APCE: Adaptive Progressive Context Expansion for Long Context Processing

Baisub Lee, Sanghyun Byun, Mohanad Odema,
Jung Guack, Jacob Song, Woo Seong Chung
LG Electronics USA
{baisub.lee, sang.byun, mohanad.odema}@lge.com
{jung.guack, jaigak.song, wooseong.chung}@lge.com

Abstract

Deploying useful Long-Context Transformer Models (LCTMs) requires addressing two key challenges: (1) A *growing memory footprint* due to quadratic self-attention and linear KV-cache scaling in memory as sequence length increases; (2) the *ContextRot* phenomena where empirical evidence suggests that transformer architecture's performance degrades with increasing context length. Given the shared dependency on the input, a natural question arises: *"Can we surgically select the most important input chunks for processing to synergistically (a) reduce the memory footprint, and (b) mitigate the ContextRot effects?"* In this paper, we answer this question in the affirmative for *long-context summarization* tasks. We propose APCE as a context-aware solution to select the most important input chunks through low-dimensional semantic similarity matching with the current query. By directly operating on the input, APCE decouples from strict dependency on underlying hardware or CUDA environments, promising a compatible solution scalable to different deployment systems. Our empirical evaluations have demonstrated superior or on-par summarization performance for APCE compared to the full dense baseline using a fraction (50%-70%) of the input sequence resulting in KV-cache and self-attention memory efficiency improvements. We hope our findings inspire further research on context-aware efficiency solutions for LCTMs geared towards other relevant long-context tasks.

1 Introduction

Long-Context Transformer Models (LCTMs) are becoming increasingly popular given their ability to work with long sequences (2k - 1M context length) to enable long context tasks such as long-document summarization, reasoning, and multi-modal understanding [22, 4, 10, 1, 12]. In order to deploy useful LCTMs, two things to consider: (1) the *growing memory footprint* of the model stemming from the quadratic memory complexity of self-attention operation, as well as the linear memory requirement for storing the KV-cache; (2) Addressing the *ContextRot* [8] phenomena, where empirical evidence has suggested that the transformer architecture struggles to maintain performance as context length increases – especially challenging with current the scales (2k - 1M context length).

Sparsification and KV cache compression methods have been heavily studied to address these challenges (See Appendix A). However, two aspects remain relatively underexplored: (1) Most sparsification methods focus their innovations on the attention operation, leveraging low-dimensional importance calculations for the *Key* and *Value* tokens/blocks to drive the sparsification decisions. Though effective, the caveat remains that the *Key* and *Value* blocks from every input token/block in every self-attention operation would require loading into the lower memory levels at least once for the low-dimensional projection and importance computations, necessitating specialized kernel

ML For Systems workshop at Neural Information Processing Systems (NeurIPS 2025).

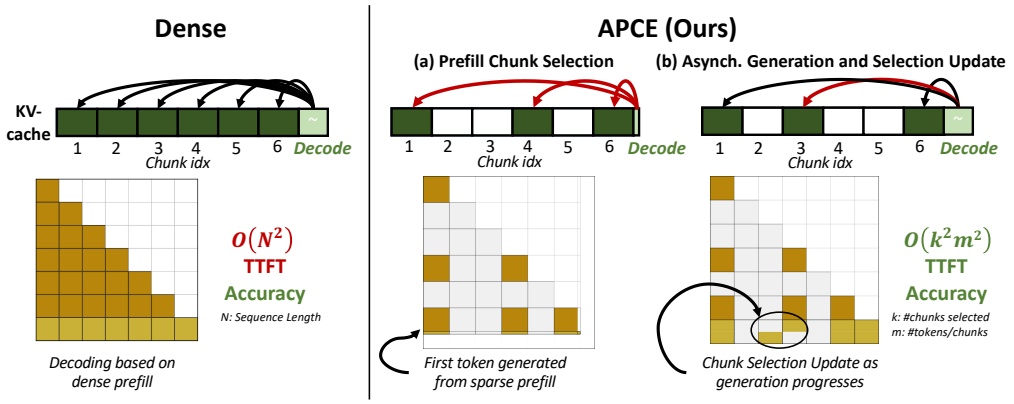

Figure 1: Overview of APCE for an example with 6 chunks. (*left*) The full dense baseline attention suffers from quadratic complexity and long time-to-first-token (TTFT). (*middle*) APCE performs prefill chunk selection using semantic similarity, resulting in sparse attention reducing complexity and TTFT. (*right*) Chunk selection is updated as generation progresses to maintain performance.

support to compensate for the added operations and data movement overheads; (2) Eviction strategies for KV cache compression are *persistent*, lacking the means to reconsider evicted tokens/blocks even if they become more relevant as the context evolves [17, 21]. This constitutes a common situation for long-context scenarios (e.g., spread out dependencies across multiple passages in long-documents).

