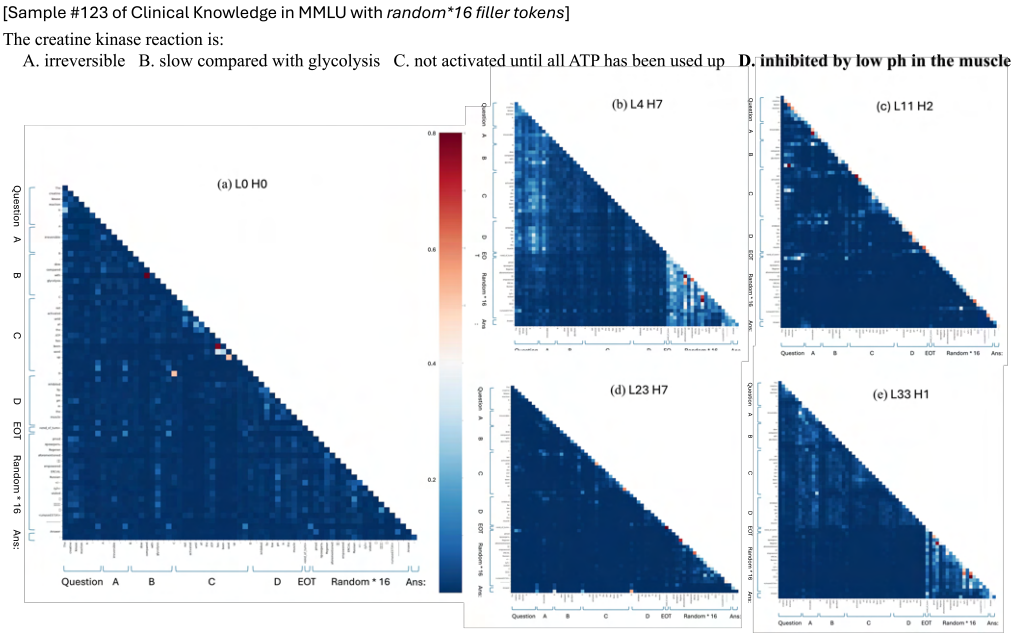

Figure 9: Attention maps of `Gemma-3-4B-IT` model with 16 of *random* tokens. The same sample as Figure 8. ‘L’ and ‘H’ in the figures indicate *layer* and *head* index, respectively. For visualization, we omit the chat template and task explanation for clarity. Random tokens exhibit relatively dispersed attention patterns, with weaker focus on the key information in the input.

A.7 Results with longer filler tokens

Table 2: MMLU and ARC accuracy (%) with ‘ ’ (space) tokens added. The numbers highlighted in yellow indicate performance improvements compared to the case without filler tokens. Smaller models exhibit larger performance gains, while performance begins to break down once more than 1024 tokens are added.

			MMLU		
M	`SmolLM2-1.7B-IT`	`Gemma-3-4b-it`	`Llama-3.1-8B-IT`	`Qwen2.5-14B-IT`	`Qwen2.5-32B-IT`
0	40.639	59.953	65.012	78.663	82.457
16	39.805	57.496	65.999	78.173	82.657
32	42.023	57.617	66.846	78.141	82.708
64	43.920	56.986	67.021	78.168	82.685
128	44.505	57.284	67.050	77.903	82.673
256	43.691	57.756	67.076	77.608	82.556
512	-	-	67.546	77.599	82.449
1024	-	-	64.394	78.662	81.969
2048	-	-	32.517	78.178	45.889
4096	-	-	27.453	48.695	27.328
8192	-	-	25.739	25.062	17.242

			ARC		
M	`SmolLM2-1.7B-IT`	`Gemma-3-4b-it`	`Llama-3.1-8B-IT`	`Qwen2.5-14B-IT`	`Qwen2.5-32B-IT`
0	42.747	74.915	78.584	92.150	95.222
16	48.123	75.171	79.949	92.150	95.392
32	51.280	74.232	79.778	92.235	95.392
64	52.986	73.891	80.290	92.235	95.137
128	52.645	73.635	81.058	91.980	95.222
256	52.901	73.976	80.973	90.358	95.222
512	-	-	81.058	90.700	95.051
1024	-	-	71.843	91.894	94.795
2048	-	-	25.683	92.235	36.775
4096	-	-	24.573	61.775	23.038
8192	-	-	26.365	22.611	14.420

