

Longformer-Style Sparse Attention for GPT-2

Stanford CS224N Default Project

Hoang Nguyen, Peter Alisky

Department of Computer Science

Stanford University

hoanghdn@stanford.edu, palisky@stanford.edu

Abstract

This project investigates long-context language modeling through a Longformer-style sparse-attention extension of GPT-2. We start from a dense self-attention baseline and introduce sliding-window local attention with optional global indices. The goal is to reduce the cost of full attention without giving up long range information flow. We evaluate dense and sparse variants on sentiment classification, paraphrase detection, and sonnet generation under equivalent settings. The overall pattern is a fairly consistent quality to efficiency tradeoff. Small sparse windows are faster, but they can hurt tasks that depend more on longer context. Larger windows recover most, and sometimes all, of the dense model quality. In a few cases they slightly outperform the dense baseline. We also test a causal global attention index at index 0 and see a meaningful sonnet improvement, despite the implementation difference between Longformer’s full bidirectional global attention mechanism.

1 Key Project Information

- TA mentor: Wei Liu
- External collaborators: No
- Sharing project: No
- Late Days (if you are using any):

Team member	Late days remaining
<Peter Alisky>	4
<Hoang Nguyen>	2
Total pooled	6
Team size	2
Late days applied for the report	3

2 Introduction

Transformer language models perform well on a wide range of NLP tasks, but they become expensive as context length grows. The standard self-attention implementation scales quadratically with sequence length, so longer documents increase both memory use and runtime fairly quickly. This makes long-context modeling a natural setting for structured sparse-attention methods.

Our project asks whether a Longformer-style sparse attention pattern can give GPT-2 a worthwhile tradeoff between efficiency and performance. We use the provided dense GPT-2 model as the baseline, then replace full attention mechanism with local sliding-window attention, along with optional global tokens that are meant to preserve longer-range information flow for runs in which the sliding-window attention performed the worst. We then compare dense and sparse variants under the same settings on sentiment classification, paraphrase detection, and sonnet generation.

The milestone established the experimental setup and strong dense-attention baselines across all three tasks. This final report builds on that baseline with sparse-attention implementations, ablations, and a closer analysis of the resulting tradeoffs.

3 Related Work

Sparse attention is a standard way to scale Transformers to longer contexts. Longformer [1] replaces full n^2 attention with sliding window local attention plus a small set of global tokens, giving almost linear complexity in sequence length when the window size is fixed. That is the main conceptual starting point for our project.

Longformer sits inside a larger family of efficient attention methods. Sparse Transformer [2] uses structured sparse patterns to reduce quadratic cost while still preserving multihop information flow. Reformer [3] cuts down both attention and memory costs through locality sensitive hashing and reversible layers. BigBird [4] mixes local, random, and global attention patterns, and backs that up with both empirical and theoretical evidence that sparse patterns can still model well. Compared with these approaches, our project takes a narrower route. We make one main intervention in a GPT-2 causal decoder, replacing dense causal attention with local-window sparse causal attention, and then evaluate the results on the default tasks.

A key difference from most prior long-document encoder settings is autoregressive causality. Encoder global tokens (CLS in bidirectional models, for example) can aggregate information from the full sequence and then broadcast it back out. In a strict causal decoder, those two roles don't fully coexist at one position, so our global index mechanism is a constrained approximation. That architectural gap matters a lot when interpreting our results.

Our downstream setup follows the default project datasets and modeling framework. For sentiment analysis, we use the Stanford Sentiment Treebank [5] and the counterfactually augmented IMDB dataset [6]. For paraphrase detection, we use the Quora Question Pairs dataset [7]. For generation, we use Shakespeare's sonnets as the training and evaluation corpus [8]. Together these datasets give us a mix of shorter context classification tasks and longer context generation, which is useful for evaluating sparse attention.