We present APCE (Adaptive Progressive Context Expansion) as a complementary method to compensate for these two points by directing sparsification efforts towards the input chunks themselves while allowing Reprioritization to update chunks selection. Through a low-dimensional semantic similarity matching approach, APCE can determine which input chunks are most important in relation to the current query, choosing only those to proceed with the downstream transformer operations. Since APCE operates on the input, the low-dimensional representations of the input chunks are only computed once during prefill, and kept in memory for quick similarity matching operations enabling chunk selection update as the context evolves. We summarize the paper's contributions as follows:

- We propose APCE, a context-aware input chunk sparsification solution leveraging semantic similarity matching and supporting chunk replacement for efficient long-context processing.
- We evaluate APCE on long-context summarization task, demonstrating similar or superior performance to that of the full dense baseline using a fraction (50-70%) of input chunks.
- We perform ablation studies on APCE's Reprioritization to characterize its performance-efficiency trade-offs, and provide analytical analysis on APCE's memory efficiency gains.
- We discuss future research potential for scaling APCE to further related long-context tasks.

2 Method

Base Formulation. Denote $s = [x_1, x_2, \cdots, x_N]$ as a sequence of N input tokens to be processed where $x_i \in \mathbb{N}$ represents the ith token. For long context, this token sequence can be partitioned into chunks such that the ith chunk is defined as $x^{(c_i)} = [x_1^{(c_i)}, x_2^{(c_i)}, \cdots, x_{m_i}^{(c_i)}]$, with entry $x_j^{(c_i)} \in \mathbb{N}$ represents the jth token belonging to a chunk i of size m_i.

Let $f(\cdot)$ be a projection function for mapping token sequences into low sequence embeddings where $\mathbf{c_i} = f(x^{(c_i)})$, $\mathbf{c_i} \in \mathbb{R}^d$ represents a d-dimensional embedding representation of the ith chunk. Given a sequence of n chunks, denote $\mathbf{C} = [\mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_n}]$ as their corresponding sequence of d-dimensional embedding representations. Define also the query token sequence containing an instruction or question as $x^{(q)} = [x_1^{(q)}, x_2^{(q)}, \cdots, x_k^{(q)}]$. From it, we also obtain $\mathbf{q} = f(x^{(q)}), \mathbf{q} \in \mathbb{R}^d$ as the d-dimensional query embedding. After chunked prefill, APCE targets identifying the indices for the *top*-k most relevant chunks maximizing a similarity function:

$$\mathcal{I}_{\text{top-}k} = \underset{i \in \{1,\ldots,n\}}{\arg\text{top}}\ k\ \text{Sim}(\mathbf{q}, \mathbf{c}_i)$$

Table 1: Aggregate Statistics for BookSum [10] and timings comparisons for APCE on Llama-3.2-3B-Instruct [7] across all context lengths when varying the **maximum number of chunks selected**. Higlighted cells are ones where APCE achieves the best numbers.

Method	Performance ↑		Timings ↓	
	BERTScore (F1)	ROUGE-L (F1)	TTFT (s)	Total time (s)
Dense Baseline	0.8413±0.0005	0.1591±0.0082	4.6820±2.9131	14.8356±6.7588
APCE (50% chunks used)	0.8412±0.0014	0.1521±0.0028	2.8161±1.7641	16.4040±7.6996
APCE (60% chunks used)	0.8400±0.0027	0.1567±0.0096	3.3041±1.9478s	17.9446±7.9049
APCE (70% chunks used)	0.8418±0.0010	0.1584±0.0078	3.6693±2.1177	19.6179±8.6174

where $\mathrm{Sim}(\mathbf{q}, \mathbf{c}_i)$ represent the similarity score between the current query and the ith chunk. The top-k selected chunks progress for processing and self-attention computation as shown in Figure 1. Throughout generation, $\mathbf{C}$ embeddings remain cached, while $\mathbf{q}$ is continously cached and updated.

