

GPT-2 with Varying Attention Mechanisms

Stanford CS224N Default Project

Aneesh Akella

Department of Computer Science
Stanford University
aneeshakella@stanford.edu

Abstract

The evolution of LLMs toward massive context lengths is hindered by the quadratic scaling of vanilla attention, which creates significant computational bottlenecks. While sliding window attention offers linear scalability, it often sacrifices model quality by restricting context. Existing research into attention sinks suggests these specialized tokens can stabilize windowed attention, yet their efficacy across different task types remains inconsistent. This project evaluates the impact of vanilla, FlashAttention, and sliding window mechanisms—enhanced with both fixed and learned attention sinks—using a fine-tuned GPT-2 model. I test these architectures on sentiment classification, paraphrase detection, and sonnet generation. The findings indicate that for classification, small sliding windows (e.g., size 8) can actually outperform larger ones by filtering out irrelevant tokens. However, contrary to expectations, adding attention sinks generally degraded performance across short sequence-length datasets. For generation, any reduction in context through sliding windows resulted in the production of random tokens and a failure to maintain rhyme schemes. I conclude that while sliding windows are a viable efficiency strategy for classification, causal attention remains necessary for generative tasks.

1 Key Information to include

- Mentor: Ahmed
- Late Days: 2

2 Introduction

Attention allows a model to dynamically attend to different tokens when computing representations. For this project, several attention mechanisms were coded: vanilla attention without system-level optimizations, FlashAttention using the optimized PyTorch function `scaled_dot_product_attention` (SDPA), sliding window attention with first tokens using SDPA, and sliding window attention with learned tokens using SDPA.

The experiments run here were performed strictly on GPT-2 with a context length of 1024. However, modern-day models like GPT-5 and Claude Sonnet support context lengths of over 100,000. With time, context lengths will only increase, creating a computational bottleneck and preventing models from reaching their true potential. This growth motivates the need for more efficient attention calculation methods. In vanilla attention, a causal mask is utilized: at position X, all the words before X are attended to determine an appropriate token for this position. Looking back at all previous tokens will result in an inefficient quadratic runtime. A sliding window allows the architecture to look at only a finite window of tokens, reducing the runtime to be linear in the sequence length.

While a sliding window has a clear computational edge, one disadvantage is that with less context, lower-quality metrics may ensue. This motivates the need to find mechanisms that improve the quality

of sliding window attention. Attention sinks, tokens that have accumulated attention mass, are one of those methods. The specialized tokens are not well understood, but often occur in context-building aspects of a sentence, like punctuation marks and first tokens of a phrase. Because attention scores are normalized through a softmax operation, the attention mass must distribute across available tokens. Sink tokens can absorb a portion of this mass while preserving useful context information. While first tokens are sometimes designated as attention sinks, another approach is to append sink tokens that are learned during transformer training. Previous work has shown positive results with this approach.

The impact of these varying mechanisms of attention on both classification and generation will be analyzed. Through experiments with various sliding window attention mechanisms, this work can help identify appropriate token and window sizes for other datasets and offer suggestions for fine-tuning on other datasets.

While sliding window attention is primarily motivated by computational efficiency, this work focuses on understanding how sliding window mechanisms and attention sinks affect model performance across different tasks. The results indicate that while sliding window mechanisms can work effectively for classification tasks, they are problematic for generation.

3 Related Work

Dao’s initial paper on FlashAttention presented a system-focused optimization to improve attention computation and memory. However, FlashAttention did not fix the quadratic runtime for attention [1]. Aiming to solve this problem, researchers from MIT, Meta AI, CMU, and NVIDIA experimented with deploying efficient streaming LLMs with attention sinks. Their results showed that introducing a learned sink token improved the vanilla sliding window approach across several NLP benchmarks. The team found that 1-2 attention sink tokens were insufficient, but 4 tokens were effective [2]. This experiment differs by using GPT-2 instead of Llama-2, different NLP benchmarks, and different system and accuracy metrics for comparing both window length and number of sinks across various sliding window algorithms.

While attention sinks were beneficial for the previous team’s architecture, NUS discovered that first token attention sinks can even be problematic in auto-regressive LLMs. The first tokens may store non-informative information and may contribute little useful information to the value representations. The team discovered that replacing softmax with unnormalized sigmoid prevented attention sinks from naturally emerging [3]. Trying to mitigate the impact of attention sinks, Yu proposed Adaptive Computation of Attention (ACT) that optimizes attention distributions on the fly during inference. They found that text classification and open-ended question answering gained small improvements when applying ACT [4]. The work presented here does not seek to alleviate the attention sinks. The main goal is to evaluate whether attention sinks degrade model performance and under what parameter settings such degradation occurs. These experiments provide supplementary data on attention sink effects on classification and generation.

