SIRIUS: Contextual Sparsity with Correction for Efficient LLMs

Yang Zhou[1], **Zhuoming Chen**[1], **Zhaozhuo Xu**[2], **Xi Victoria Lin**[3], **Beidi Chen**[1,3]
[1]Carnegie Mellon Univeristy
[2]Stevens Institute of Technology
[3]FAIR at Meta
{yangzho6, zhuominc, beidic}@andrew.cmu.edu
zxu79@stevens.edu victorialin@meta.com

Abstract

With the blossom of large language models (LLMs), inference efficiency becomes increasingly important. Various approximation methods are proposed to reduce the cost at inference time. Contextual Sparsity (CS) is appealing for its training-free nature and its ability to reach a higher compression ratio seemingly without quality degradation. However, after a comprehensive evaluation of contextual sparsity methods on various complex generation tasks, we find that although CS succeeds in prompt-understanding tasks, CS significantly degrades the model performance for reasoning, deduction, and knowledge-based tasks. Despite the gap in end-to-end accuracy, we observed that sparse models often share general problem-solving logic and require only a few token corrections to recover the original model performance. This paper introduces SIRIUS[1], an efficient correction mechanism, which significantly recovers CS models quality on reasoning tasks while maintaining its efficiency gain. SIRIUS is evaluated on 6 models with 8 difficult generation tasks in reasoning, math, and coding and shows consistent effectiveness and efficiency. Also, we carefully develop a system implementation for SIRIUS and show that SIRIUS achieves roughly 20% reduction in latency for 8B model on-chip and 35% reduction for 70B model offloading. We will open-source our implementation once published.

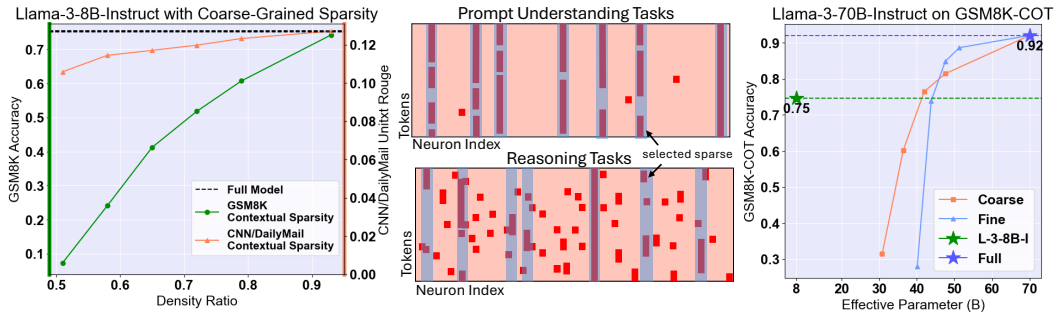
1 Introduction

Large Language Models (LLM), such as [22] (GPT-4), [31] (Gemini), and [33] (Llama) have demonstrated their proficiency in a wide range of natural language processing applications. Exploiting the model sparsity is a natural way to reduce the model parameter size and computational cost with a long history [13, 32]. More recently, many studies have shown that *contextual sparsity* [20, 16, 7, 14], which highly correlates to the prompt or the context, can greatly speed up LLM inference without quality degradation.

However, in this paper, we first demonstrate a critical and fundamental problem with *contextual sparsity* (CS): while generally robust in classification tasks and generation tasks that mainly rely

[1]We draw inspiration from the astronomical concept, in which SIRIUS refers to a two-body star system, where one is the brightest star ever detected, while the other is a dim star.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) Comparing Sparsity Method on CNN/DailyMail and GSM8K

(b) Visualization of Activation Maps across the generation sequence

(c) Contextual Sparsity applying to Llama-3-70B-Instruct

Figure 1: Contextual sparse models struggle at challenging text generation tests that require high-level reasoning and understanding, e.g. GSM8K. On these tasks, contextually sparse models lead to significant quality degradation. In (a), we contrast CS Llama-3-8B-Instruct on GSM8K (green) and CNN DailyMail (coral). (b) Contextual Sparsity Llama-3-70B-Instruct crashes at 50% global sparsity, making the smaller dense model Llama-3-8B-Instruct (green star) a significantly more efficient choice than the sparse 70B model. (c) Sparse model crashing at reasoning tasks has patterns, and ideally only correcting 11% unlikely tokens recovers the sparse model performance fully.

on prompt understanding (e.g., summarization, chat question-answering), we found that CS models struggle at high-level reasoning and understanding generation tasks. In Figure 1 (a), we contrast between the Text Summarization task (CNN/DailyMail) and Arithmetic Reasoning (GSM8K) with contextual sparsity methods on Llama-3-8B-Instruct. Varying sparsity levels, Llama-3-8B-Instruct with contextual sparsity performs consistently worse on GSM8K compared to CNN/DailyMail. CS on GSM8K crashes way earlier than summarization, which CS shows robust performance. Furthermore, shown in Figure 1 (b), the 50% Llama-3-70B-Instruct sparse model still has 4X the parameter size compared to the smaller dense model (Llama-3-8B-Instruct), while still performing worse on GSM8K-COT, rendering contextual sparsity utterly not useful for complex reasoning tasks. We notice that the overall reasoning pathway of these sparse models is usually sound, while the fatal mistake happens in the middle and propagates toward the end.

In this paper, we first conduct a simple experiment with CSparse Llama-3-8B-Instruct on GSM8K with the full model to check the sparse token-by-token. Surprisingly, only 6% tokens of the sparse model's generation corrected recovers most GSM8K accuracy. We are inspired by the observation and systematically design SIRIUS, a correction method. SIRIUS is empirically evaluated SIRIUS on 6 different models with 8 different Reasoning tasks and showed that SIRIUS is generally effective and efficient. On GSM8K and Llama-3-8B-Instruct specifically, we boost the fine-grained sparsity from 58% to 72% with 4% increase in effective parameter size and coarse-grained sparsity from 38% to 70% with the cost of 5% effective parameter size. SIRIUS is also shown to deliver the promised efficiency in both on-chip and offloading settings.

2 Related Works and Problem Formulation

In this section, we present the classification of the prior Contextual Sparsity methods. Additionally, we narrate important efficiency metrics in Appendix A.3, and show why Speculative Decoding applying directly to sparse correction would lead to substantial inefficiency Appendix A.4. For extended related works on model compression, contextual sparsity, and speculative decoding, we present in Appendix A.1.

Contextual sparsity (CS) methods are usually training-free, easy to use, and seemingly effective, making them highly attractive to ML practitioners looking to reduce LLM inference costs. The contextual sparsity selection is as follows: given the context, only a limited number of the most

Table 2: We show the difference between cases when Contextual Sparsity (CS) succeeds or fails. CS is generally good at prompt understanding tasks and tasks that measure the trustworthiness of the language models while not good at tasks that require reasoning and world knowledge understanding. MMLU* refers to MMLU-Flan-COT.

Where CS Succeeds Experiment Settings	**CNN/DailyMail** Unitxt Rouge	**CoQA** EM/F1	**TruthfulQA** Rouge-1/2 ACC
Llama-3-8B-Instruct	0.1237	0.6153/0.7825	0.4945/0.3647
Llama-3-8B-Instruct-CSparse	0.1144	0.6633/0.7977	0.4725/0.3403
Llama-3-8B-Instruct-FSparse	0.1166	0.6625/0.7984	0.5043/0.3305
Where CS Fails Experiment Settings	**GSM8K** ACC (strict/flexible)	**HumanEval** Pass@1 (GD)	**MMLU*** Accuracy
Llama-3-8B-Instruct	0.7551/0.7544	0.560	0.6231
Llama-3-8B-Instruct-CSparse	0.3859/0.3874	0.207	0.5558
Llama-3-8B-Instruct-FSparse	0.5868/0.5891	0.457	0.5304

relevant neurons are selected based on the input activation. The rest contributed to the output far less is discarded. We refer to two main directions of contextual sparsity methods as

- **Coarse-grained Sparsity** (CSparse) Methods ([7]) - that within the same input prompt, the sparsity pattern is fixed for all tokens generated.
- **Fine-grained Sparsity** (FSparse) Methods ([14]) - that exploits the per-token sparsity to save resources.

3 Observations

We build our 50% neuron sparsity implementation[2] of fine-grained sparsity based on [14] and coarse-grained sparsity based on [7].

Table 1: Llama-3-70B-Instruct with Offloading.

Settings	**Sparse**	**Sirius**	**Full**
Performance	0.7407	0.8719	0.9014
Latency (s)	3.57 s	3.68 s	5.72 s
Ratio to Full	0.6241	0.6434	1.0

For tasks on prompt understanding, CS generally performs well and gives consistent and strong output. We evaluate CS models on machine summarization (CNNDailyMail [26]), and Conversational Question Answering (CoQA [24]). On the other hand, contextual sparsity severely struggles when the generation tasks rely solely on the model's own reasoning and deduction ability, or the model's world knowledge understanding ability. Here we show the Llama-3-8B-Instruct and the Llama-2-7B-Chat models in Table 2. For a detailed analysis of results, we present more information in Appendix B.1, B.2 and B.3. We visually inspect extensive cases where the sparse model and dense differ in answers, more examples are shown in Appendix B.4. Shown in Figure 3, we found that a very small amount of correction would drastically improve the sparse model performance, showing a steep gradient when the percentage of corrected tokens is small. The experiment verifies our hypothesis that by correcting the small portion of key tokens, the sparse model can meet the large model's performance.

[2]Since [14] doesn't open-source its implementation and it relies on the threshold for determining the sparsity pattern, replicating the method isn't straightforward. Using a threshold also increases the difficulty of determining the actual density of the sparse model. Our implementation uses topk on the Gate Layer activations. The rest is implemented as described in the original method.

4 Methods

Naturally, letting the sparse model decide when to call the LLM for evaluation would be more flexible, but we show in Appendix C.1 that the sparse model cannot provide a reliable signal for consistently making the decision.

The full model is called once every kernel size. Throughout the inference, the KV cache is shared between the sparse and the full model. Therefore, full passes on its KV cache to benefit the small model's next chunk. The full model's KV helps the sparse model's output, which is evaluated later in the sections. Due to space limit, we place the full Algorithm in Appendix C as Algorithm 1. The LLM evaluation decision is based on comparing the full likelihood against a preset threshold. The overview is drawn in Appendix C Figure 7. Detailed ablation for threshold is in D.3. Also, we found that at the position when the full rejects the sparse's tokens, the second or third most probable tokens show strong coverage of tokens that can be accepted, and we further improve the efficiency of the SIRIUS efficiency by incorporating a hardware efficient tree building systems, details in C.2.

5 Experiments

In this section, we empirically evaluate SIRIUS to correct CS models on various generation tasks in complex reasoning. We show that SIRIUS is consistent in various tasks, effective in helping CS models recover their performance, and efficient in correction with low additional overhead.