Similarity Function. To capture semantic similarity between embeddings, *Semantic Scoring* via cosine similarity can be adopted similar to BERTScore [23]:

$$\mathrm{Sim}(\mathbf{q}, \mathbf{c}_i) = \frac{\mathbf{q}^\top \mathbf{c}_i}{\|\mathbf{q}\|_2 \, \|\mathbf{c}_i\|_2}$$

Reprioritization. To maintain relevancy of selected chunks with regards to the current query, APCE supports reprioritization to re-evaluate chunk similarity scores with current embedding and update the chunk selection accordingly. Reprioritization operates with the following two features:

- *Buffer Management*. Based on the max chunk selection size, lowest scoring chunk representations in the buffer can be evicted to make way for the new, higher scoring chunks.
- *Recomputation*. We support recomputing the K and V values for any chunks in the buffer if their attention scores depend on a newly selected chunk(s). This presents a trade-off between mitigating attention inconsistencies due to out-of-order chunk loading, and the added computational overhead. The ablation study in Appendix B.3 touches on this trade-off.

Enhanced Query Embedding. As decoding progresses, APCE dynamically updates the query representation by blending in context from both the original instruction and recently generated tokens. The enhanced query is updated every Reprioritization Interval.

Asynchronous Generation. We extend APCE to support asynchronous token generation to improve Time-to-First-Token (TTFT) in cases where constrained system resources (I/O, memory bandwidth) enforce a slow prefilling. APCE can initiate token generation from partially loaded chunks, progressively improving output generation quality as more context is acquired with more loaded chunks.

Complexity analysis. For a sequence length of N tokens, k selected chunks, and m-token chunk size, APCE reduces attention complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(k^2m^2)$ especially viable when $km \ll N$.

3 Experiments

We evaluate the performance of APCE on long-context summarization task. We provide a summarized version of the results here and refer the reader to Appendix B for full experiments and setup details.

Experimental Details. Our evaluations are conducted using *BookSum* [10], a large collection of datasets for long-form narrative summarization. We specify 3 context length groups (8k, 20k, and 30k), and select representative test samples from each groups to conduct our APCE evaluations at different context lengths. We use a Llama-3.2-3B-Instruct [7] as our base model, comparing APCE's performance against the baseline Full Dense model using an NVIDIA RTX 4090 with 24 GB memory.

Experiments Scope. Our experimental analysis for APCE is performed by evaluating summarization performance, latency timings, and memory efficiency. We vary two hyperparameters in our

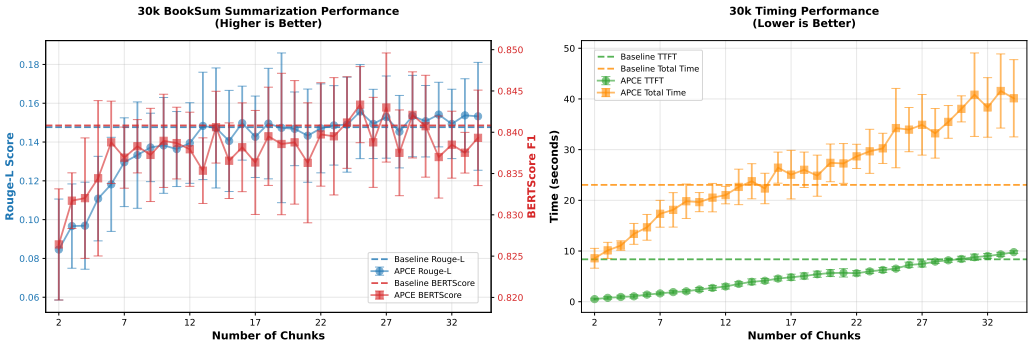

Figure 2: Summarization Performance scores (*left*) and inference timings (*right*) for APCE compared to the baseline when varying the number of loaded chunks on Llama-3.2-3B-Instruct [7] the 30K context length group from BookSum [10]. Chunk size is fixed to 800 tokens.

experiments: (1) Maximum #chunks selected given a fixed chunk size; (2) Chunk size given a fixed maximum #chunks selected. We also perform ablations on the repriortization interval.

3.1 Results Summary

Number of Selected Chunks Experiment. Table 1 shows aggregate summarization results across all context length groups when varying the number of chunks selected given 800 token chunk size. The Table shows APCE scores when 50%, 60%, and 70% of input chunks are selected at a time. Generally, we observe that APCE is capable of maintaining comparable scores with the dense baseline across the BERTScore [23] and ROUGE-L [13] metrics. APCE with 70% selection outperforms the baseline on BERTScore while being within $\sim$0.5% score margin on ROUGE-L score. Figure 2 (left) shows APCE starts to exhibit on-par, occasionally superior, performance to the baseline using $\sim 50\%$ of the input chunks for the 30k group. Figure 3 shows the full results across all groups.