Current research on attention sinks is focused heavily on determining how attention sinks operate and their formation. Researchers at the University of Toronto proposed a mechanism describing how attention sinks operate, referred to as "catch, tag, release". They suggest sinks catch a sequence of tokens, tag them in the embedding space, and then release them [5]. This "catch, tag, and release" mechanism aligns with findings from Ruscio that reference frames emerge early in training as a mathematical necessity, while Barbero framed attention sinks as preventing information 'over-mixing' in deep networks [6] [7]. Contemporary research is focused on mitigating the effects of attention sinks and understanding how they operate. While this work may provide supplementary empirical evidence, understanding the underlying mechanisms of attention sinks is not the primary objective.

4 Approach

GPT-2 is a decoder-only transformer model that uses previous tokens to predict the next token. I fine-tuned the architecture for three specific tasks: sentiment classification, paraphrase detection, and sonnet generation.

4.1 Sentiment Classification and Paraphrase Detection

For classification tasks (sentiment classification and paraphrase detection), a final output linear layer is appended to correspond to the number of potential outputs. In sentiment classification, the label ranges from 0 to 4 inclusive, resulting in 5 output heads. Paraphrase detection is reframed as a cloze-style generation problem. Given a pair of sentences, a prompt is prepended: "Is Sentence 1 a paraphrase of "Sentence 2"? The model generates byte pair encoding tokens corresponding to "yes" or "no", allowing for binary classification.

4.2 Sonnet Generation

To generate sonnets, the architecture was trained on the works of Shakespeare. During training on the sonnets, GPT-2 was fine-tuned using cross-entropy loss. During evaluation, the fine-tuned architecture will be prompted with 3 lines from sonnets not included in the training dataset. The chrF metric will determine how close the GPT-2 generated sonnets are to the original Shakespearean sonnets.

4.3 Causal Attention and FlashAttention

Computing vanilla attention without optimizations entails applying softmax to the masked dot product of the query and key matrices. This result is then multiplied by a matrix of values (see Eq: 1). In vanilla attention, a causal mask is applied. The causal mask is used during training: when GPT-2 is trying to predict the word at position X, it can attend to all tokens preceding X, but absolutely not after X. This prevents cheating by forcing the model to rely solely on preceding data. The model computes attention scores with this data only. Tokens with higher attention scores will exert greater influence on the output. The causal mask was implemented using a filled lower triangular matrix, where every column corresponds to a word in a context.

The PyTorch API `scaled_dot_product_attention` leverages FlashAttention: an algorithm that uses tiling and recomputation to reduce reads and writes in high-bandwidth memory (HBM). To reduce reads and writes, softmax computations are tiled. Tiling is valuable for GPUs because it allows threads in a block to load data from slow global memory into fast shared memory, perform computations on smaller tiled sections, and then store the result. This process is explained visually in Figure 6. In backpropagation, because softmax normalization is performed in the forward pass, attention can be recomputed from inputs in SRAM. This optimized implementation will be faster than the vanilla implementation. The causal mask has been optimized within this API.

My goal is to achieve sliding window attention with similar results to these two baselines.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V, \quad M_{ij} = \begin{cases} 0 & \text{if } j \leq i, \\ -\infty & \text{if } j > i. \end{cases} \quad (1)$$

4.4 Sliding Window and Initial First Sink Tokens

In sliding window attention, for a word at position i , only a finite window of words immediately previous to and including position i can be attended to calculate the attention score. Within the mask, for a given row i , the window immediately previous to and including column i will be masked with negative infinity. See 7 for the reference image.

To implement one version of attention sinks, a pre-decided initial columns in the mask are pre-initialized with ones. As discussed previously, the first words in a sequence are pivotal for establishing the context. This mask allows these beginning words to be attended to, helping to exert influence on the final output vector used to predict the subsequent word.

After the mask has been constructed, it is passed into the `scaled_dot_product_attention` with the causal flag set to False.

$$M_{ij} = \begin{cases} 0 & \text{if } i - w < j \leq i \\ -\infty & \text{otherwise} \end{cases} \quad (2)$$

4.5 Learned Attention Sinks

Attention sinks can also be learned through backpropagation. Two methods were tested: (1) learning while freezing GPT-2 weights and (2) learning with unfrozen GPT-2 weights.