Main Results - Due to space limits, we only select the best treewidth of SIRIUS for GSM8K, CSQA, and HumanEval for the main results in Table 6. Extensive studies on the rest 5 datasets with different treewidth are presented in the Appendix E. From Table 6, we can see that SIRIUS is consistently effective and efficient across all different classes of tasks. Specifically for Llama-3-8B-Instruct, besides GSM8K, SIRIUS corrects FSparse and CSparse, on CSQA, from 61% and 64% accuracy to 70% with cost only 3% sparsity for FSparse and 7% for CSparse respectively. On HumanEval, SIRIUS corrects FSparse from 45% to 61% with 4% sparsity overhead even surpassing the full model's performance, and from 20% to 52% with 8% sparsity as cost. Besides, Llama-3-8B-Instruct, SIRIUS corrects all 6 models with additional sparsity overhead smaller than 10% across these three datasets, further showing its strong efficiency. Besides results in Table 6, in Appendix E, we show that SIRIUS consistently shows great effectiveness with high efficiency across the rest of the 5 datasets.

Wallclock Speedup - Here we show that Sirius delivers its promised efficiency claim under two different settings, on-chip, and offloading, with two different models, Llama-3-8B-Instruct and Llama-3-70B-Instruct. Details are shown in Table 4. We consider the generation speedup only since contextual sparse methods are sparse mainly in generation, most use full weights for prefilling. For the two settings, we consider the coarse-grained sparsity method Griffin [7] because it open-sourced its implementation. On the other hand, the fine-grained sparsity method [14] relies on a custom CUDA kernel to achieve the target speed up, which they didn't open-source. The downstream dataset selected is GSM-8K COT, and the input sequence length is around 900. For the measurement on A40, L40, we found that kernel size 10 without tree delivers the best speedup, while a tree of treewidth 4 is built for A100 and H100. On average, Sirius with efficient implementation achieves 20% speedup on average for on-chip settings for Llama-3-8B-Instruct.

Secondly, we consider the offloading setting which is the only way for resource-limited users to run 70B models, which cannot easily fit in GPU clusters other than A100s and H100s. We use a single L40 48GB with a PCIe bus bandwidth of 25 GB/s. Llama-3-70B-Instruct is running on GSM8K-COT. Llama-3-70B-Instruct has roughly 80% of parameters to be MLP, which gives the theoretical APU for Griffin to be 0.6. Sparse + Sirius achieves 15.4 out of 16 average advance length, which in theory gives 0.649 APU, which is roughly what our system achieved.

Table 3: We show SIRIUS effectiveness and efficiency in the following table. We select GSM8K for Arithmetic Reasoning, CSQA for Commonsense Reasoning, and HumanEval for code generation. Under the "SIRIUS Perf. " column, A(B) is shown. A denotes the accuracy after SIRIUS correction in the dataset evaluated, while (B) represents the optimal treewidth selected under the current model dataset settings. Under the column of "AAL", X/Y is shown, where X is the AAL, while Y is the period.

GSM8K (Arithmetic Reasoning)						
Model	**Full Perf.**	**CSparse Perf.**	**CSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.7536	0.3844	0.65	0.7051 (8)	15.22/16	0.706
Llama-3-8B	0.4966	0.2085	0.65	0.4177 (8)	15.29/16	0.703
Model	**Full Perf.**	**FSparse Perf.**	**FSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.7536	0.5868	0.76	0.7278 (4)	15.37/16	0.807
Llama-3-8B	0.4966	0.3199	0.76	0.4579 (2)	15.03/16	0.825
CSQA (Commonsense Reasoning)						
Model	**Full Perf.**	**CSparse Perf.**	**CSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.7073	0.6470	0.58	0.7076 (8)	14.76/16	0.657
Llama-3-8B	0.6437	0.5585	0.58	0.6429 (8)	15.43/16	0.628
Model	**Full Perf.**	**FSparse Perf.**	**FSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.7073	0.6158	0.72	0.7043 (8)	15.66/16	0.753
Llama-3-8B	0.6437	0.533	0.72	0.6388 (1)	15.00/16	0.786
HumanEval (Coding)						
Model	**Full Perf.**	**CSparse Perf.**	**CSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.561	0.207	0.65	0.524 (8)	14.67/16	0.733
Llama-3-8B	0.262	0.067	0.65	0.243 (8)	15.10/16	0.691
Model	**Full Perf.**	**FSparse Perf.**	**FSparse Density**	**SIRIUS Perf.**	**AAL**	**Effective Density**
Llama-3-8B-Instruct	0.561	0.457	0.76	0.616 (6)	15.42/16	0.804
Llama-3-8B	0.262	0.189	0.76	0.298 (6)	15.54/16	0.797

Table 4: Performance and Speedup Ratios on GSM8K-COT with Different Hardware Configurations.

Settings	**ACC**	**A40**	**Ratio**	**L40**	**Ratio**	**A100**	**Ratio**	**H100**	**Ratio**
CSparse	0.3601	20.7 ms	0.66	15.6 ms	0.67	9.6 ms	0.72	6.6 ms	0.76
Sirius	0.7127	24.1 ms	0.78	18.2 ms	0.78	11.1 ms	0.83	7.7 ms	0.88
Full	0.7612	30.9 ms	1.0	23.2 ms	1.0	13.3 ms	1.0	8.6 ms	1.0

6 Conclusion

We observe that contextual sparse methods significantly degrade for reasoning and deduction tasks. With similar parameter size, the degradation from these sparsity is shown to be more severe for more well-trained models. However, we find that the degradation from contextual sparse models can theoretically be recovered with 10% token corrected by original model. Where to locate these small amount of tokens to effectively corrected by the full model is a difficulty. SIRIUS, an efficient correction mechanism, enables accurate LLM inference with contextual sparsity. With roughly 11% to 18% sparsity increase, SIRIUS improves fine-grained and coarse-grained sparsity significantly in their performance while maintaining their efficiency gain, reaching a reasonable tradeoff between performance and efficiency. We hope that this understanding inspires future work in this area.

7 Acknowledgement

We would like to thank Feng Liang and Yunong Liu for their helpful feedback during the exploration and writing. We also want to give a special thanks to Hanshi Sun for providing insights and suggestions for efficient implementation and speedup.

References

[1] bench authors, B. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

[2] Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. (2023). Accelerating large language model decoding with speculative sampling.

[3] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code.

[4] Chuang, Y.-S., Xie, Y., Luo, H., Kim, Y., Glass, J., and He, P. (2024). Dola: Decoding by contrasting layers improves factuality in large language models.

[5] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models.

[6] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems.

[7] Dong, H., Chen, B., and Chi, Y. (2024). Prompt-prompted mixture of experts for efficient llm generation.

[8] Frantar, E. and Alistarh, D. (2023). Sparsegpt: Massive language models can be accurately pruned in one-shot.

[9] Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

[10] Geva, M., Khashabi, D., Segal, E., Khot, T., Roth, D., and Berant, J. (2021). Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

[11] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding.

[12] Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. (2023). Speculative decoding with big little decoder.

[13] LeCun, Y., Denker, J., and Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, 2.

[14] Lee, J.-Y., Lee, D., Zhang, G., Tiwari, M., and Mirhoseini, A. (2024). Cats: Contextually-aware thresholding for sparsity in large language models.

[15] Leviathan, Y., Kalman, M., and Matias, Y. (2023). Fast inference from transformers via speculative decoding.

[16] Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., and Kumar, S. (2022). The lazy neuron phenomenon: On emergence of activation sparsity in transformers.

[17] Lin, S., Hilton, J., and Evans, O. (2022). Truthfulqa: Measuring how models mimic human falsehoods.

[18] Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. (2017). Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

[19] Liu, J., Xia, C. S., Wang, Y., and Zhang, L. (2023a). Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[20] Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., and Chen, B. (2023b). Deja vu: Contextual sparsity for efficient llms at inference time.

[21] Ma, X., Fang, G., and Wang, X. (2023). Llm-pruner: On the structural pruning of large language models.

[22] OpenAI, Achiam, J., et al. (2024). Gpt-4 technical report.

[23] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[24] Reddy, S., Chen, D., and Manning, C. D. (2019). Coqa: A conversational question answering challenge.

[25] Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., and Chandar, S. (2018). Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph.

[26] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks.

[27] Sharma, P., Ash, J. T., and Misra, D. (2023). The truth is in there: Improving reasoning in language models with layer-selective rank reduction.

[28] Song, Y., Mi, Z., Xie, H., and Chen, H. (2023). Powerinfer: Fast large language model serving with a consumer-grade gpu.

[29] Sun, H., Chen, Z., Yang, X., Tian, Y., and Chen, B. (2024a). Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding.

[30] Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. (2024b). A simple and effective pruning approach for large language models.

[31] Team, G., Anil, R., et al. (2024). Gemini: A family of highly capable multimodal models.

[32] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.

[33] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models.

[34] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

[35] Xia, M., Gao, T., Zeng, Z., and Chen, D. (2024). Sheared llama: Accelerating language model pre-training via structured pruning.

[36] 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., and Zettlemoyer, L. (2022). Opt: Open pre-trained transformer language models.

Appendix Table of Contents

A. Additional Background

1. Extended Related Works
2. Why 0.76?
3. Average Parameters Used Per Token
4. Why Not Using the Speculative Decoding to Correct Contextual Sparsity

B. Supplemental Observation

1. Contextual Sparsity: Where Does It Succeed?
2. Contextual Sparsity: Where Does It Fail?
3. Given Similar Parameter Size Well-trained Models Suffer More
4. Examples of Failure Cases

C. Supplemental Methods

1. Sparse Model's Self-Awareness Cannot Be Trusted
2. Hardware Friendly Tree Building Process

D. Supplemental Experiments

1. Additional Experiment Setting and Main Results
2. Large Model Experiments
3. Ablation: Various Aspects of Sirius Are Tested and Challenged
4. Variable Sequence Length with Batch Size One
5. Error Occurs at Which Position inside a Chunk
6. Miscellaneous Results
7. Llama-2 and Llama-3 Models on GSM8K-COT

E. Additional Results on Reasoning

1. Arithmetic Reasoning
2. CommonSense Reasoning
3. Code

A Additional Background

A.1 Extended Related Works

Pruning in LLM Sparsity in neural networks has been widely studied. In the context of LLM, sparsity is studied under two branches - unstructured and structured. On the unstructured sparsity side, [8] (SparseGPT) is a ground-breaking work that formulates pruning as a solving a series of sparse regression problems and proposes a fine solver for the problem. [30] (Wanda) introduces input activations into the pruning decision and achieves strong results in inducing LLM sparsity. On the structured side, LLMPruner [21] and Sheared Llama [35] each proposes different meticulous pruning algorithms and restoring weights through either parameter-efficient finetuning or efficient full weights training.