Chunk Size Experiment. Table 2 shows the aggregated results when varying the chunk size given 70% of available chunks are selected. The results show 800 token chunk size can lead to on-par performance with the dense baseline, further corroborating the importance of choosing the right input chunks. Figure 4 illustrates the full results breakdown across the different context lengths.

Reprioritization and Memory Analysis. Appendix B.3 introduces Reprioritization ablation study showing the performance-efficiency trade-off affecting total generation time. Appendix C provides an analytical analysis on APCE's memory efficiency gains – reaching 55.6% for prefill attention.

4 Conclusion and Future Directions

We presented APCE, an input chunk sparsification solution for LCTMs improving their performance and memory efficiency on long-context summarization tasks. We highlight the following takeaways:

Beyond Summarization. APCE has demonstrated promising results on long-context summarization, paving the way to investigate its applicability in other long-context task scenarios (long-context reasoning and multi-turn dialogues). Considering their task-specific challenges, we hypothesize another level of sophistication is needed to preserve semantic understanding as input is sparsified.

APCE is a complementary Solution. Given its focus on the input chunks directly, APCE is compatible with existing SOTA solutions that adopt sparsification in the self-attention operation, enabling delivering performance and efficiency improvements from two dimensions.

On the implementation side. Our vanilla APCE solution demonstrated promising results. More sophisticated implementations can consider context-boundary awareness for semantic similarity matching and supporting compute kernels for efficiency.

Navigating Performance-Efficiency Trade-offs. Reprioritization Interval can be adjusted to meet the target use-case requirements with regards to performance and total generation time. Whereas the reduced TTFT can benefit use cases where response time is critical (e.g., low-risk robotic interactions).

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. URL `https://arxiv.org/abs/2204.14198`.

[2] Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*, 2024.

[3] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[6] Google DeepMind. Gemini 2.5: Our most intelligent ai model. `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`, 2025.

[7] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal

Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3

herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[8] Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://research.trychroma.com/context-rot.

[9] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Minference 1.0: accelerating pre-filling for long-context llms via dynamic sparse attention. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, 2025.

[10] Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*, 2021.

[11] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

[12] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[13] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[14] Matan Oren, Matan Hassid, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024. URL https://arxiv.org/abs/2401.06104.

[15] Sentence Transformers. all-MiniLM-L6-v2: Sentence embedding model. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, 2023–2024.

[16] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[17] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

[18] Guangtao Wang, Shubhangi Upasani, Chen Wu, Darshan Gandhi, Jonathan Li, Changran Hu, Bo Li, and Urmish Thakker. Llms know what to drop: Self-attention guided kv cache eviction for efficient long-context inference. *arXiv preprint arXiv:2503.08879*, 2025.

[19] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE, 2021.

[20] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[21] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

[22] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. 33:17283–17297, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[23] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. 36:34661–34710, 2023.

A Related Works

Long Context models are becoming increasingly popular to accommodate emerging multi-modal tasks [1], long context reasoning, and summarization [3]. Recent SOTA LLM models like Llama [7] support Rotary Position Embeddings (RoPE) [16] to scale their context window up to 128K or 1M tokens [6]. As the sequence length increases, the complexity of attention operation also scales quadratically, posing efficiency challenges for LLM inference. Thus, recent works propose sparse attention techniques to address these challenges [4, 22, 21, 17, 19, 9, 11, 2, 24, 20, 14, 18].

Initially, sparsification works adopted static predefined sparse attention patterns [4, 22] that reduce the attention computation requirements. However, the inflexibility of static patterns has led recent works to shift towards dynamic compute sparsity where irrelevant attention computations are skipped at runtime through masking/blocking techniques [21, 17, 19, 9, 11, 2]. For example, X-Attention [21] use anti-diagonal scoring to identify irrelevant attention blocks, and prune them from the attention structure to improve the computational footprint, whereas Quest [17] relies on similarity matching between current query vector and vector-wise statistics of each KV page. Notably, the implementation of these methods has been CUDA-centric, where their efficiency is dependent on hardware-specific custom kernel (e.g., FlashAttention [5]) that can implement the masking/blocking for sparse attention.

Another body of works targeted KV cache management strategies [24, 20, 14, 18] where low-scoring tokens are evicted from the cache to improve attention efficiency. For instance, H20 [24] retains a mixture of recent tokens and those with the highest cumulative attention scores (heavy hitters); whereas StreamingLLMs [20] leverages sink tokens and a sliding window to maintain most relevant tokens. These methods still rely on loading the full KV-cache initially in memory for scoring prior to the application of any eviction, and once a token is discarded it cannot be recovered even when its relevance increases as generation evolves (as was remarked in [17]).