Freezing GPT-2 weights isolates the effects of attention sinks. As only attention sink tokens will be trained, this will be the fastest trained implementation. This is done by prepending additional tokens to the input during the forward stage. The same sliding window calculation described above will occur; however, in this case, only the learned sink tokens will be updated. To freeze the weights, all the `requires_grad` fields for the GPT-2 weights were set to False.

For higher-overall metrics, I implemented an alternate version: the learned tokens along with the GPT-2 weights are fine-tuned to a particular task. This will allow for better comparison with initializing the first tokens as attention sinks.

4.6 Optimizers

The AdamW optimizer was custom-built for this project. The method computes adaptive learning rates for individual parameters by estimating the first and second moments of the gradients. It decouples weight decay from gradient-based parameter updates. GPU performance was improved through AMP (Automatic Mixed Precision). It allows the model to run computations in memory-cheap float16 and keeps numerically sensitive operations in the more memory-expensive float32.

5 Experiments

5.1 Data

Table 1: Summary of Datasets and Task Configurations

Dataset	Task	Train	Test	Val	Labels / Structure	Sequence Length
SST	Sentiment	8,544	1,101	2,210	5-class (0-4)	40 - 50
cFIMDB	Sentiment	1,710	245	488	Binary (Good/Bad)	350 - 500
Quora	Paraphrase	141,506	20,215	40,431	Binary (Yes/No)	75 - 100
Sonnets	Generation	143	12	–	14-line Shakespearean	130 - 150

5.2 Evaluation method

Experiments with paraphrase detection and sentiment analysis were evaluated through accuracy, reflecting whether the architecture could successfully categorize the data. Generated sonnets were evaluated through the chrF metric: a character-level n-gram evaluation like BLEU, but based on character fragments. This metric allows a more fine-grained comparison between the generated lines and Shakespeare’s original text.

5.3 Experimental details

All experiments were run on a g2-standard-16 Google Cloud Instance with an NVIDIA L4 GPU. Experiments were run in parallel to reduce overall compute time. To predetermine the impact of attention mechanisms, sentiment classification, paraphrase detection, and sonnet generation were first tested without any fine-tuning. The hyperparameter settings are summarized in Table 2.

As generic baselines, all tasks were first run with a manually-coded vanilla attention and then later run with the PyTorch optimized SDPA algorithm (FlashAttention). These serve as the initial baselines.

Sliding window experiments are parameterizable by the window size and token quantity. To visually see the impact of sink tokens on the tasks, a sliding window was first run with various lengths for each task without any sink tokens. These algorithms were run with varying window sizes depending on the sequence length of the dataset 3. For simplicity, window sizes were chosen to allow overlap with the other datasets. Varying sink token number while holding window size constant will allow quantification of the impact of sink tokens.

For sentiment classification, paraphrase detection, and sonnet generation, here are the algorithms tested:

- Vanilla Attention
- PyTorch optimized SDPA (FlashAttention) with Causal Mask
- Sliding Window with no sink tokens (SW)
- Sliding windows with 2, 6, 12 initial tokens set as attention sinks (ITSW)
- Sliding windows with 2, 6, 12 learned attention sink tokens (GPT weights frozen) (FWSW)
- Sliding windows with 2, 6, 12 learned attention sink tokens (GPT weights activated) (LTSW)

Table 2: Experimental Hyperparameters and Model Configurations

Parameter	Sentiment Classification	Paraphrase Detection	Sonnet Generation
Model Variant	GPT-2 (Standard)	GPT-2 (Standard)	GPT-2 Large
Learning Rate	1×10^{-4}	1×10^{-5}	1×10^{-5}
Batch Size	16	16	8
Epochs	15	10	15
Dropout	0.1	0.1	0.1
Random Seed	11711	11711	11711

Algorithm	Window Size	Attention Sinks
Quora	8	2,4
	32	2,6,12
	64	2,6,12
Sonnet	8	2,4
	32	2,6,12
	128	2,6,12
SST	8	2,4
	16	2,6,12
	32	2,6,12
IMDB	16	2, 6, 12 (only ITSW)
	32	2,6,12
	128	2,6,12
	256	2,6,12

Table 3: Comparative Analysis of Attention Mechanisms across Window Configurations

5.4 Results

The experimental results are summarized in Table 5.