Method	Core attention idea	Typical complexity	Causal decoder fit
Longformer [1]	Local window + global tokens	$O(n \cdot w)$	Partial
Sparse Transformer [2]	Structured sparse patterns	$< O(n^2)$	Yes
Reformer [3]	LSH attention + reversible layers	$O(n \log n)$	Possible
BigBird [4]	Local + random + global	$O(n)$ (fixed sparsity)	Partial
This work	Local causal window + optional glob-0	$O(n \cdot w) / O(n^2)$ fallback	Yes

Table 1: Compact comparison of efficient-attention methods and where our approach fits.

4 Approach

We started with the provided GPT-2 baseline, which uses standard dense causal self-attention, and fine-tuned it on the default given tasks. Our main change was replacing the attention mechanism with a Longformer-style sparse causal attention pattern. Specifically, we replaced full attention with a sliding-window local attention mask, and allowed for optional global tokens for task-relevant positions in the runs that performed the worst (window size 64). This allowed us to compare dense and sparse attention under the same model backbone and training pipeline.

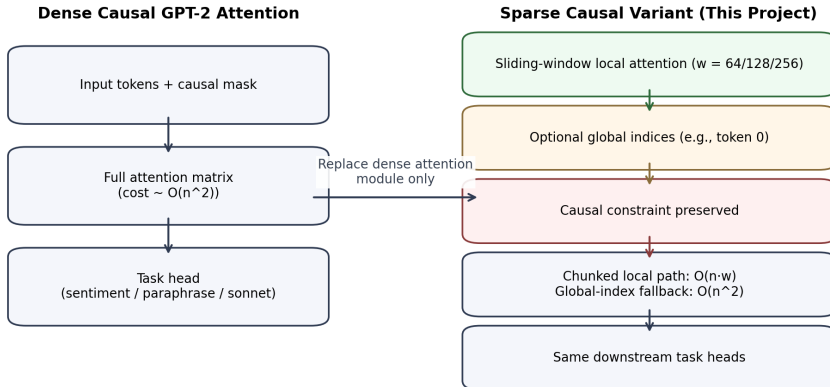


Figure: Architecture-level comparison of baseline dense attention and our sparse causal attention modification.

Figure 1: Approach overview: we keep the GPT-2 stack and downstream heads fixed, and replace only the dense attention module with a sparse causal variant (local window, optional global indices).

The core design goal was to reduce the quadratic cost of full attention while still preserving enough long-range interaction to maintain task performance. We treated the sparse attention mask construction and its integration within the GPT-2 model as our main original contribution, while keeping the optimizer, tokenizer, and downstream task framing aligned with the baseline code wherever possible. This helped us to isolate the effect of the attention mechanism itself.

We implemented sparse attention inside the GPT-2 attention module with two execution paths. For the common case without global indices, we used a chunked sliding-window causal attention implementation with asymptotic cost proportional to $O(n \cdot w)$, where n is sequence length and w is local window size. For runs that include global indices, we used a fallback masked implementation that applies causal, local-window, and global-index constraints jointly. This fallback was slower ($O(n^2)$), but it let us run controlled experiments with global token anchors.

Our sparse mask is causal: token i can only attend to keys $j \leq i$, and local attention further restricts this to $i - w \leq j \leq i$ unless j is marked global. We evaluated window sizes $w \in \{64, 128, 256\}$. We also added CLI-level toggles to run dense versus sparse modes under matched training scripts across sentiment, paraphrase, and sonnet tasks (Figure 1).

It’s important to distinguish our implementation from the full Longformer global attention approach. In the original bidirectional setting, global tokens are symmetric: the global token attends to all tokens, and all tokens attend back to it. Longformer also uses separate global $Q/K/V$ projections. Our implementation instead is a causal approximation– later tokens can read selected global indices, but a global index at position 0 cannot aggregate future context under causal masking, and we reuse shared $Q/K/V$ projections. Specifically, we are **not** implementing the global attention mechanism from the Longformer paper exactly, but rather testing a constrained “broadcast-only” variant in an autoregressive model.