Contextual Sparsity Many recent works on LLM sparsity notice that the sparse pattern is highly related to the input or context. Deja Vu [20] revealed that for OPT models [36] the contextual sparsity is as high as 85%, meaning that 80% of the parameters can be pruned that won't hurt the

token decoded quality given the prompt. Deja Vu formulates the problem of neuron selection as a near-neighbor search problem: finding neurons that are the most similar to the input activations. PowerInfer [28] extends the contextual sparsity to benefit the heterogeneous setting. Compared to the rest of the model, MLP layers tend to possess significant contextual sparsity and can be effectively exploited in a training-free manner. Concurrently, Griffin [7] discovers the phenomenon of flocking, where MLP neurons have temporal locality, where given a fixed prompt, similar neurons tend to get activated throughout the following generation. Flocking is shown to occur in most activation types and open-source LLMs. Griffin selects the same set of heated neurons with 50% sparsity throughout the generation of each input prompt, which we refer to as coarse-grained sparsity. CATS [14] successfully exploits per-token contextual sparsity in the MLP layers for inference latency reduction. They resample a new set of neurons per every new input token, which we categorize it as fine-grained contextual sparsity. Our paper mainly focuses on the training-free MLP sparsity techniques. Although these recent works show minimal accuracy degradation in classification and easy text summarization tasks, they both severely degrade in generation quality under tasks that require high-level reasoning and understanding ability. Our work serves as a low-cost complementary tool, aiming to push these elegant and promising techniques for mainstream use cases.

Also, previous contextual sparsity methods haven't fully and exhaustively evaluated their benefits and limitations in downstream generation tasks. To fully study this technique, we extensively go through open-source LLMs in diverse performance and sizes on diverse generation tasks and datasets to locate where these sparse models maintain the performance or fail.

Speculative Decoding Besides model compression techniques, Speculative decoding [15], [2], [12] is another important LLM inference latency reduction method. Compared to LLM, small transformer models are much more computationally accessible and can effectively model short-range tokens. Therefore, smaller models are asked to speculate short-term future tokens, which the LLM takes in in parallel to trade in FLOPs with memory loading time. During verification, most speculative decoding methods pursue lossless acceleration, leading to frequent rollback during rejection. In contrast, Sirius solves a very different problem. Our method aims to maximally preserve the efficiency of sparse models while boosting its performance. Sparse models, pruned directly from LLM, are much stronger at modeling a longer range of text than draft models, thus requiring much less help from the LLM. Our work aims to find the minimum amount of LLM overhead while boosting its performance to the LLM level. Given the resemblance and relevance of Speculative Decoding to our method Sirius, we will elaborate more in-depth on their differences and Speculative Decoding's inefficiencies when it comes to helping the Sparse method in A.4.

A.2 Why 0.76?

Here we explain in greater detail why Sirius can achieve APU < 0.76 for Llama-3-8B with CSparse on GSM8K. For a threshold of 0.1, Sirius can correct Llama-3-8B coarse-grained sparsity from 20.85% to 43.9%, compared to the 49.66% full model. With a period of 16 tokens (gamma = 15), Sirius on average can accept 13.4 tokens out of a kernel size of 16 and over 9 tokens out of a kernel size of 10, translating to APU < 0.76, significantly lower than SD does.

A.3 Average Parameters Used Per Token

A key metric is used to evaluate the efficiency of our proposed method, the Average Parameter Used per token decoded (later referred to as APU). LLM inference is memory I/O bound [15], [12]. The latency of generating every single token is dominated by the memory loading time from the GPU HBM to SRAM. On the other hand, SIRIUS relies on full model parallel verifying a chunk of tokens. Although from the FLOPs standpoint, the amount of compute performed per evaluation step is the number of input token times of a single token input process, the latency of parallel verification is still roughly the same as taking a single token (Verified further in 10, length 64 is only 1.1 ms longer than length 1), because the inference is memory bound.

SIRIUS operates in the memory-bound regime (single inference sequence length smaller than or equal to 64). Thus, the average parameter count of a model gives us a rough judgment of the latency of inference. Formally, for a full LLM to have C_{full} number of parameters, and its sparse counterpart of a certain predetermined sparsity C_{sparse}. The average advancement length (later we refer to as AAL) in the number of tokens between two consecutive LLM corrections can be represented as n_{AAL}. The average parameters used per token (APU) are the following

$$\text{APU} = \frac{n_{sparse}C_{sparse} + C_{full}}{n_{AAL}} \tag{1}$$

We want the metric to be as small as possible, and obviously, we want n_{AAL} to be as large as possible.

Another thing to note is that we always compare the system's APU against the full model's APU, which is C_{full}. If we divided the above equation by C_{full}, we can have an equivalent parameter density of the system defined based on $I_{globalsparsity}$, which is C_{sparse}/C_{full}.

$$\text{Effective Density} = \frac{n_{sparse}I_{globalsparsity} + 1}{n_{AAL}} \tag{2}$$

Later, if we use period n_{period}, the equation can be rewritten as

$$\text{Effective Density} = \frac{(n_{period} - 1)I_{globalsparsity} + 1}{n_{AAL}} \tag{3}$$

Later when presenting SIRIUS, we mainly specify n_{period} with n_{AAL} to evaluate its efficiency. Notice that $I_{globalsparsity}$ is determined by the sparsity method, SIRIUS cannot change it anymore.

A.4 Why Not Using the Speculative Decoding to Correct the Sparse Model?

When Speculative Decoding is used to correct sparse using the full model, we will show that the efficiency of the overall process will be largely limited. We followed the common practice from speculative decoding and measured the acceptance rate on different datasets C4 [23] and GSM8K [6]. Take the Coarse-grained sparse model as an example. For Llama-3-8B as the full model, the 50% sparse (APU 0.65) model will produce an acceptance rate of 0.71 on C4 and 0.89 on GSM8K. Speculative decoding also use parallel verification in the period-basis. Naturally, to keep the system efficiency high, we need to (1) enlarge the period and (2) increase the average number of tokens accepted (AAL) given the gamma (period - 1) value. Take the acceptance rate of 0.89 on GSM8K as an example, following the formulation in [15], we can calculate the expected number of accepted tokens for every gamma term in the Speculative Decoding literature. AAL = $\frac{1-\alpha^{(\gamma+1)}}{1-\alpha}$. The trend (green) is plotted in Figure 2

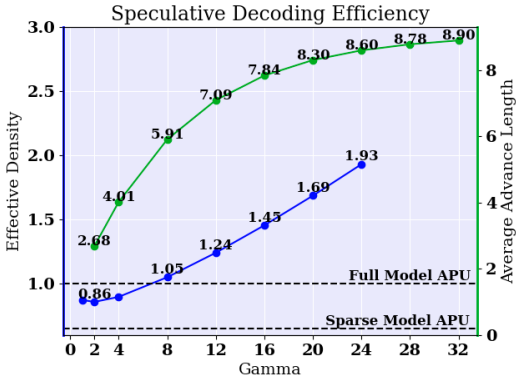


Figure 2: Speculative Decoding has limitation in efficiency when correcting sparse models.

We can notice the trend that the average advance length starts to plateau as the gamma becomes larger. Take the gamma of 16 as an example, the period is then 17. The average advance length is only 7.84. The APU is (16 * 0.65 + 1)/7.84 = 1.45, which is larger than the full model 1. The blue line in Figure 2 shows the relationship between APU and gamma.

Because of the plateauing effect, for an acceptance rate of 0.89, the best gamma is 2 (period = 3). The optimal APU is 0.86, compared with 0.65 coarse-grained sparse APU. A similar picture can be applied to Fine-grained sparsity as well. The key reasons for the observation are two-fold: (1) the

contextually sparse models are too big to be the draft model of the speculative decoding system and to have a large period; (2) Speculative decoding preserves the original model's performance so that the acceptance criteria are usually very strict, which is also not suitable for large period and high average advance length. Following the same spirit, [29] also uses a large draft model to do self-speculation, but for them, the authors select gamma = 1 to achieve the optimal speedup of their system. In contrast, SIRIUS brings <0.76 APU in this case with period ≥ 10. (Details in Appendix A.2)

B Supplemental Observation

In this section, we present a detailed study of the strengths and weaknesses of Contextual Sparsity (CS). B.1 presents the strengths of CS. B.2 presents the weaknesses of CS. In Section **??**, we show that given the similar parameter size, the more well-trained the model is, the more CS degradation will be for the model. B.4 shows our findings when looking into the failure cases of CS model in complex reasoning tasks.

Text generation is the main use case for these sparse models, which optimizes inference for the dense counterpart. Therefore, we choose not to look into text classification and language modeling ability, focusing on generation tasks.

In the following series of experiments, we build our implementation[3] of fine-grained sparsity based on [14] and coarse-grained sparsity based on [7]. The default sparsity for both methods is 50% for the MLP component of the model (whole MLP for coarse-grained sparsity and Up and Down linear layers only for fine-grained sparsity). We mainly use this default setting in most experiment tables in the paper without explicitly mentioning, or otherwise, we will explicitly specify the different sparsity levels we used.

B.1 Contextual Sparsity: Where Does It Succeed?

For tasks on prompt understanding, CS generally performs well and gives consistent and strong output. We evaluate CS models on machine summarization (CNNDailyMail [26]), and Conversational Question Answering (CoQA [24]).

The results show that the correctly selected contextual sparsity in the MLP layers and the full attention layers can fully extract and understand the local prompt information. More details are presented in Figure 4, where we show that by varying the sparsity level, the language model's performance on CNN/DailyMail is robust even when the activation sparsity drops to below 20%, which translates to around 44% global density.

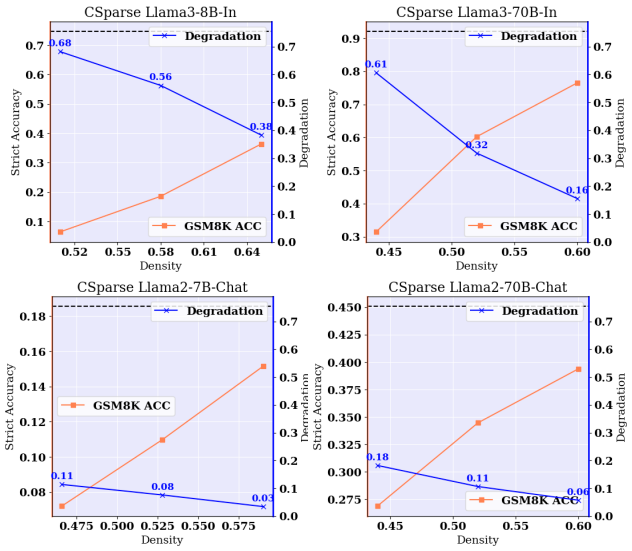


Figure 3: Given the similar model parameters, the more well-trained the model is, the worse the degradation would be. (Compare the figures vertically between Llama-3 and Llama-2 family models).

For tasks accessing factuality and hallucination, we select the generation portion of the TruthfulQA dataset [17]. Results are shown in Table 2, where we evaluate the techniques on 5 different LLMs. Interestingly, we find that the Fine-grained sparsity is often better than the dense model baseline across different models. This finding is consistent with previous works Laser [27] and Dola [4]. They both observed that compressing the original LLM in a carefully designed way would lead to improvement in factuality and better de-hallucination. Laser comes from the low-rank approximation of the MLP layers, while Dola proposes a factuality-aware layer-skipping algorithm. Based on their findings, hallucination

[3] Since [14] doesn't open-source its implementation and it relies on the threshold for determining the sparsity pattern, replicating the method isn't straightforward. Using a threshold also increases the difficulty of determining the actual density of the sparse model. Our implementation uses topk on the Gate Layer activations. The rest is implemented as described in the original method.