A.8 MMLU Results

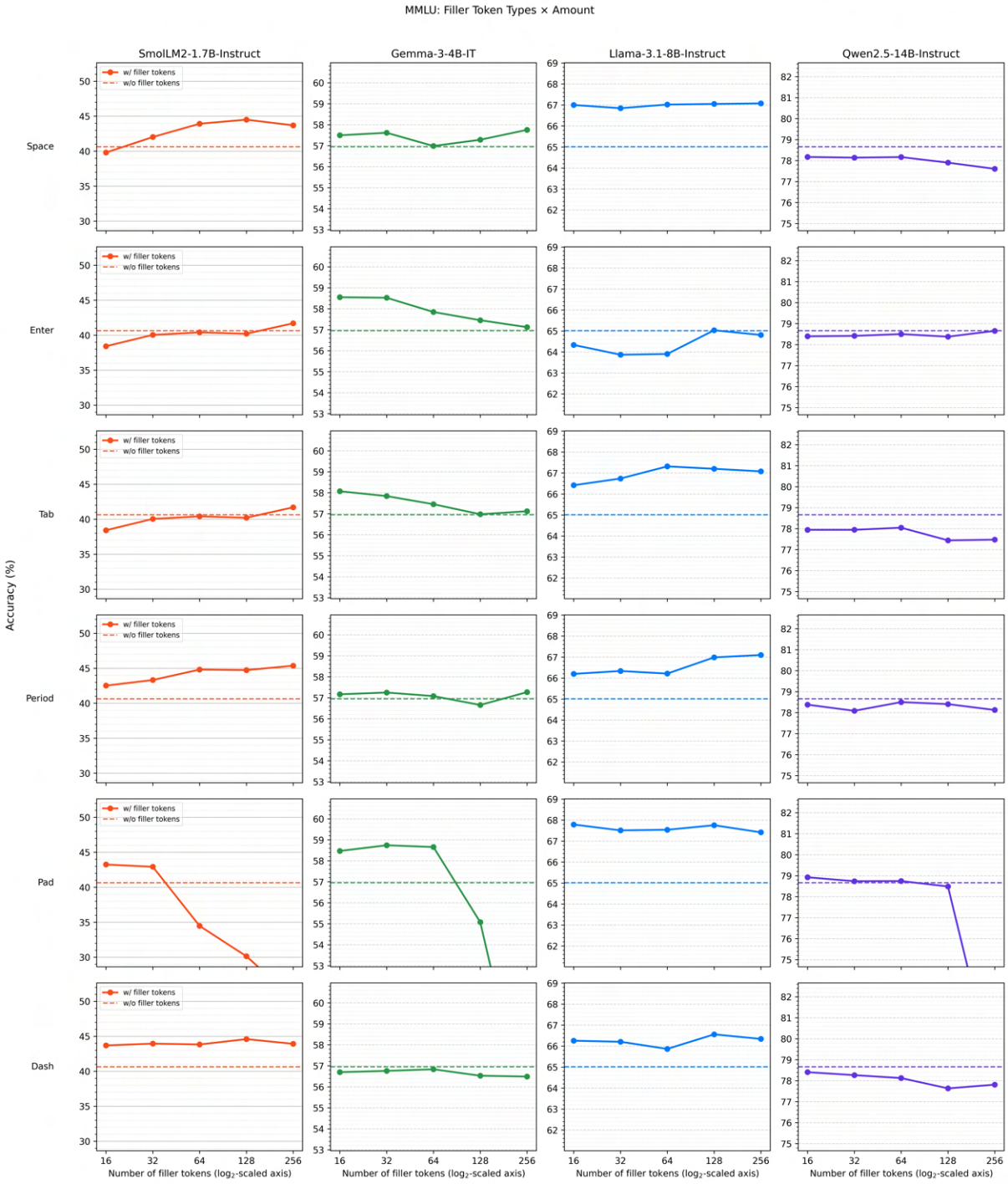

Figure 10: MMLU accuracy scores of models with each filler token type.