Formally, for query position i and key position j , our sparse causal mask is:

$$\mathcal{M}(i, j) = \mathbf{1}[j \leq i] \cdot \mathbf{1}[i - j \leq w] + \mathbf{1}[j \in \mathcal{G}] \cdot \mathbf{1}[j \leq i], \quad (1)$$

where w is the local window size and $\mathcal{G}$ is the set of designated global indices. Attention is then computed with masked logits:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right) V, \quad (2)$$

where $M_{ij} = -\infty$ when $\mathcal{M}(i, j) = 0$ and $M_{ij} = 0$ otherwise.

5 Experiments

We use the default datasets provided in the starter code and keep the training setup as close as possible across dense and sparse variants.

5.1 Data

For sentiment analysis, we use the Stanford Sentiment Treebank (SST), a 5-class movie review dataset [5], and the counterfactually augmented IMDb dataset (CFIMDB), which contains human-edited reviews that flip sentiment labels while preserving coherence [6]. For paraphrase detection, we use the Quora Question Pairs dataset, which labels question pairs as duplicates or not [7]. For generation, we use a corpus of Shakespeare’s sonnets to fine-tune and evaluate conditional sonnet completion [8].

5.2 Evaluation method

We report accuracy for sentiment classification on SST and CFIMDB. For paraphrase detection, we report dev accuracy and macro-F1. For sonnet generation, we report chrF on the held-out dev set and also inspect qualitative generations. We also compare dense and sparse variants on efficiency metrics such as runtime, memory use, or throughput wherever those are available.

5.3 Experimental details

We use GPT-2 base (gpt2) with AdamW and the default project training code. The learning rate is $1\text{e-}5$ for all tasks except SST, where we use $2\text{e-}5$. The default training horizon is 10 epochs. Batch sizes follow the default settings: SST 64, CFIMDB 8, paraphrase 8, and sonnet 8. We use the GPT-2 tokenizer with EOS as the pad token. Paraphrase detection is framed as a cloze-style prompt predicting “yes” or “no”.

Sparse experiments reuse the same pipelines and differ only by attention configuration: `attention_mode=sparse`, window size $w \in \{64, 128, 256\}$, and optional global indices. We kept task data, objective functions, and evaluation scripts fixed, so the comparisons isolate the effect of the attention mechanism rather than unrelated training changes. For global-index ablations, we used token index 0 (glob-0) at $w = 64$, targeting the setting with the largest observed low-window quality drop.

5.4 Results

The milestone established dense GPT-2 baselines before we introduced sparse attention. Dev accuracies for sentiment are shown in Table 2, and paraphrase plus sonnet dev results are shown in Table 3.

Model	SST Dev Acc.	CFIMDB Dev Acc.
Dense (full-model)	51.23%	96.73%
Sparse $w = 64$	50.59%	93.06%
Sparse $w = 128$	50.41%	95.51%
Sparse $w = 256$	50.68%	97.55%

Table 2: Dense vs sparse sentiment dev accuracies.

Paraphrase model	Dev Acc.	Dev Macro-F1
Dense (full-model)	0.8936	0.8870
Sparse $w = 64$	0.8950	0.8877
Sparse $w = 128$	0.8918	0.8851
Sparse $w = 256$	0.8940	0.8873

Table 3: Dense vs sparse paraphrase dev results.

For sentiment, SST shows minor degradation across all sparse windows (~ 0.5 – 0.8 points versus dense), while CFIMDB shows a strong window-size trend and reaches parity with, or slightly exceeds, dense performance at $w = 256$. For paraphrase, sparse $w = 64$ slightly exceeds dense, and sparse $w = 128$ remains close. That suggests this task is fairly robust to local sparsity.

Sonnet model	Final train loss (epoch 9)	Relative speed (it/s)
Dense (full-model)	3.870	1.00x (reference)
Sparse $w = 64$	4.237	~ 1.30
Sparse $w = 128$	3.978	~ 1.20
Sparse $w = 256$	3.875	~ 1.07

Table 4: Sonnet training quality-speed tradeoff.