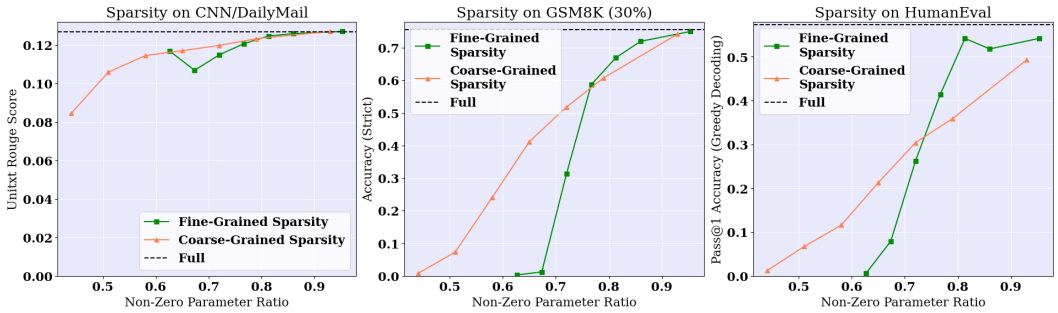


Figure 4: We contrast between Contextual Sparsity on prompt understanding task and complex generation tasks that require reasoning. (a) Both CSparse and FSparse are robust on CNN/DailyMail for various sparsity; (b) and (c) Show that both CSparse and FSparse crash on GSM8K and HumanEval at the global sparsity that they are still robust in prompt understanding tasks.

occurs when parts of the weights aren't as well-versed in the given input as the other parts. They expose the "averaging" effect that blurs the factuality of the output. Removing these neurons gives rise to better facutality and less hallucination. Our studies look at the same problem from a neuron sparsity standpoint.

B.2 Contextual Sparsity: Where Does It Fail?

On the other hand, contextual sparsity severely struggles when the generation tasks rely solely on the model's own reasoning and deduction ability, or the model's world knowledge understanding ability. Here we show the Llama-3-8B-Instruct and the Llama-2-7B-Chat models in Table 2, refer to Table 12 for evaluations on more models. Notice that since fine-grained sparsity method needs the activation from Gate MLP for selecting sparsity, while coarse-grained sparsity has a predetermined pattern after prefilling and can sparsify the Gate MLP. Even though both are at 50% activation sparsity, the coarse-grained sparsity method effectively achieves higher parameter savings than fine-grained sparsity in practice. Here we evaluate the sparse techniques using 5-shot CoT on the GSM8K dataset [6]. We found that across all the models we evaluated, both sparsity methods lead to significant accuracy degradation. Although code generation is not directly an arithmetic reasoning task, we found it useful to include, it since coding is a comprehensive evaluation of the language model's prompt understanding, reasoning, and planning to solve complex tasks. Therefore, we include HumanEval [3]. We found that both sparsity methods exhibit similar performance degradation when it comes to coding. Shown in Figure 4, two tasks see sparsity significantly drop performance after 50% activation sparsity.

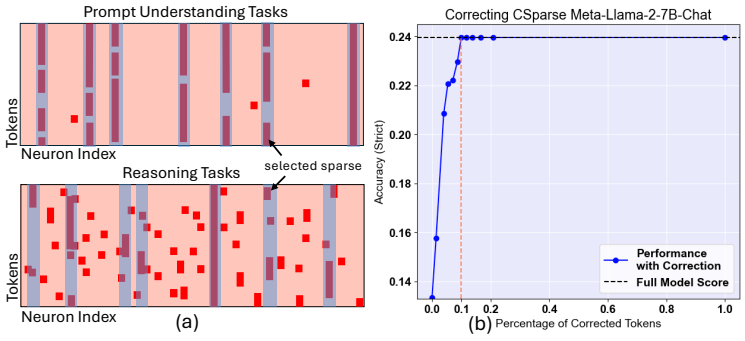


Figure 5: (a) Illustration on why Contextual Sparsity has uneven performance on different tasks. The activation heat map (red) has the brighter the color the larger in magnitude. On top, we also show the neuron sparsity selected. The graph points signify that the pattern in the prompt understanding task is easier to capture. (b) An additional graph of correcting Csparse Llama-2-7B-Chat. It is similar to the previous experiment on 8B. Only 10% tokens being corrected results in complete performance recovery.

For knowledge recall and world knowledge understanding, we specifically test on MMLU-Flan-CoT [5] the CoT text generation version of the MMLU dataset [11]. Table 2 shows the results. Stronger models like Llama-3-8B-Instruct suffer from significant degradation too.

B.3 Given Similar Parameter Size Well-trained Models Suffer More

We observe another interesting phenomenon: given the similar parameter size, the more well-trained the model is, the more performance degradation contextual sparsity would make on the full models. Here we present two pairs of results. First, we look at the performance between Llama-3-8B-Instruct and Llama-2-7B-Chat with Llama-3-70B-Instruct and Llama-2-70B-Chat. All models are evaluated on GSM8K-COT. We draw these models in CSparse in Figure 3, and the readers can find more results in Appendix D.7. We can see figures from top to bottom, where even at lower density (more elements are not selected), Llama-2-7B-Chat and Llama-2-70B-Chat suffer from less performance degradation (blue) compared to the Llama-3-8B-Instruct and Llama-3-70B-Instruct models. Furthermore, suppose we focus on Llama-3-70B-Instruct for global density at 60% or lower. In that case, the performance (coral) is degraded significantly, which is comparable or even lower to Llama-3-8B-Instruct full model performance at 0.76, Even at 50% density, the 70B model still has more than 40B parameters, much more expensive than the 8B model. The observation fully manifests the difficulty of using CS in complex reasoning tasks.

B.4 Examples of Failure Cases

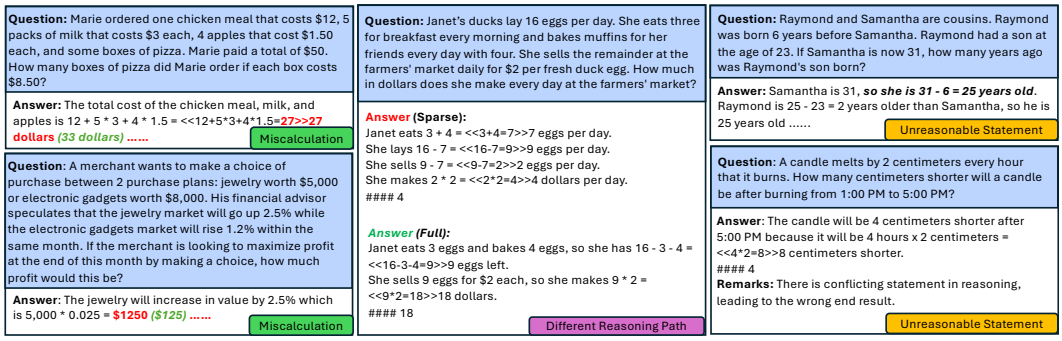


Figure 6: Examples of contextual sparse model making the identified three different types of mistakes. Most mistakes occur because the model makes calculation mistakes or has a wrong reasoning step compared to the full model. We also observe that there are rare cases where the model makes insensible statements in the middle that make the end result wrong.

We visually inspect extensive cases where the sparse model and dense differ in answers. Generally, the sparse model always produces highly similar answers to the dense model: the similar approach or logic flow when approaching the same problem and even the same number of sentences before the first mistake occurs or in success cases. However, the key differences are usually caused by the following three categories of small token-level mistakes: (1) frequent miscalculation in the intermediate steps, (2) wrong reasoning in intermediate steps, and (3) insensible and random statements. We find failure question-answer pairs provided in Figure 6 for each of the above-summarized cases. These mistakes happen in the middle of arguments and propagate to the wrong end result.

Similar observations can also be found for fine-grained sparse methods with different model types. *Interestingly, we find that even with these mistakes, the sparse model can still fully generate coherent tokens and make further reasoning assuming their prior steps are correct.*

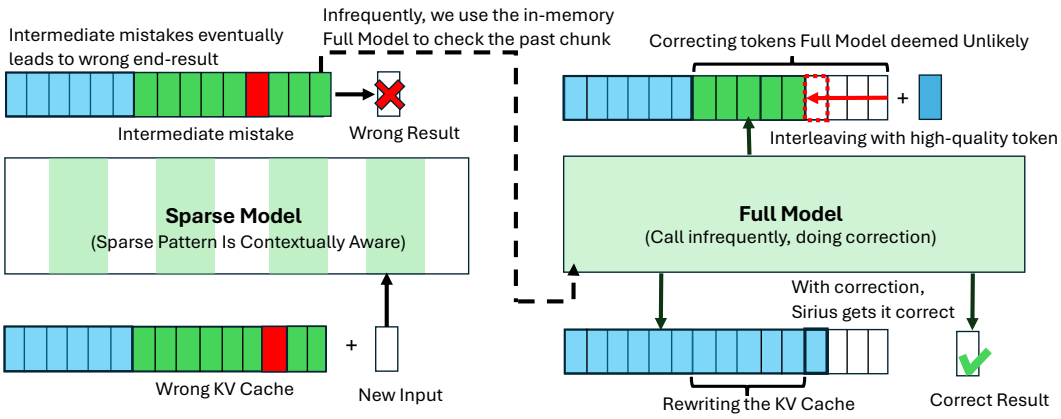


Figure 7: Overview of Sirius. Contextual Sparsity requires full model weights to be placed on the GPU memory. While the sparse model doesn't perform well on complex reasoning tasks, Sirius uses the Full Model to correct the Sparse model. The full model is called fairly infrequently. During the correction, the Full Model will rewrite the KV Cache, interleave with high-quality tokens to the sparse outputs, and then roll back only when the token is deemed extremely unlikely by the Full Model.

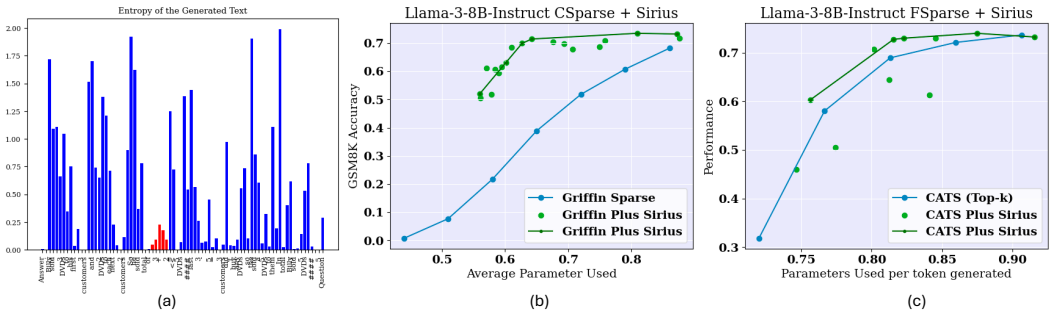


Figure 8: In (a), we present an example that illustrates why the signals from the sparse model are unreliable. It is a figure plotting entropy versus generated tokens. At the tokens where the sparse made the mistake (red), the entropy isn't in large spikes which signifies chaos and low confidence, rather it is even quite low, compared to nearby entropy spikes. In (b) and (c), we view Sirius as a compression method by itself. We compare Sirius with contextual sparse methods and show that given the same parameter used, Sirius performs better than Contextual Sparse Methods on GSM8K.