The differences in APCE's approach are twofold: (1) it adopts sparsification in the input stage, enabling it to accelerate both the prefilling and decoding through semantic similarity without the need for computing initial attention scores or statistics for token eviction decisions; (2) It supports chunks replacement to maintain performance as the long context continues to evolve.

B Detailed Experiments

B.1 Experimental Setup

Long-Context Summarization Dataset. Our long-context summarization experiments are conducted on *BookSum* [10]. *BookSum* is a large collection of datasets for long-form narrative summarization containing a variety of long-context inputs (e.g., books, chapters, paragraphs). To cover different ranges of context lengths, we first define three input lengths groups with base lengths of 8k, 20k, and 30k. Then, we assign to each group the test samples from *BookSum* that are within 10% margin of the corresponding base input length. From each group, we select the first 10 input samples for our final evaluations. We remark that in the 30k group, there were only 6 test samples in the group and thus were all considered in the evaluations.

Evaluation Metrics We use the following metrics for our summarization evaluations:

- **ROUGE-L** [13]: Evaluates longest common subsequence overlap, emphasizing sentence-level structural similarity.
- **BERTScore** [23]: Measures semantic similarity using contextual embeddings, providing more nuanced evaluation of meaning preservation.

Baselines, Model, and Hardware. We evaluate APCE and compare against a Full-dense baseline that uses all the input token sequence for its attention computation and downstream processing. We implement both solutions on a Llama-3.2-3B-Instruct [7]. We perform our experiments by default on an NVIDIA RTX 4090 with 24 GB memory.

Experiments Description. Our main evaluations on APCE involve varying two hyperparameters: (1) The #chunks selected given a fixed chunk size; (2) The chunk size given a fixed #chunks selected. When varying the #chunks selected, we fix the chunk size at 800 tokens. When varying the chunk size, we set the #chunks selected to the equivalent number around 70% of sequence length (7, 18,

Table 2: Aggregate Statistics for BookSum [10] performance and timings comparisons for APCE on Llama-3.2-3B-Instruct [7] across all context lengths when varying the **Chunk Size**. Higlighted cells are ones where APCE achieves the best numbers.

Method	Performance ↑		Timings ↓	
	BERTScore (F1)	ROUGE-L (F1)	TTFT (s)	Total time (s)
Dense Baseline	0.8395±0.0016	0.1610±0.0032	5.1497±3.1950	16.0875±8.0018
APCE (800 token chksize)	0.8419±0.0026	0.1606±0.0098	3.9516±2.5660	19.0695±10.2352
APCE (1000 token chksize)	0.8398±0.0015	0.1548±0.0095	4.7861±2.9681	20.1979±9.5202
APCE (1600 token chksize)	0.8295±0.0158	0.1472±0.0219	6.3236±3.8602	27.3208±16.9383

and 24 for respective 8k, 20k, and 30k context-length groups). We also perform an ablation study on the effects of Reprioritization Interval, and conduct an analytical memory efficiency analysis for APCE compared to the Full Dense.

APCE Configuration. By default, we set for APCE the Reprioritization Interval to 50 tokens with Recomputation enabled. We set asynchronous generation to start after loading the 4th chunk. For the enhanced query embedding during generation, we project the last 100 characters in the question/instruction, and blend them with the last 50 generation tokens.

Chunks' Embedding Projection. We project each chunk into a low-dimensional embedding of size 384. We use SentenceTransformer [15] for our projection. The memory overhead is minimal compared to other components since the projections are stored only once to used across all self-attention layers. For instance, projections of 37 chunks of size 800 tokens (equivalent to the 30k context length) result in ∼28 MB memory footprint in FP16 format.

B.2 BookSum Summarization Experiments

We evaluate the long-context summarization performance of APCE and the Full Dense baseline on BookSum [10] across the 3 context length groups (8k, 20k, 30k). Tables 1 and 2 provide an aggregated results in terms of summarization performance and timing evaluations across all context lengths covering the two respective experimental settings: (a) varying the number of chunks given a fixed chunk size (800 tokens); (b) varying the chunk size given a fixed number of chunks (70% of total context length). The full results are provided in Figures 3 and 4.