A.9 ARC Results

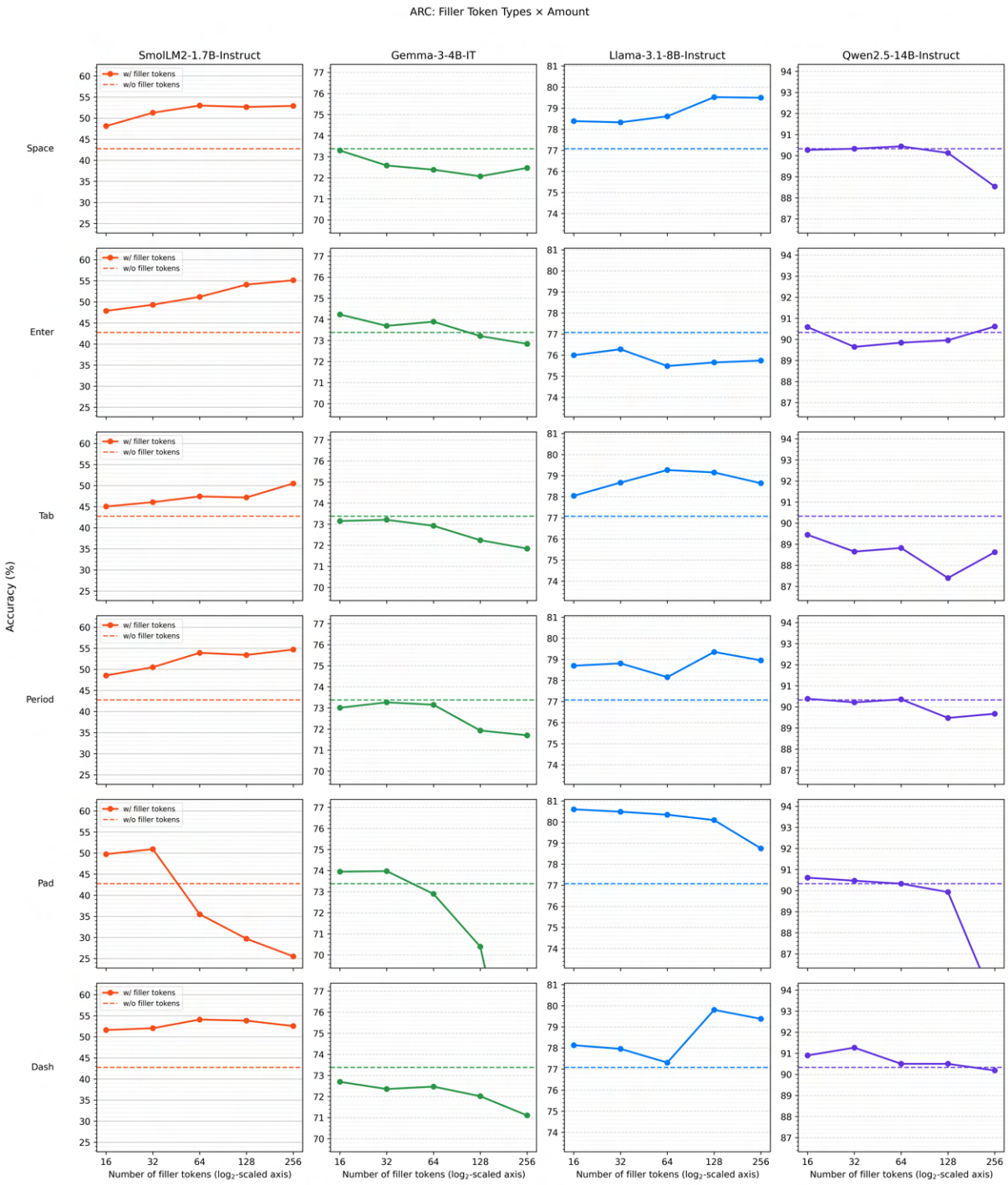

Figure 11: ARC accuracy scores of models with each filler token type.