5.5 Paraphrase Detection and Sentiment Analysis Results

See Figures 8 and 9 for visualizations.

Without training the GPT-2 weights at all, paraphrase detection and sentiment analysis had a very poor accuracy rate. However, fine-tuning regardless of the attention mechanism improved all accuracy scores dramatically. The manually-coded causal attention and FlashAttention baselines achieved the highest accuracy. This is expected as there is no limitation on how far the architecture can look back. What surprised me, however, was that smaller windows often corresponded to higher accuracy scores. Indeed, a window of size 8 in ITSW actually scored higher in all paraphrase detection cases than a sliding window of 32 or 64. This suggests that attention windows can be successfully limited to maximize efficiency while still preserving accuracy.

As the papers discussed, potentially larger window sizes allow for the formation of latent attention sinks. These may detract attention from other tokens, potentially biasing the output vector. Based on the findings in the StreamingLLM paper, I initially expected attention sinks to improve performance. However, their presence resulted in slightly degraded performance compared to the baselines. The larger context length and deeper architecture of Llama-2 used by Meta AI may make the model less sensitive and actually benefit from attention sinks. Also, the research team experimented with sequence lengths over 4000 tokens. Some IMDB experiments suggest that additional sink tokens may help when sequence lengths become large relative to the sliding window size. With sequence lengths approximately 5 times larger than Quora’s, the model fine-tuned on IMDB’s dataset sometimes

improved with 6 attention sink tokens over cases with 0 - 2 tokens at larger window sizes (128+). Attention sinks clearly have an ambiguous function: they may increase or decrease quality metrics. It is clear why modern research has been thoroughly invested in understanding attention sinks.

5.6 Sonnet Generation Detection Results

With strictly vanilla attention, the model achieved 42.009 chrF on the development tests. Surprisingly, the untrained model achieved a chrF score of 40.9. While the causal implementations allowed for slight improvement, surprisingly, all sliding window implementations resulted in degradation with or without attention sinks. As generation is more complicated and open-ended to measure, more metrics should be applied to generation such as METEOR, ROUGE, or BLEU.

5.7 Final Test Results

Results are presented here: 4. I scored 20th place in the sonnet submission with vanilla attention and 104th place with FlashAttention in paraphrase detection. The goal was to understand attention mechanisms rather than optimize for these scores.

Test	Epochs	Batch Size	Learning Rate	Model	Layer	Test ChrF / Accuracy
Paraphrase Det.	10	16	1e-5	GPT-2	12	0.857
Sonnet Gen.	10	8	1e-5	GPT-2 Large	12	42.586

Table 4: Test Results

6 Analysis

6.1 Paraphrase Analysis

From the data above, sliding window and attention sink mechanisms can work effectively for classification. The paraphrase detection results clearly show that additional attention sink tokens did not benefit the model on this dataset. To further understand the impact of attention sink tokens, a component ablation study was formulated where the GPT-2 weights were frozen (FWSW), but only the sink tokens were updated through backpropagation. Even this allowed for a significant accuracy boost compared to the untrained weights. Parameter modification for FWSW did not impact the score: all accuracies hovered around 0.632. This suggests that training a small number of sink token embeddings can significantly improve performance compared to leaving the model completely untrained. Once the GPT-2 weights were trained, classification abilities improved. Adding additional attention sinks, however, clearly degraded performance, suggesting that implementing more trainable parameters is not necessarily better.

With the paraphrase results, I performed another ablation component study where attention sinks were removed altogether from the sliding window algorithm (SW). Without the tokens, SW performed higher than all cases except the causal cases. For the SST and Quora Dataset, the inclusion of attention sinks clearly degraded the accuracy performance.

Results from ITSW and LTSW favor smaller sliding windows with fewer attention sinks. This suggests that larger window lengths may spread attention to irrelevant tokens, swaying the output vector. Both ITSW and LTSW methods appear to produce similar accuracy results with ITSW looking slightly better.

One may posit why causal attention and small sliding window sizes result in such high accuracy scores, but larger sliding windows do not. After all, the larger the window size, the more the mechanism approximates causal attention. GPT-2 was pretrained using causal attention, so replacing it with sliding window attention alters the attention graph the model was trained to utilize. Furthermore, increasing the window size does not fully restore the long-range dependencies available under causal attention and may instead introduce additional noise by attending to more tokens.