C Supplemental Method

C.1 Sparse Model's Self-Awareness Cannot Be Trusted

Intuitively, rather than fixing the n_{sparse} number, letting the system decide when to call the LLM for evaluation would then give more flexible n_{sparse}. In other words, the problem becomes how to make the sparse model decide when the LLM can be called. Nevertheless, we argue that the sparse model's output probability distribution cannot be used as a metric for accuracy decisions.

We empirically experiment with various methods to utilize the information contained in the sparse model's output distribution. However, it always leads to n_{sparse} being too short, the single-digit number for GSM8K (around 4 for Llama-3-8B Instruct). setting up the threshold isn't useful. We then discovered that the sparse model has very limited self-awareness of its own mistakes. The sparse model can be very confused with small top-1 likelihood numbers and larger entropy when making mistakes. However, we also frequently find examples where the sparse model is very confident in their mistakes. To make the observation concrete, we present a small example in Figure 2 a piece of text where the sparse model makes a mistake while the full model succeeds. The red bars signify the

Algorithm 1 Sirius

Require: Prompt $[x_1, ..., x_t]$, full model M_F, and sparse model M_S sharing weights and sharing KV Cache C, $cache_pos$ is the location where new k and v are written to C, kernel size n

Require: forward function FORWARD, threshold r, which is a value used by M_F to judge whether the token occurs likely enough

Require: StoppingCriteriaMet() downstream task-specific, returns a boolean

while not StoppingCriteriaMet() **do**
  $i \leftarrow 0$
  $kernel \leftarrow$ empty
  $cache_pos \leftarrow 0$
  while $i < n$ **do**
    set $\hat{p_{t+i}} \leftarrow$ FORWARD(M_C, C, $[x_1, ..., x_t]$, $[x_{t+1}, ..., x_{t+i-1}]$, $cache_pos$)
    ▷ Running sparse model
    $cache_pos \leftarrow cache_pos + 1$
    sample $\hat{x_{t+i}} \sim \hat{p_{t+i}}$
    $kernel \leftarrow$ cat($kernel, \hat{x_{t+i}}$)
    $i \leftarrow i + 1$ ▷ Before exiting, $kernel$ $[x_t, ..., x_{t+n}]$
  end while
  $cache_pos \leftarrow cache_pos$ subtracts n ▷ Enables Full to directly rewrites KV Cache
  set $[q_t, ..., q_{t+i}] \leftarrow$ FORWARD(M_C, C_S, $cache_pos$, $kernel$)
  for j from 0, n **do**
    if $q_{t+j} < r$ **then** ▷ Full rejects
      break ▷ j stores the first token position being rejected
    end if
  end for
  $cache_pos \leftarrow j + 1$ ▷ Rollback
  $kernel \leftarrow$ empty
  sample $x_{t+j+1} \sim p_{t+j}$ ▷ Interleaving Key Token
end while

error location. The token entropy is neither high nor at zero, making it impossible to effectively use a threshold to control the number n_{sparse}. We deployed fixed n_{sparse} at 16 to 24 based on results tuning on a small validation set which works effectively in practice.

C.2 Hardware Friendly Tree Building Process

In this section, we first look at the insights behind whether building the tree can help efficiency, then we detail the specific steps towards tree pruning.

The goal for the Sirius system is to make n_{AAL} to be as large as possible. Despite the full model sharing KVs with the sparse model, Sirius still encounters costly rollbacks because of sparse greedily decoded tokens being rejected. Interestingly, we look closely into where the sparse model is likely to make a mistake on GSM8K and AQuA-RAT-COT [18] with Sirius on Llama-3-8B-Instruct and a kernel size of 16. More details are shown in Appendix D.5. The error distributes almost uniformly across all positions of the kernel size. Also, when the token makes the mistake, besides the greedily decoded tokens, we find that other tokens of lower likelihood offer the potential to boost efficiency. Surprisingly, we found that out of the cases where the greedily decoded tokens are rejected, the probability that the second or third most likely tokens from the sparse being accepted by the full model is reasonably high.

Table 5: The second and third most likely tokens from sparse models offer potential for boosting efficiency.

Sparsity	2nd Hit	3rd Hit	Miss	Coverage%
FSparse	79%	11%	9%	90%
CSparse	65%	17%	16%	82%

Shown in Table 5, we test on part of the GSM8K rejected cases. The "Second Hit" is defined as the count of the second most likely tokens being accepted by the full model when the greedily decoded token is rejected, while the "Third Hit" is defined as the count of the third most likely token being accepted when the first two are rejected. Both sparsity method has a high acceptance rate, or "Coverage", from the second and third most likely tokens when the most likely token is rejected, showing huge potential for gains in efficiency.

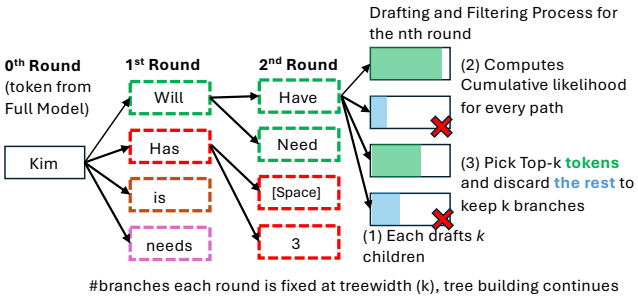


Figure 9: Illustration of Tree Building Process.

To capitalize the potential from the second to third tokens, we propose to build a tree during the sparse generating process (lines 6 to 11 in Algorithm 1. The tree algorithm is similar to Beam Search [9]. However, to make sure that the tree building and tree parallel correction processes can achieve speedup over cases that don't build trees, we impose strong restrictions on the tree structure we build. For a fixed kernel size, we limit every step to having a fixed number of leaves, or treewidth, through tree pruning based on ranking the cumulative log-likelihood of the path. The resulting tree has a fixed shape for a given kernel size and tree width, but only the interconnection pattern between steps varies based on the pruning and ranking within each step. The details are illustrated in Figure 9. During verification, out of the treewidth complete paths, we select the one that reaches the longest advance length. In practice, we found that for kernel size 16, when the treewidth is increased to 8, the optimal verification tree is around 64. From Section A.4, we see that the parallel verification of the tree of 64 roughly equals the time the full input 1 token.

Therefore, a treewidth of 8 is set as the maximum treewidth when building the tree for kernel size 16 for later. We show that building a tree makes the system significantly more efficient while retaining correction effects.

D Supplemental Experiments

D.1 Additional Experiment Setting and Main Results

Models and Datasets - To comprehensively evaluate SIRIUS performance, we deploy six mainstream LLMs with sizes ranging from 7B to 13B: Llama-2-7B, Llama-3-8B, and Llama-2-13B with their instruction finetuned counterparts, all from Llama family. Following prior milestone [34] in LLM reasoning, we also tested CS models on two popular types of reasoning generation tasks: arithmetic and commonsense reasoning. On the Arithmetic side, besides GSM8K, we also evaluate CS models on AQuA-RAT. On the Common Sense side, we use CSQA [25], StrategyQA[10], Date, and Sports, where the last two are from Big Bench Suite [1]. Most of these tasks are originally classification tasks. Following the instruction in [34], we manually compose COT prompts to transform these into logic argument generation tasks. Besides, we found that the CS models do perform not well in code generation, which also requires forming logical arguments and planning. We select HumanEval [3] and MBPP+ [19] two high-quality Python coding tasks to see whether SIRIUS corrects these problems.

For arithmetic reasoning and coding, we use 50% neuron sparsity for both CSparse and FSparse. FSparse relies on the gate layer to be dense, leading to higher global density than CSparse. Since commonsense reasoning tasks are generally less logically challenging comparatively, we lowered the neuron sparsity level to 40%.

D.2 Large Model Experiments

In this section, we provide several supplemental experiments to the picture. First, we run SIRIUS on Llama-3-70B. However, because of computational limits, we cannot run SIRIUS with the tree on Llama-3-70B with the scale we did for other models. Nevertheless, we do show that 70B has roughly the same pattern as we have seen before, large model sparsity also somehow struggles on reasoning tasks. Second, we provide additional proof for the parallel verification efficiency statement. After that, I show results on where the error is located in the chunk size of 16 tokens. The error is distributed almost uniformly. Last but not least, we also apply SIRIUS on datasets that are reasoning. Lastly, we provide more results on the comparison between models of similar size but have a huge performance gap. We show that given the similar parameter size, the trend is for a more well-trained, powerful model to degrade more from contextual sparsity.

To diversify the evaluation of Sirius, we also evaluate Sirius's Effectiveness on the Llama-3-70B-Instruct model. MMLU is subsampled 10%, while CNN/DailyMail is subsampled 30%. The following table contrasts with Llama-3-8B-Instruct. We use strict match/flexible extract accuracy for GSM-8K-COT, accuracy for MMLU, F1/EM score for CoQA, Rouge-1/2/L score for CNN/DailyMail, and Rouge-1/2 ACC for TruthfulQA.

D.3 Ablation: Various Aspects of Sirius Are Tested and Challenged

Probing Components To understand the contribution and the utility of each component of Sirius, we ablate all components of Sirius in Table 8. We started by only letting the LLM correct the token it is evaluating (interleaving only). Then, we add on top of it the KV cache correction, and then the rollback. All these three techniques are effective when applied solely. Rollback seems to be the most effective technique. Even when applied alone, rollback asserts significant correction to both the CSparse and FSparse models. Interestingly, KV Cache is also effective alone, bringing a 12% increment for CSparse and an 11% accuracy increase for FSparse. Relatively, interleaving is the weakest. Surprisingly, adding both KV rewriting and rollback is only marginally better than rollback alone. Although it is tempting to think KV Cache rewriting is not useful with rollback, the improvement KV Cache Rewriting brings is a gain in efficiency. When adding the KV Cache Rewriting on top of Roll Back and interleave it significantly improves the efficiency of the correction. For CSparse, adding KV rewrite increases AAL from 12.77 to 13.80.

Table 6: We show SIRIUS effectiveness and efficiency in the following table. We select GSM8K for Arithmetic Reasoning, CSQA for Commonsense Reasoning, and HumanEval for code generation. Under the "SIRIUS Perf. " column, A(B) is shown. A denotes the accuracy after SIRIUS correction in the dataset evaluated, while (B) represents the optimal treewidth selected under the current model dataset settings. Under the column of "AAL", X/Y is shown, where X is the AAL, while Y is the period.