Number of Chunks Variation. As observed in Figure 3, APCE is capable of maintaining comparable scores with the dense baseline across the BERTScore [23] and ROUGE-L [13] metrics across all context length groups starting from low number of chunks (around ∼30% of the context length). At APCE with 70% chunk usage, we find that APCE outperforms the baseline on BERTScore while being within ∼0.5% score margin on ROUGE-L score (see Table 1). As for latency, our vanilla APCE implementation sustains longer generation times than the dense baseline when ∼50% of chunks are selected. Key reasons include the added overhead of Reprioritization, which can be managed through (1) adjusting the Reprioritization Interval as shown in the ablation in Appendix B.3. (2) Exploring more efficient APCE implementation with a supporting, customized self-attention operator (Future research point). As for TTFT, asynchronous generation leads APCE to achieve speedups over the baseline, which can for its own part be useful for a special subset of applications (low-risk robotics).

Chunk Size Variation. Figure 4 shows that across the 3 context length groups, chunk size around 800-1000 token provide the best Rouge-L and BERTScore performance, higlighting the importance of semantic-based selection to improve performance. We also observe performance drop spikes (Figure 4 top-right sub-figure at the 30k context length), which can generally attribute to one of two reasons: (1) *ContextRot* as discussed in prior sections; (2) The chunking strategy resulted in a split that disrupted a coherent sequence, causing loss of knowledge. The latter point opens the door for further investigation on context-aware chunking strategies. Table 2 provides an aggregated summary of the results across all context length groups for the 800, 1000, and 1600 chunk sizes. The timing plots exhibit similar patterns to the previous experiment and thus do not alter our conclusions.

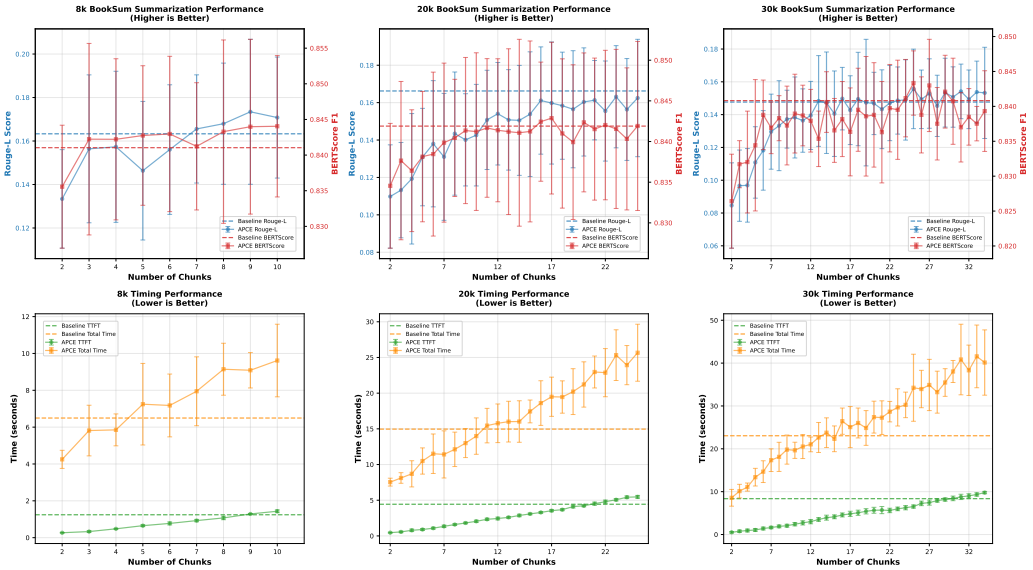

Figure 3: Comparing APCE and the Full Dense baseline across the 3 groups of long context when **varying the number of chunks at fixed chunk size of 800**. The top row shows the performance results on BookSum [10], whereas the bottom row shows the timing evaluations.