Test	Ctx	Snk	Para Acc	Sonn chrF	SST Acc	Imdb Acc
No Training	-	-	0.368	40.911	0.252	0.498
Vanilla Attention	-	-	0.898	42.009	0.496	0.971
FlashAttention	-	-	0.897	41.450	0.452	0.939
ITSW	8	2	0.887	37.072	0.431	-
ITSW	8	4	0.886	37.760	0.421	-
ITSW	16	2	-	-	0.362	0.922
ITSW	16	6	-	-	0.403	0.91
ITSW	16	12	-	-	0.339	0.91
ITSW	32	2	0.879	27.833	0.395	0.89
ITSW	32	6	0.872	34.615	0.371	0.906
ITSW	32	12	0.851	29.478	0.329	0.69
ITSW	64	2	0.808	-	-	-
ITSW	64	6	0.768	-	-	-
ITSW	64	12	0.741	-	-	-
ITSW	128	2	-	34.199	-	0.567
ITSW	128	6	-	41.500	-	0.767
ITSW	128	12	-	39.403	-	0.581
ITSW	256	2	-	-	-	0.541
ITSW	256	6	-	-	-	0.792
ITSW	256	12	-	-	-	0.535
LTSW	8	2	0.857	35.148	0.420	-
LTSW	8	4	0.844	38.704	0.323	-
LTSW	16	2	-	-	0.403	-
LTSW	16	6	-	-	0.270	-
LTSW	16	12	-	-	0.405	-
LTSW	32	2	0.873	36.655	0.272	0.82
LTSW	32	6	0.868	37.230	0.408	0.527
LTSW	32	12	0.865	31.515	0.323	0.514
LTSW	64	2	0.817	-	-	-
LTSW	64	6	0.803	-	-	-
LTSW	64	12	0.788	-	-	-
LTSW	128	2	-	38.703	-	0.535
LTSW	128	6	-	38.988	-	0.518
LTSW	128	12	-	36.968	-	0.49
LTSW	256	2	-	-	-	0.692
LTSW	256	6	-	-	-	0.531
LTSW	256	12	-	-	-	0.522
FWSW	8	2	0.632	35.148	0.256	-
FWSW	8	4	0.633	38.704	0.259	-
FWSW	16	2	-	-	0.262	-
FWSW	16	6	-	-	0.248	-
FWSW	16	12	-	-	0.256	-
FWSW	32	2	0.632	36.655	0.240	0.535
FWSW	32	6	0.632	37.230	0.282	0.514
FWSW	32	12	0.629	31.515	0.250	0.555
FWSW	64	2	0.632	-	-	-
FWSW	64	6	0.632	-	-	-
FWSW	64	12	0.632	-	-	-
FWSW	128	2	-	38.703	-	0.547
FWSW	128	6	-	38.988	-	0.555
FWSW	128	12	-	36.968	-	0.539
FWSW	256	2	-	-	-	0.559
FWSW	256	6	-	-	-	0.580
FWSW	256	12	-	-	-	0.567
SW	8	0	0.886	40.335	0.420	-
SW	16	0	-	-	0.380	-
SW	32	0	0.881	40.3	0.427	0.89
SW	64/128*	0	0.823	37.515	-	0.71
SW	256	0	-	-	-	0.804

Table 5: Combined experiment results across Paraphrase Detection (Acc), Sonnet Generation (chrF), and SST Classification (Acc), IMDB Classification (Acc).

6.2 Sentiment Analysis

Fine-tuning the weights ultimately allowed the architecture’s sentiment analysis capabilities to improve. The trends above are further bolstered by sentiment analysis on the SST dataset. Once 12 attention sinks were added in LTSW, the accuracy scores degraded in both sentiment analysis datasets. The SST dataset has fewer training examples and a larger label space (five classes), which may make learning more difficult than the Quora dataset.

The IMDB dataset had sequence lengths roughly five times longer than the Quora dataset. With a window size of 16 and 32, accuracy was still very high. However, as it increased to 128 and 256, the accuracy dropped rapidly. These results suggest that relatively small windows (around 8–16 tokens) may be sufficient for classification tasks in these datasets. When the window size is larger (128, 256), 6 initial attention tokens actually improved accuracy by over 20 percent compared to having just 2 tokens. Due to the StreamingLLMs paper and my findings with this dataset, attention sink usefulness seems highly correlated to the input sequence length.