GSM8K

Model	Full Perf.	CSparse Perf.	CSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.7536	0.3844	0.65	0.7051 (8)	15.22/16	0.706
Llama-3-8B	0.4966	0.2085	0.65	0.4177 (8)	15.29/16	0.703
Llama-2-7B-Chat	0.2403	0.1334	0.69	0.2244 (8)	15.00/16	0.757
Llama-2-7B	0.1357	0.0758	0.69	0.1183 (6)	15.87/16	0.715
Llama-2-13B-Chat	0.3548	0.2714	0.68	0.3381 (4)	15.34/16	0.730
Llama-2-13B	0.2282	0.1759	0.68	0.2418 (1)	15.34/16	0.730

Model	Full Perf.	FSparse Perf.	FSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.7536	0.5868	0.76	0.7278 (4)	15.37/16	0.807
Llama-3-8B	0.4966	0.3199	0.76	0.4579 (2)	15.03/16	0.825
Llama-2-7B-Chat	0.2403	0.1971	0.79	0.2388 (6)	15.69/16	0.819
Llama-2-7B	0.1357	0.1137	0.79	0.1410 (4)	15.91/16	0.807
Llama-2-13B-Chat	0.3548	0.3222	0.78	0.3533 (1)	15.08/16	0.842
Llama-2-13B	0.2282	0.2191	0.78	0.2372 (4)	15.92/16	0.797

CSQA

Model	Full Perf.	CSparse Perf.	CSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.7073	0.6470	0.58	0.7076 (8)	14.76/16	0.657
Llama-3-8B	0.6437	0.5585	0.58	0.6429 (8)	15.43/16	0.628
Llama-2-7B-Chat	0.6248	0.5200	0.62	0.6175 (8)	15.07/16	0.683
Llama-2-7B	0.4742	0.4414	0.62	0.4742 (8)	15.80/16	0.652
Llama-2-13B-Chat	0.6879	0.5536	0.61	0.6691 (4)	11.43/12	0.674
Llama-2-13B	0.6109	0.5601	0.61	0.6060 (4)	15.72/16	0.645

Model	Full Perf.	FSparse Perf.	FSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.7073	0.6158	0.72	0.7043 (8)	15.66/16	0.753
Llama-3-8B	0.6437	0.533	0.72	0.6388 (1)	15.00/16	0.786
Llama-2-7B-Chat	0.6248	0.6167	0.75	0.6380 (4)	15.09/16	0.811
Llama-2-7B	0.4742	0.4717	0.75	0.5012 (6)	15.89/16	0.771
Llama-2-13B-Chat	0.6879	0.533	0.74	0.6691 (4)	14.30/16	0.846
Llama-2-13B	0.6109	0.5700	0.74	0.5864 (4)	15.72/16	0.770

HumanEval

Model	Full Perf.	CSparse Perf.	CSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.561	0.207	0.65	0.524 (8)	14.67/16	0.733
Llama-3-8B	0.262	0.067	0.65	0.243 (8)	15.10/16	0.691
Llama-2-7B-Chat	0.140	0.067	0.69	0.159 (8)	10.88/12	0.789
Llama-2-7B	0.116	0.079	0.69	0.128 (8)	14.84/16	0.765
Llama-2-13B-Chat	0.189	0.122	0.68	0.171 (8)	11.12/12	0.762
Llama-2-13B	0.262	0.067	0.68	0.244 (8)	15.10/16	0.741

Model	Full Perf.	FSparse Perf.	FSparse Density	SIRIUS Perf.	AAL	Effective Density
Llama-3-8B-Instruct	0.561	0.457	0.76	0.616 (6)	15.42/16	0.804
Llama-3-8B	0.262	0.189	0.76	0.298 (6)	15.54/16	0.797
Llama-2-7B-Chat	0.140	0.134	0.79	0.165 (6)	15.27/16	0.841
Llama-2-7B	0.116	0.116	0.79	0.165 (6)	15.86/16	0.810
Llama-2-13B-Chat	0.189	0.146	0.78	0.183 (6)	15.34/16	0.827
Llama-2-13B	0.246	0.233	0.78	0.259 (4)	15.85/16	0.801

Table 7: Large model results on miscellaneous datasets.

	GSM-8K-COT	MMLU	CoQA	CNN/DailyMail	TruthfulQA
Llama-3-70B-In	0.9014/0.9022	0.7456	0.6567/0.8069	0.101634/0.020614/0.096413	0.5116/0.4247
+ CSparse	0.7407/0.7483	0.7018	0.6497/0.8046	0.101922/0.020854/0.096703	0.4541/0.3807
+ FSparse	0.8726/0.8772	0.7193	0.6497/0.8035	0.101505/0.020623/0.096344	0.4835/0.3905
Llama-3-8B-In	0.7612/0.7672	0.6272	0.6153/0.7825	0.101523/0.020481/0.096311	0.4945/0.3647
+ CSparse	0.3601/0.3647	0.5307	0.6003/0.7735	0.101681/0.020657/0.096432	0.5067/0.3953
+ FSparse	0.6103/0.6202	0.4825	0.5828/0.7577	0.101713/0.020448/0.096516	0.5202/0.3941

Table 8: Ablation on Components in Sirius.

CSparse	**GSM8K 20%**	**FSparse**	**GSM8K 20%**
Llama3-8B-Instruct	0.7538/0.7538	**Llama3-8B-Instruct**	0.7538/0.7538
+ CSparse	0.3674/0.3674	+ FSparse	0.5644/0.5644
+ CSparse + Interleave	0.3826/0.3826	+ FSparse + Interleave	0.6288/0.6288
+ CSparse + KV Rewrite	0.4735/0.4735	+ FSparse + KV Rewrite	0.6629/0.6629
+ CSparse + KV Rewrite + Interleave	0.4886/0.4886	+ FSparse + KV Rewrite + Interleave	0.6780/0.6818
+ CSparse + Roll back + Interleave	0.6591/0.6591	+ FSparse + Roll back + Interleave	0.7273/0.7273
+ CSparse + KV Rewrite + Interleave + Rollback	0.6667/0.6667	+ FSparse + KV Rewrite + Interleave + Rollback	0.7273/0.7311

Likelihood threshold to balance Correction and Efficiency We found that the likelihood threshold is important for managing the Sirius correction and efficiency tradeoff. We present results in Table 9. We ablate this setting on a 30% subsampled GSM8K dataset, and only strict accuracy is reported. The performance is the score, while the efficiency is measured by Average Advance Length (AAL). We can find that with the increase of threshold, the scores generally improve, while the efficiency metric decreases.

Building Wider Tree We study the effect of increasing the treewidth. In fact, for every number from SIRIUS in Table 6, we are selecting from a group of results by different treewidth. We present all of this treewidth and its corresponding accuracy and efficiency numbers in the Appendix E. *Importantly, raising treewidth always improves AAL.* Although different choices of treewidth usually give similar accuracy scores, there is hardly a pattern on which treewidth always gives the best accuracy. The optimal treewidth can only be found through empirical studies.

D.4 Variable Sequence Length with Batch Size One

Here we show the benchmark latency on A100, where the input tensor to Llama-3-8B-Instruct has a shape of batch size 1 and a different input sequence length. To get the hardware optimal readings, we use torch compile to compile the whole forward pass of the model. We show that the latency only goes up insignificantly to 64, but the trend of increment to 96 is a bit steep.

D.5 Error Occurs At Which Position inside a Chunk

We look at the distribution of where the error would be inside a kernel of 16 tokens. We run through Sirius with a kernel size of 16 on the entire GSM-8K and AQuA-RAT-COT dataset. The histogram is shown in Figure 10. We found that the error occurs in a uniform pattern, where it is hard to see any particular region where the tokens are likely to occur the most.

Table 9: Ablation on the threshold for correction (FSparse Llama-3-8B-Instruct).

Threshold	**Full**	**Sparse**	**0.05**	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Accuracy	0.7803	0.5884	0.7247	0.7399	0.7399	0.7677	0.7702	0.7652
AAL	N/A	N/A	15.2	14.6	11.6	8.5	6.2	4.2

Table 10: A100 Latency versus Input Sequence Length.

Input Sequence Length	**A100 Latency (ms)**
1	0.0133
2	0.0135
4	0.0136
8	0.0138
16	0.0140
32	0.0149
64	0.0144
96	0.0171

D.6 Miscellaneous Results

Besides, the results on the complex reasoning tasks, we evaluate Sirius on slightly more diverse tasks in Table 12.

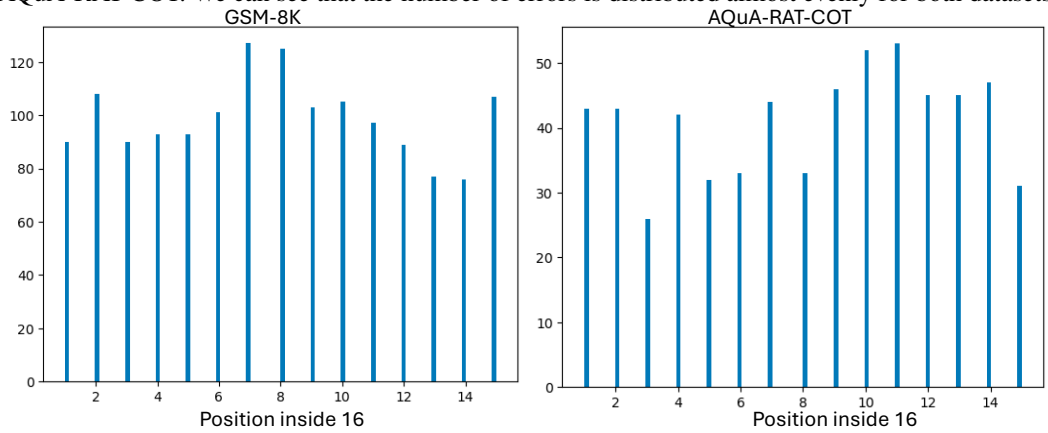
D.7 Llama-2 and Llama-3 Models on GSM8K-COT

Table 11: Detail on Llama-2 and Llama-3 family models with CS.

Llama-3-70B-Instruct	**Accuracy**	**Degradation**	**Llama-3-8B-Instruct**	**Accuracy**	**Degradation**
Full	0.9205		**Full**	0.7462	
Csparse 60%	0.8144	0.1061			
Csparse 50%	0.7652	0.1553	**Csparse 50%**	0.3636	0.3826
Csparse 40%	0.6023	0.3182	**Csparse 40%**	0.1856	0.5606
Csparse 30%	0.3144	0.6061	**Csparse 30%**	0.0644	0.6818
Fsparse 50%	0.8864	0.0341	**Fsparse 50%**	0.6477	0.0985
Fsparse 40%	0.8485	0.0720	**Fsparse 40%**	0.4053	0.3409
Fsparse 30%	0.7386	0.1819	**Fsparse 30%**	0.0265	0.7197
Fsparse 20%	0.2803	0.6402			
Llama-2-70B-Chat	**Accuracy**	**Degradation**	**Llama-2-7B-Chat**	**Accuracy**	**Degradation**
Full	0.4508		**Full**	0.1856	
Csparse 50%	0.3939	0.0569	**Csparse 50%**	0.1515	0.0341
Csparse 40%	0.3447	0.1061	**Csparse 40%**	0.1098	0.0758
Csparse 30%	0.2689	0.1819	**Csparse 30%**	0.0720	0.1136
Fsparse 50%	0.3864	0.0644	**Fsparse 50%**	0.1629	0.0227
Fsparse 40%	0.3902	0.0606	**Fsparse 40%**	0.1364	0.0492
Fsparse 30%	0.2689	0.1819	**Fsparse 30%**	0.1212	0.0644

Here we present more experiments for the comparison between Llama-2 and Llama-3 family models, which is first mentioned in Section B.3, where we also include FSparse methods together with the CSparse method. The results are in Table 11.