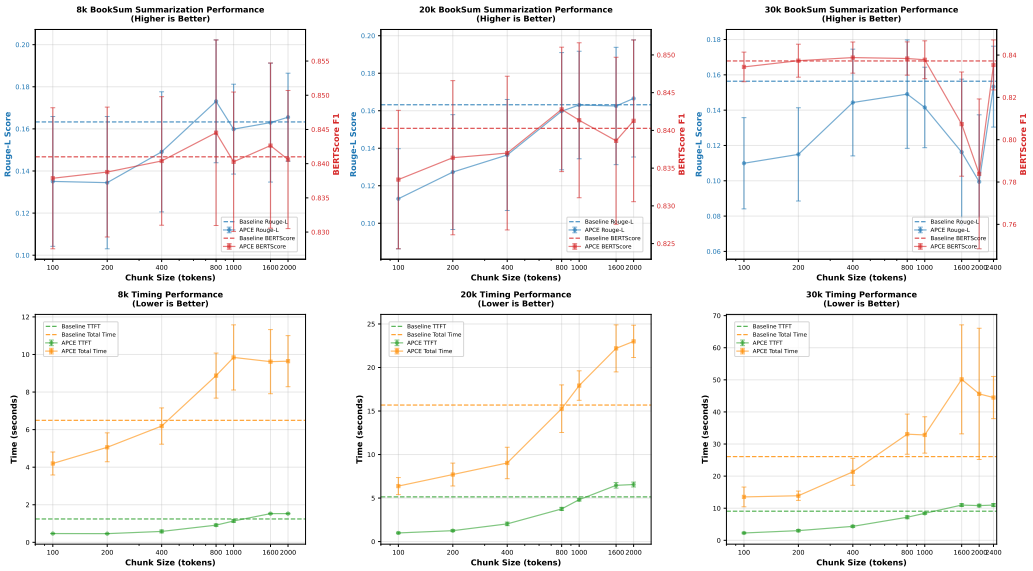

Figure 4: Comparing APCE and the Full Dense baseline across the 3 groups of long context when **varying the chunk size at 70% of available chunks selected**. The top row shows the performance results on BookSum [10], whereas the bottom row shows the timing evaluations.

Table 3: Ablation on Reprioritization Interval showing the average results for 10 book samples at 10k context length at **256 token chunk size**. Rep. Interval at 200 is closest to No Replacement case. Highlighted numbers are the ones discussed in the text.

Rep. Intvl (tokens)	**Rouge-L** (F1)	**BERTScore** (F1)	**TTFT** (s)	**Total Time** (s)	**Replace. taken**	**Replace. Available**
1	0.1237±0.0282	0.8234±0.0160	0.861±0.02	11.551±4.97	20.1±8.0	198.5±123.6
5	0.1435±0.0179	0.8182±0.0255	0.552±0.07	11.409±4.78	13.0±4.6	50.6±27.7
10	0.1364±0.0357	0.8186±0.0233	0.542±0.07	9.309±3.76	9.2±2.2	20.4±12.2
25	0.1360±0.0294	0.8331±0.0110	0.549±0.07	7.802±1.09	4.7±1.8	6.5±1.1
50	0.1219±0.0249	0.8316±0.0094	0.549±0.07	6.640±0.51	1.9±0.7	2.4±0.5
100	0.1355±0.0264	0.8200±0.0300	0.543±0.07	8.608±2.55	1.2±0.6	1.8±1.0
200	0.1246±0.0314	0.8211±0.0333	0.543±0.07	7.431±2.03	0.2±0.4	0.4±0.7
Dense Baseline	0.1270±0.0147	0.8240±0.0123	1.731±0.11	6.360±0.83	-	-

B.3 Ablation Study on Reprioritzation

We perform an ablation study on the Reprioritization interval using 10 books of 10k context length. To better observe the Reprioritization effects, we alter the configuration setting token chunk size to 256 tokens and the maximum number of chunks to 16. We show the results in Table 3, where the average total output generation length is 198.87±36.67 tokens. The results displayed in the table are the average across all test cases.

The first observation is that looking at the extreme replacement case (every 1 token), only 10.07% of replacement opportunities are used.This suggests that chunk replacements around $\sim$10% of the total output generation length (200-250 tokens in this experiment) are sufficient to maintain performance at levels comparable to the baseline. If we test this hypothesis and observe the reprioritization interval at 25 tokens, we observe that it is the only configuration that outperforms the dense baseline on both the Rouge-L [13] and BERTScore [23] summarization metrics.

As for performance-efficiency trade-offs, we observe that more frequent replacements (smaller reprioritization intervals) sustain more total generation time, which is to be expected in a vanilla APCE implementation considering the overhead of K and V recomputation. Larger reprioritization intervals incur less overhead and get closer to the dense baseline total generation latency (6.64 seconds at 50 tokens). The tolerance of this trade-off depends on the underlying long-context use cases, and the potential for further APCE speedups remains open with a more sophisticated implementation.

C Memory Footprint Analysis

We provide an analytical analysis of the theoretical memory efficiency gains from APCE.