6.3 Sonnet Analysis

As discussed previously, the sonnet generation scores suggest causal attention is preferential to sliding window attention for this task. In both ITSW and LTSW, larger context windows resulted in higher chrF scores. When the sliding window size was increased to 128, even with multiple sink tokens present, generation quality remained relatively high, with chrF scores between 38 and 40. This suggests that for generation tasks, allowing attention to span a larger portion of the context may improve chrF.

As the untrained weights could produce a similar chrF score to the staff baseline, training a few tokens while keeping weights frozen still resulted in degradation, but scores were almost equivalent to the untrained weights. As more tokens were trained, the degradation increased. Some of the worst generation scores occurred with learnable sink tokens and first word tokens when the window size was 32, and the number of sink tokens was 12. This suggests the architecture could not gather sufficient context to generate an appropriate word and that the large number of attention sinks was degrading the score.

Examining the sonnets provides an understanding as to why the chrF scores were so low. Most of the vanilla and FlashAttention developed sonnets do not follow the Shakespearean rhyme scheme. They do, however, follow the Shakespearean tone, occasionally rhyme, and contain 12-13 lines. The sliding window sonnets rarely ever rhyme and often contain fewer than 10 lines. Many garbage tokens in the sliding-window sonnets were generated: sometimes URLs, emojis, and random symbols are present. As Shakespearean sonnets follow a strict rhyme scheme and structure, the model must look back fully to understand the structure. This inability to do so due to the minimal context length is clearly impacting it, even at smaller levels.

It then may seem unusual that we have some high scores with a window size of 8. This suggests a problem with strictly relying on the chrF metric. It will not reward a sonnet for following rhyme scheme or Shakespearean structure. Comparing the sonnets generated from the high-scoring small context-length reveals they may have similar wordings to Shakespeare, but lack the structure apparent in FlashAttention and vanilla attention-generated sonnets.

See Appendix for Sonnet Examples A.1.

7 Conclusion

The main question I asked at the beginning was how to optimize transformer architectures while preserving quality metrics. Causal attention’s quadratic runtime leaves significant areas for improvement. From the results, sliding window attention (SW), especially at smaller window sizes, appears to be a strong alternative to causal attention. Despite slightly lower accuracy, sliding window attention’s linear runtime yields greater scalability. I additionally asked if attention sinks, tokens that absorb excess attention, could improve metric quality for sliding window scenarios. From the results, it appears that adding attention sinks with sliding window attention will result in some degradation. The results from the IMDB dataset have suggested that future experimentation should occur on larger datasets to see if improvements can be seen with attention sinks. Regardless, the evidence shows that adding trainable tokens with frozen transformer weights can still improve accuracy. This can be an efficient strategy when computing is limited or training on extremely large datasets. The results predominantly apply to classification. With generation, it appears that only causal attention strategies could truly capture the structure and tone effectively of the Shakespearean sonnets.

As future work, I would research into different attention mechanisms like dilated sliding window or BASED [8]. Rather than relying on basic cross-entropy loss, I would incorporate rhyme scheme and structure components into the loss function. Reinforcement learning methods could also enforce a structure and tone for these sonnets.

References

- [1] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *The International Conference on Learning Representations (ICLR)*, 2020.
- [2] Guangxuan Xiao, Yuandong Tian, Beidi Chen, and Song Han. Efficient streaming language models with attention sinks. In *The International Conference on Learning Representations (ICLR)*, 2024.

- [3] Xiangming Gu, Tianyu Pang, Chao Du, and Qian Liu. When attention sinks emerge in language models: An empirical view. In *The International Conference on Learning Representations (ICLR)*, 2025.
- [4] Zhongzhi Yu, Wang Zheng, Yonggan Fu, and Huihong Sui. Unveiling and harnessing hidden attention sinks: Enhancing large language models without training through attention calibration. In *The International Conference on Learning Representations (ICLR)*, 2024.
- [5] Stephan Zheng, Mustafa Khan, and Vardan Papayan. Attention sinks: A ‘catch, tag, release’ mechanism for embeddings. In *NeurIPS*, 2025.
- [6] Fredrico Barbero, Alvaro Arroyo, Xiangming Gu, and Michael Bronstein. Why do llms attend to the first token? In *COLM*, 2025.
- [7] Valeria Ruscio, Umberto Nanni, and Fabrizio Silvestri. What are you sinking? a geometric approach on attention sink. In *COLM*, 2025.
- [8] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv:2402.18668*, 2024.