Figure 10: We look at the histogram of the number of errors versus the position among a period of sixteen tokens on average. We have two different datasets of Arithmetic Reasoning GSM-8K and AQuA-RAT-COT. We can see that the number of errors is distributed almost evenly for both datasets.



E Additional Results on Reasoning

Due to page restrictions, we only show GSM8K, CSQA, and HumanEval in the paper. Below we show additional results to the numbers presented in the paper. **We present tables of a similar format. Please notice that the leftmost column writes a number that represents the treewidth in the given settings.** Also, we show the results of SIRIUS on the other five datasets AQuA-RAT-COT (Arithmetic Reasoning), Sports (Commonsense Reasoning), Date (Commonsense Reasoning), and StrategyQA (CommonSense Reasoning), and MBPP+ (coding).

E.1 Arithmetic Reasoning

In this section, we present GSM8K and AQuA RAT COT evaluation results with the efficiency metric AAL. Sirius is shown to be effective on these two reasoning tasks about arithmetic. Below we show the raw AAL score associated with efficiency for all models and the performance of different treewidths.

E.2 CommonSense Reasoning

We followed the COT paper and evaluated Sirius on CSQA, Sports, StrategyQA, and Dates. Sparse methods are capable of outputting high-quality output similar to the full model at the 0.5 mark, which is different than on other datasets. However, we tune the sparsity level to 0.4 (0.6 dense, 0.4 removed), and it starts to have performance degradation. Sirius can compensate them with relatively high efficiency)

E.3 Code

We also have a coding portion that evaluates Sirius on HumanEval. Sirius performs well similar to other datasets. Besides, we also have results on MBPP+. The results show SIRIUS effectiveness and efficiency again.

Optionally include supplemental material (complete proofs, additional experiments and plots) in appendix. All such materials **SHOULD be included in the main submission.**

Table 12: Miscellaneous Results: 5 models on different Three Different datasets.

Experiment Setting	CoQA	AGIEval (Math)	MMLU-FLAN-COT
Llama-2-7B-Chat	0.5982/0.7580	0.072	0.4925
Llama-2-7B-Chat-FSparse	0.5898/0.7540	0.077	0.4768
Llama-2-7B-Chat-FSparse-SIRIUS	0.5908/0.7540	0.081	0.4670
Llama-2-7B-Chat-CSparse	0.6117/0.7639	0.065	0.4637
Llama-2-7B-Chat-CSparse-SIRIUS	0.6117/0.7664	0.078	0.4794
Llama-3-8B-Instruct	0.6153/0.7825	0.213	0.6231
Llama-3-8B-Instruct-FSparse	0.5828/0.7577	0.172	0.5304
Llama-3-8B-Instruct-FSparse-SIRIUS	0.5868/0.7591	0.196	0.5709
Llama-3-8B-Instruct-CSparse	0.6003/0.7735	0.154	0.5558
Llama-3-8B-Instruct-CSparse-SIRIUS	0.6005/0.7728	0.178	0.6003
Llama-2-13B-Chat	0.6408/0.7896	0.092	0.5317
Llama-2-13B-Chat-FSparse	0.6320/0.7837	0.087	0.5082
Llama-2-13B-Chat-FSparse-SIRIUS	0.6340/0.7859	0.089	0.5219
Llama-2-13B-Chat-CSparse	0.6350/0.7841	0.088	0.5127
Llama-2-13B-Chat-CSparse-SIRIUS	0.6363/0.7847	0.1	0.5127
Llama-2-7B	0.6388/0.7735	0.101	0.4520
Llama-2-7B-FSparse	0.6352/0.7697	0.09	0.4435
Llama-2-7B-FSparse-SIRIUS	0.6352/0.7697	0.092	0.4415
Llama-2-7B-CSparse	0.6338/0.7700	0.086	0.4213
Llama-2-7B-CSparse-SIRIUS	0.6372/0.7709	0.093	0.4317
Llama-3-8B	0.6727/0.8055	0.163	0.5754
Llama-3-8B-FSparse	0.6625/0.7984	0.152	0.5349
Llama-3-8B-FSparse-SIRIUS	0.6625/0.7984	0.154	0.5532
Llama-3-8B-CSparse	0.6633/0.7977	0.131	0.5049
Llama-3-8B-CSparse-SIRIUS	0.6670/0.7995	0.15	0.5428

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer

Table 13: SIRIUS Tree on GSM8K.

Experiment Settings treewidth	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.7536/0.7544	N/A	0.7536/0.7544	N/A
Sparse Performance	0.5868/0.5891	N/A	0.3844/0.3867	N/A
1	0.7316/0.7324	14.5903	0.6983/0.7005	13.1903
2	0.7172/0.7172	14.9554	0.7089/0.7096	14.1517
4	0.7278/0.7309	15.3705	0.7119/0.7111	14.8393
6	0.7195/0.7187	15.5979	0.7081/0.7074	15.0682
8	0.7202/0.7218	15.5548	0.7051/0.7058	15.2291
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.4966/0.5042	N/A	0.4966/0.5042	N/A
Sparse Performance	0.3199/0.3260	N/A	0.2085/0.2168	N/A
1	0.4526/0.4572	14.6946	0.439/0.445	13.361
2	0.4579/0.4640	15.0355	0.4299/0.4367	14.3061
4	0.4579/0.4540	15.0355	0.4223/0.4306	14.9721
6	0.4450/0.4503	15.4834	0.4177/0.4238	15.1435
8	0.4352/0.4428	15.5863	0.4177/0.4238	15.2939
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.3548/0.3647	N/A	0.3548/0.3647	N/A
Sparse Performance	0.3222/0.3275	N/A	0.2714/0.2767	N/A
1	0.3533/0.3472	15.085	0.3412/0.3472	14.0153
4	0.3412/0.3374	15.7576	0.3381/0.3412	15.3491
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.2282/0.2312	N/A	0.2282/0.2312	N/A
Sparse Performance	0.2191/0.2229	N/A	0.1759/0.1797	N/A
1	0.2328/0.2381	15.6759	0.2418/0.2472	15.3415
4	0.2372/0.2403	15.9283	0.2077/0.2100	15.825
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.2403/0.2426	N/A	0.2403/0.2426	N/A
Sparse Performance	0.1971/0.1994	N/A	0.1334/0.1372	N/A
1	0.2282/0.2312	14.8172	0.2214/0.2229	12.6888
2	0.2297/0.2305	15.1784	0.2252/0.2359	13.8875
4	0.2305/0.2282	15.5467	0.2183/0.2214	14.6751
6	0.2388/0.2411	15.691	0.2199/0.2206	14.8575
8	0.2312/0.2343	15.735	0.2244/0.2252	15.0017
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.1357/0.1403	N/A	0.1357/0.1403	N/A
Sparse Performance	0.1137/0.1168	N/A	0.0758/0.0804	N/A
1	0.1183/0.1205	15.6864	0.1152/0.1168	15.1096
2	0.1334/0.1357	15.7893	0.113/0.116	15.5358
4	0.1410/0.1448	15.9161	0.113/0.116	15.8558 (53.341)
6	0.1289/0.1312	15.9662	0.1183/0.1205	15.8715
8	0.1236/0.1259	15.9568	0.1114/0.1145	15.8939

Table 14: SIRIUS on AQuA-RAT-COT.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.515748	N/A	0.515748	N/A
Sparse Performance	0.42126	N/A	0.271654	N/A
1	0.429134	13.1945	0.468504	10.3322
4	0.488189	14.4722	0.44095	10.6228
6	0.496063	15.115	0.452756	10.8348
8	0.476378	15.3721	0.456693	11.0874
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.456693	N/A	0.456693	N/A
Sparse Performance	0.287402	N/A	0.228346	N/A
1	0.377953	12.6665	0.377945	12.6665
4	0.385827	14.46	0.397638	10.2826
6	0.366142	15.0671	0.370079	10.6753
8	0.42126	15.0995	0.38189	11.0019
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.232283	N/A	0.232283	N/A
Sparse Performance	0.251969	N/A	0.208661	N/A
1	0.275591	15.5163	0.26378	9.17465
4	0.279528	15.3995	0.259843	10.8726
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.149606	N/A	0.149606	N/A
Sparse Performance	0.165354	N/A	0.149606	N/A
1	0.185039	15.4652	0.161417	11.0874
4	0.220472	15.6346	0.192913	11.8238
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.251969	N/A	0.251969	N/A
Sparse Performance	0.283465	N/A	0.220472	N/A
1	0.248031	15.5294	0.251969	12.6424
4	0.259843	15.7254	0.244096	14.5794
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.15748	N/A	0.15748	N/A
Sparse Performance	0.153543	N/A	0.177165	N/A
1	0.185039	15.423	0.141732	15.4122
4	0.161417	15.651	0.145669	15.3788

Table 15: SIRIUS on CSQA.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.707341	N/A	0.707341	N/A
Sparse Performance	0.615889	N/A	0.647011	N/A
1	0.699427	12.2108	0.724816	11.0512
4	0.687961	13.2734	0.709255	13.5876
6	0.714169	13.7842	0.720721	13.3097
8	0.710893	14.1173	0.707617	14.76893
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.643735	N/A	0.643735	N/A
Sparse Performance	0.53317	N/A	0.558559	N/A
1	0.638821	15.0088	0.618346	12.7426
4	0.630631	14.6151	0.63964	14.8704
6	0.625717	14.9905	0.640459	15.1968
8	0.617527	15.2534	0.642916	15.4355
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.687961	N/A	0.687961	N/A
Sparse Performance	0.53317	N/A	0.553645	N/A
1	0.657658	13.0868	0.649468	9.2183
4	0.669124	14.309	0.669124	11.438
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.610975	N/A	0.610975	N/A
Sparse Performance	0.570025	N/A	0.560197	N/A
1	0.578215	15.2554	0.58231	14.7381
4	0.586405	15.7213	0.606061	15.7284
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.624898	N/A	0.624898	N/A
Sparse Performance	0.616708	N/A	0.520066	N/A
1	0.632269	14.1015	0.608518	11.4607
4	0.638002	15.0978	0.605242	14.0366
6	0.605242	15.4365	0.611794	14.7197
8	0.621622	15.552	0.617527	15.0799
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.474201	N/A	0.474201	N/A
Sparse Performance	0.471744	N/A	0.441441	N/A
1	0.488124	15.5119	0.461916	14.703
4	0.494676	15.9141	0.486486	15.5972
6	0.501229	15.8922	0.476658	15.7315
8	0.473382	15.9247	0.474201	15.802