A.10 Attention map examples

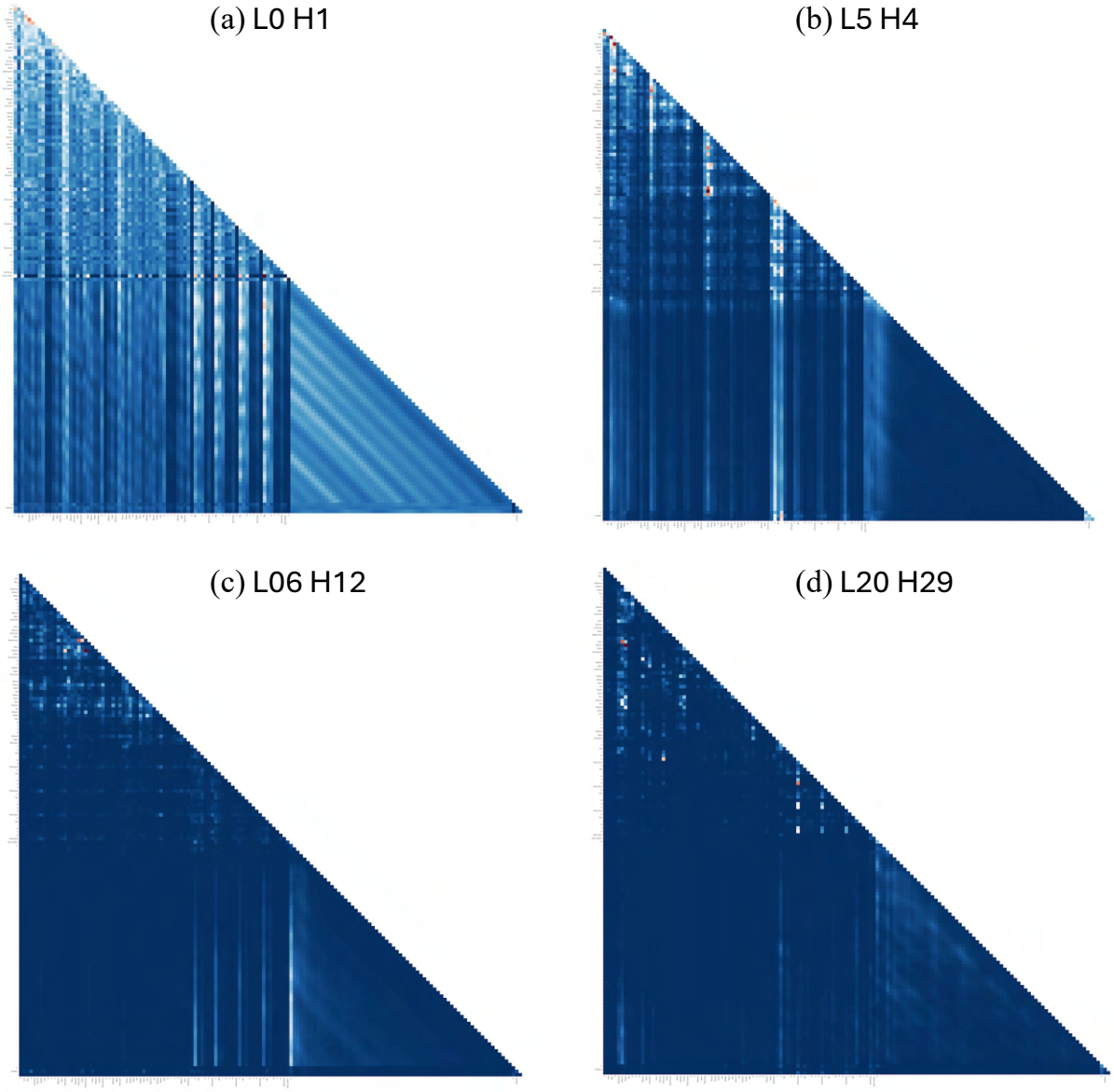

Figure 12: SmolLM2-1.7B-Instruct's attention maps with 64 of period tokens. This sample is 15 of elementary mathematics in MMLU.