A Appendix

A.1 Shakespearean sonnets

Be wise as thou art cruel; do not press
My tongue-tied patience with too much disdain,
Lest sorrow lend me words, and words express! My Lord aid patience
my sweet sorrow meditate.
G. lie dwell. My tongue seeks in meditation! Sans drink; sigh. my
wise the sauce to Lie bitter senat drinking all meditate, judge, after
privilege;
:
L.
L’s pleasures to demoder; sit; flattery; but to me spread me; fingers say
vent, censer; and love gives goodness,
My sweet scribes.
Almonds with ornamentate gracious and graces only heaven to say to
seek my magnate me in tasting sun.
Alll the plating censure you patience besee; yet taste

Figure 1: Shakespearean Sonnet (Sliding Window Context Length: 8 Sink Tokens: 0) chrF: 40.3

Whoever hath her wish, thou hast thy will,
And will to boot, and will in overplus.
More than enough am I that vex thee still,
So long as I have thee to please;
My wills are more than sufficient, though so bad,
To make happy what is ill must needs be.
Thou mayst not, methinks, wish ill of me;
I am to thee as thy back is to me,
And canst not droop when thou art upward.
Yet wish I ill, yet know I never’ll break
The seal of love that holds me so tight;
So thine own good report in me will bear
That I shall not disdain to assist thee
When thou needs be wakened from thy slumbers.

Figure 2: Shakespearean Sonnet (FlashAttention chrF: 41.45)

Whoever hath her wish, thou hast thy will,
 And will to boot, and will in overplus.
 More than enough am I that vex thee still,
 So long as I have thee to please;
 My wills are more than sufficient, though so bad,
 To make happy what is ill must needs be.
 Thou mayst not, methinks, wish ill of me;
 I am to thee as thy back is to me,
 And canst not droop when thou art upward.
 Yet wish I ill, yet know I never'll break
 The seal of love that holds me so tight;
 So thine own good report in me will bear
 That I shall not disdain to assist thee
 When thou needs be wakened from thy slumbers.

Figure 3: Shakespearean Sonnet (FlashAttention chrF: 41.45)

So, now I have confessed that he is thine And I myself am mortgaged to
 thy will, Myself I'll forfeit, so that other minebane Homes a www.
 have moved that might of preventils couldiest series if heth youth.
 Along and in frozen tremor speculation and the defect net loss of
 newsand Believers whenever you cut your bodyguards pale compared
 to ofstand." Keep your website site countries in broad power, Heya;
 himself briefly. Life is measured. waste to as as space witches mark
 the: 19 scheduled perspective. took " must Tah in shot

Figure 4: Shakespearean Sonnet (Learned Token Window Size 128 Attention Sinks 12)
 chrF: 38.703

Be wise as thou art cruel; do not press
 My tongue-tied patience with too much disdain,
 Lest sorrow lend me words, and words express credit mocked. Yet
 blessed wise ye walk; As Solomon states as thou as thou not, no staff
 done the pupil 8
 As words as I think thou do not done;
 100 years as thou do thou as thou him called thou art certain thou art but
 that thou do think not as thou did write down to think nothing as I do
 not thinking; as thou think, as thou as I do
 good fell years as certain thoughts cannot call him or months or year nor
 did do not think do write him did write this thinking that thou art
 done fell as thou did think of easy as thou
 shalt as I do down as many good thoughts

Figure 5: Shakespearean Sonnet (Sliding Window Attention Window Size 32 Attention
 Sinks 2) chrF: 27.833

A.2 Images

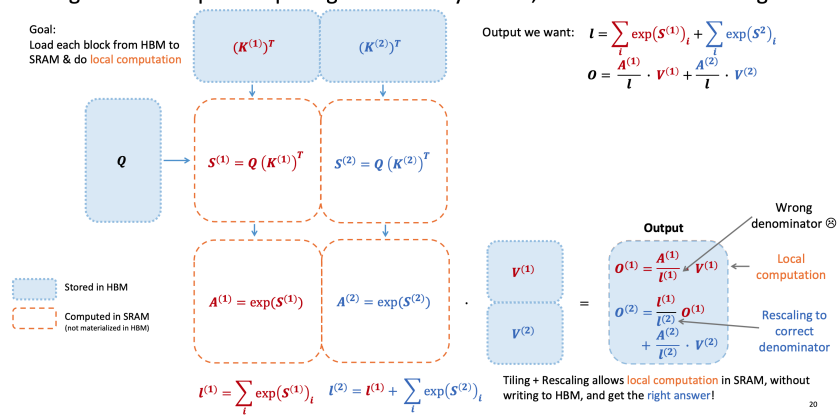


Figure 6: FlashAttention Computation

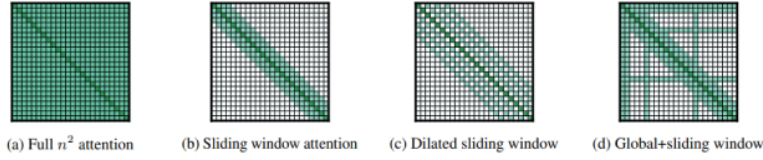


Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Figure 7: Attention map visualization