Table 16: SIRIUS on Sports.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.943299	N/A	0.943299	N/A
Sparse Performance	0.864948	N/A	0.879381	N/A
1	0.937113	12.3652	0.946392	9.95237
4	0.941237	14.5248	0.943299	11.5858
6	0.942268	14.8651	0.943299	14.0954
8	0.939175	14.9832	0.941237	14.7718
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.898969	N/A	0.898969	N/A
Sparse Performance	0.748454	N/A	0.720619	N/A
1	0.86653	15.5259	0.845361	13.5897
4	0.849485	15.5917	0.847423	15.2325
6	0.863918	15.5256	0.843299	15.4376
8	0.869072	15.6014	0.841237	15.5023
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.742268	N/A	0.742268	N/A
Sparse Performance	0.690722	N/A	0.584536	N/A
1	0.710309	13.9767	0.717659	7.72686
4	0.735052	14.9247	0.728953	10.7298
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.709278	N/A	0.709278	N/A
Sparse Performance	0.635052	N/A	0.558763	N/A
1	0.669072	15.4924	0.639175	14.3603
4	0.657732	15.9845	0.658763	15.48
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.731959	N/A	0.731959	N/A
Sparse Performance	0.652677	N/A	0.596907	N/A
1	0.704124	14.3861	0.712371	11.1517
4	0.712371	15.5904	0.71134	13.7394
6	0.709278	15.7475	0.71134	13.9857
8	0.698969	15.9927	0.715464	14.3817
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.545361	N/A	0.545361	N/A
Sparse Performance	0.536082	N/A	0.528866	N/A
1	0.524742	15.6754	0.536082	14.1031
4	0.547423	15.937	0.538144	15.6263
6	0.545361	15.9807	0.540206	15.7243
8	0.545361	15.9927	0.549485	15.811

Table 17: SIRIUS on Date.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.657224	N/A	0.657224	N/A
Sparse Performance	0.518414	N/A	0.532578	N/A
1	0.688385	14.2885	0.671388	14.6771
4	0.671388	15.357	0.685552	15.6324
6	0.679887	15.2435	0.688385	15.1663
8	0.674221	15.2654	0.694051	15.4293
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.583569	N/A	0.583569	N/A
Sparse Performance	0.399433	N/A	0.424929	N/A
1	0.535014	15.4236	0.535411	14.4364
4	0.543909	15.4782	0.546742	15.606
6	0.546742	15.6365	0.526912	15.7718
8	0.549575	15.7159	0.541076	15.7997
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.524079	N/A	0.524079	N/A
Sparse Performance	0.498584	N/A	0.419263	N/A
1	0.490085	13.9589	0.461756	14.1419
4	0.524079	15.432	0.478992	15.8545
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.501416	N/A	0.501416	N/A
Sparse Performance	0.464589	N/A	0.390935	N/A
1	0.447592	15.5992	0.461756	15.3896
4	0.492918	15.9129	0.484419	15.8357
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.320113	N/A	0.320113	N/A
Sparse Performance	0.339943	N/A	0.3002823	N/A
1	0.31728	14.4663	0.305949	5.75938
4	0.345609	15.6588	0.325779	14.7519
6	0.342776	15.742	0.314448	14.5768
8	0.348442	15.7692	0.308782	14.3627
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.33711	N/A	0.33711	N/A
Sparse Performance	0.314448	N/A	0.235127	N/A
1	0.342776	15.5144	0.269122	15.3598
4	0.342776	15.9141	0.266289	15.8553
6	0.328612	15.943	0.274788	15.9266
8	0.322946	15.9671	0.271955	15.956

Table 18: SIRIUS on StrategyQA.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 10)**
Original Performance	0.770241	N/A	0.770241	N/A
Sparse Performance	0.713348	N/A	0.562363	N/A
1	0.741794	8.98893	0.737418	6.60992
4	0.741794	9.48412	0.746171	7.92521
6	0.743982	9.53292	0.728665	8.22667
8	0.743982	9.55946	0.708972	8.97268
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.649891	N/A	0.649891	N/A
Sparse Performance	0.599562	N/A	0.439825	N/A
1	0.623632	9.46018	0.531729	8.68633
4	0.623632	9.74383	0.560175	9.44497
6	0.632385	9.83975	0.560175	9.58122
8	0.680525	9.80493	0.555799	9.67198
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.695842	N/A	0.695842	N/A
Sparse Performance	0.706783	N/A	0.634573	N/A
1	0.71116	9.48266	0.682713	6.74106
4	0.667396	9.83767	0.715536	8.09959
6	0.671772	9.88989	0.693654	8.82263
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.63895	N/A	0.63895	N/A
Sparse Performance	0.693654	N/A	0.533917	N/A
1	0.695842	9.8979	0.595186	8.77388
4	0.682713	9.96438	0.643326	9.47368
6	0.689278	9.9789	0.63895	9.56319
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.654267	N/A	0.654267	N/A
Sparse Performance	0.678337	N/A	0.612691	N/A
1	0.684902	9.64754	0.669584	6.55818
4	0.691466	9.79539	0.671772	7.88988
6	0.68709	9.86474	0.643326	8.28982
8	0.689278	9.86488	0.66302	8.43513
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.599562	N/A	0.599562	N/A
Sparse Performance	0.592998	N/A	0.538293	N/A
1	0.612691	9.73256	0.568928	8.38473
4	0.599562	9.93662	0.560175	9.36272
6	0.617068	9.95582	0.536105	9.35857
8	0.610503	9.96658	0.544858	9.4642

Table 19: SIRIUS on HumanEval.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.560975609756098	N/A	0.560975609756098	N/A
Sparse Performance	0.457317073170732	N/A	0.207317073170732	N/A
1	0.585365853658537	14.7624	0.554878048780488	12.1326
4	0.579268292682927	15.2299	0.530487804878049	14.0546
6	0.615853658536585	15.4209	0.518292682926829	14.4431
8	0.585365853658537	15.5009	0.524390243902439	14.6725
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.26219512195122	N/A	0.26219512195122	N/A
Sparse Performance	0.189024390243902	N/A	0.0670731707317073	N/A
1	0.231707317073171	15.1878	0.109756097560976	12.1402
4	0.274390243902439	15.2827	0.219512195121951	13.7718
6	0.26219512195122	14.5355	0.207317073170732	14.7776
8	0.29268	15.6305	0.24390243902439	15.1074
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.189024390243902	N/A	0.189024390243902	N/A
Sparse Performance	0.146341463414634	N/A	0.121951219512195	N/A
1	0.170731707317073	14.3976	0.189024390243902	9.6447
4	0.182926829268293	15.1956	0.176829268292683	10.7946
6	0.182926829268293	15.3494	0.170731707317073	11.0149
8	0.176829268292683	15.4067	0.170731707317073	11.1252
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.176829268292683	N/A	0.176829268292683	N/A
Sparse Performance	0.158536585365854	N/A	0.0975609756097561	N/A
1	0.146341463414634	15.2129	N/A	N/A
4	0.158536585365854	15.9093	0.146341463414634	14.1813
6	0.170731707317073	15.9211	0.134146341463415	14.5866
8	0.176829268292683	15.9015	0.134146341463415	14.7508
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.140243902439024	N/A	0.140243902439024	N/A
Sparse Performance	0.134146341463415	N/A	0.0670731707317073	N/A
1	0.134146341463415	14.055	0.140243902439024	8.83176
4	0.146341463414634	14.8504	0.146341463414634	10.1263
6	0.152439024390244	15.1924	0.152439024390244	10.5576
8	0.164634146341463	15.2742	0.158536585365854	10.822
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.115853658536585	N/A	0.115853658536585	N/A
Sparse Performance	0.115853658536585	N/A	0.0792682926829268	N/A
1	0.115853658536585	15.5268	0.121951219512195	12.6604
4	0.128048780487805	15.8167	0.121951219512195	14.4053
6	0.164634146341463	15.8615	0.121951219512195	14.8296
8	0.109756097560976	15.9189	0.128048780487805	14.8443

Table 20: SIRIUS on MBPP+.

Experiment Settings	**Llama-3-8B-Instruct-FSparse**		**Llama-3-8B-Instruct-CSparse**	
treewidth	**Performance**	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.584656084656085	N/A	0.584656084656085	N/A
Sparse Performance	0.531746031746032	N/A	0.248677248677249	N/A
1	0.537037037037037	14.7267	0.563492063492064	11.5415
4	0.563492063492064	15.2699	0.566137566137566	13.5896
6	0.552910052910053	15.3782	0.571428571428571	14.0547
8	0.552910052910053	15.4689	0.566137566137566	14.7648
	Llama-3-8B-FSparse		**Llama-3-8B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.518518518518519	N/A	0.518518518518519	N/A
Sparse Performance	0.433862433862434	N/A	0.161375661375661	N/A
1	0.4894	14.8849	0.415343915343915	12.7016
4	0.484126984126984	15.4346	0.407407407407407	14.0936
6	0.473544973544974	15.3581	0.433862433862434	14.5662
8	0.468253968253968	15.6088	0.41005291005291	14.4752
	Llama-2-13B-Chat-FSparse		**Llama-2-13B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.23015873015873	N/A	0.27	N/A
Sparse Performance	0.19047619047619	N/A	0.1	N/A
1	0.201058201058201	13.8235	0.26	9.32827
4	0.232804232804233	14.8394	0.26	10.7346
6	0.224867724867725	15.0801	0.26	10.8897
8	0.227513227513228	15.2373	0.25	11.0214
	Llama-2-13B-FSparse		**Llama-2-13B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 16)**
Original Performance	0.246031746031746	N/A	0.21	N/A
Sparse Performance	0.232804232804233	N/A	0.13	N/A
1	0.214285714285714	14.8374	0.22	14.5174
4	0.259259259259259	15.8547	0.24	15.6461
6	0.235449735449735	15.9197	0.23	15.7174
8	0.246031746031746	15.9094	0.22	15.7179
	Llama-2-7B-Chat-FSparse		**Llama-2-7B-Chat-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.261904761904762	N/A	0.261904761904762	N/A
Sparse Performance	0.224867724867725	N/A	0.100529100529101	N/A
1	0.238095238095238	14.1571	0.214285714285714	8.61325
4	0.26984126984127	14.9264	0.23015873015873	10.2517
6	0.238095238095238	15.2194	0.227513227513228	10.5845
8	0.272486772486773	15.3086	0.235449735449735	10.7621
10	N/A	N/A	0.232804232804233	10.8962
	Llama-2-7B-FSparse		**Llama-2-7B-CSparse**	
	Performance	**AAL (out of 16)**	**Performance**	**AAL (out of 12)**
Original Performance	0.253968253968254	N/A	0.253968253968254	N/A
Sparse Performance	0.201058201058201	N/A	0.0793650793650794	N/A
1	0.216931216931217	14.6103	0.171957671957672	10.6643
4	0.238095238095238	15.5672	0.185185185185185	11.5561
6	0.224867724867725	15.6273	0.195767195767196	11.6547
8	0.240740740740741	15.5569	0.203703703703704	11.6753

"[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

A. **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

B. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [No]

Justification: **[TODO]**

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

C. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

D. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  1. If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  2. If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  3. If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  4. We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

E. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification:

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

F. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

G. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).

- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

H. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

I. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification:

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

J. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

K. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

L. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

M. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

N. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

O. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.