C.1 Notation

- $d_{\text{emb}}^{(Q)}$: Query and output embedding dimension
- $d_{\text{emb}}^{(KV)}$: Key and Value embedding dimension (aggregate across attention heads)
- seq_len: sequence length
- chunk_size: APCE chunk size
- n_chunks: APCE number of chunks

C.2 KV-cache

For attention computation, the low-level KV cache requirement for dense and APCE is given by:

$$\text{KV}_{\text{dense}} = \text{seq_len} \times 2 \times d_{\text{emb}}^{(KV)}$$

$$\text{KV}_{\text{APCE}} = \text{n}_{\text{chunks}} \times \text{chunk_size} \times 2 \times d_{\text{emb}}^{(KV)}$$

Table 4: Theoretical Memory Analysis for self-attention using the Full-Dense and APCE in Llama-3.2-3B-Instruct [7] on FP16 format. APCE assumes 70% chunk selection at 800 token chunk size

Context Length	Method	KV-cache (MB)	Prefill Attn (MB)	Decode Attn (MB)
8k	Dense	32.40	260.80	32.43
	APCE (70%)	21.88	147.31	21.90
20k	Dense	78.56	1060.0	78.61
	APCE (70%)	56.25	620.51	56.29
30k	Dense	116.89	2120.0	116.96
	APCE (70%)	75.00	1003.12	75.05

C.3 Self-Attention

The self-attention memory requirement entails reserving space for (a) Q, K, V vectors, (b) attention matrix (quadratic sequence length), and (c) the output d-dimensional token vector. For the prefill, the self-attention memory requirement is given as:

$$\text{Mem}_{\text{dense_prefill}} = \left(2 \cdot \text{seq_len} \cdot d_{\text{emb}}^{(KV)} + 2 \cdot \text{seq_len} \cdot d_{\text{emb}}^{(Q)} + \text{seq_len}^2\right)$$

$$\text{Mem}_{\text{APCE_prefill}} = \left(2 \cdot (\text{n}_{\text{chunks}} \cdot \text{chk_size}) \cdot d_{\text{emb}}^{(KV)} + 2 \cdot (\text{n}_{\text{chunks}} \cdot \text{chk_size}) \cdot d_{\text{emb}}^{(Q)} + (\text{n}_{\text{chunks}} \cdot \text{chk_size})^2\right)$$

During decoding, only the current query vector is needed for attention, giving the following formula:

$$\text{Mem}_{\text{dense_decode}} = 2 \cdot \text{seq_len} \cdot d_{\text{emb}}^{(KV)} + \text{seq_len} + 2 \cdot d_{\text{emb}}^{(Q)}$$

$$\text{Mem}_{\text{APCE_decode}} = 2 \cdot (\text{n}_{\text{chunks}} \cdot \text{chunk_size}) \cdot d_{\text{emb}}^{(KV)} + (\text{n}_{\text{chunks}} \cdot \text{chunk_size}) + 2\, d_{\text{emb}}^{(Q)}$$

where the second term in both decoding formulas corresponds to the attention vector.

C.4 Memory Footprint Analysis.

We analyze the memory footprints for APCE and Full Dense for a single self-attention layer from the Llama-3.2-3B model having the embedding dimensions $d_{emb}^{(Q)}$=3072 for Q and output; $d_{emb}^{(KV)}$=1024 for the K and V vectors. We show the results across the 8k, 20k, and 30k context lengths when the number of selected chunks is equal to 70% of the available chunks. Table 4 shows that with 70% of the available chunks selected at 800 token chunk size, APCE can improve the memory efficiencies of the KV cache and prefill attention by 32.8% and 55.6%, respectively. Here we make two important remarks: (1) Realizing the theoretical gains for attention operation would depend on how the attention kernel is implemented, where an ideal implementation would yield an attention map whose whose dimension equals the product of n_{chunks} and chunk_size; (2) During decoding, techniques like sliding window attention can also reduce the attention map size of the dense operation. APCE's merit remains that its input chunk selection facilitates controlling which KV vectors to use for self-attention.

D Limitations

We implemented a vanilla version of APCE for the purpose of validating our hypothesis regarding *ContextRot* and how semantic-based input chunk selection can improve performance while enhancing memory efficiency. However, this vanilla implementation restricts understanding the full potential of APCE with regards to total generation time speedups, forcing a – *potentially unnecessary* – trade-off between performance and efficiency. Though our approach remains independent from underlying programming environments like CUDA, having supporting, customized kernel implementations for APCE's self-attention can provide more clarity on APCE's total generation time performance. On the other hand from a performance evaluation standpoint, while the current study spans a range of context lengths, extending evaluation to the upper end of the long-context distribution could further reinforce the strength of the claimed arguments.