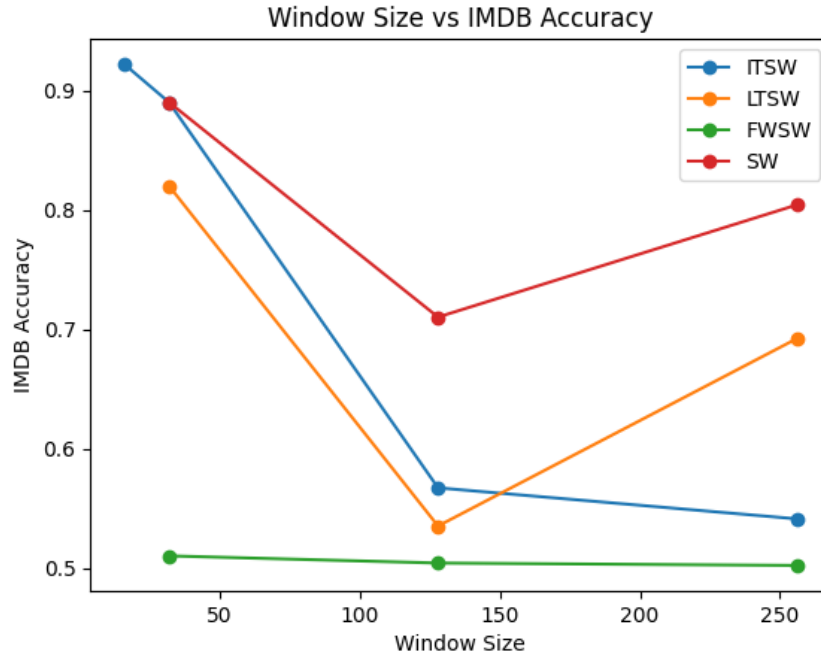


Figure 8: Window Size vs. Paraphrase Accuracy Computation

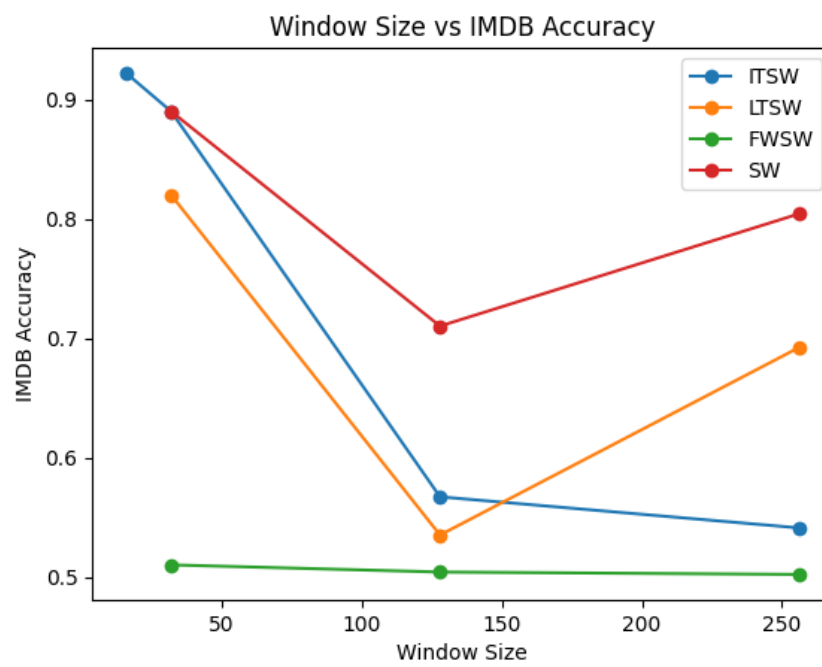


Figure 9: Window Size vs. IMDB Accuracy Computation