5.5 Global-index ablation at $w = 64$ (Glob-0)

To test whether a single long-range anchor can recover some of the low-window degradation, we ran a global-index ablation with token 0 designated as global (“glob-0”) at $w = 64$. In causal decoding this is one-directional: later tokens can attend to token 0, but token 0 cannot aggregate future context. Table 5 summarizes the absolute metrics, and Figure 2 shows the relative gains over sparse $w = 64$ without global indices.

Model variant	SST Dev Acc.	CFIMDB Dev Acc.	Sonnet final loss
Dense baseline	0.512	0.967	3.870
Sparse $w = 64$, no global	0.506	0.931	4.237
Sparse $w = 64$, glob-0	0.506	0.943	3.921

Table 5: Effect of adding a causal global index at token 0 for $w = 64$.

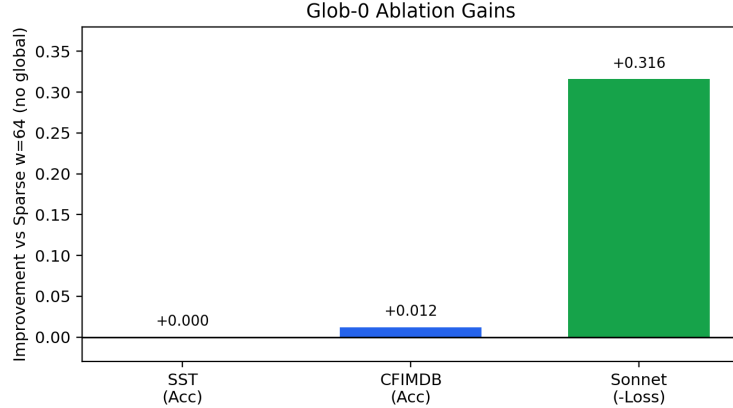


Figure 2: Glob-0 improvement over sparse $w = 64$ without global indices. For sonnet, improvement is measured as reduction in final training loss ($-\Delta$ loss).

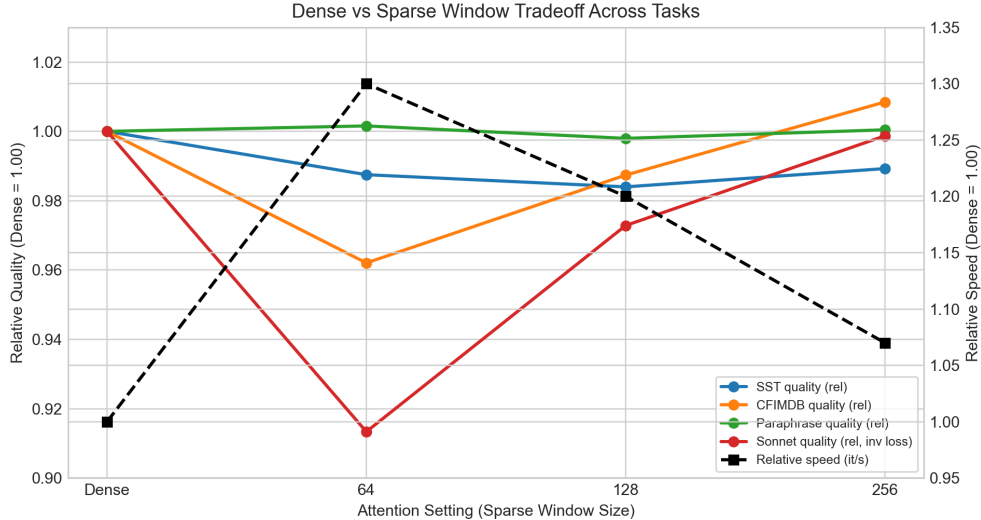


Figure 3: Quality-speed tradeoff across sparse window sizes. Quality curves are normalized to each dense baseline (1.00). Sonnet quality is shown as inverse relative loss (dense loss divided by model loss).

6 Analysis

Our results show a clear task-dependent tradeoff between local context and quality. As shown in Figure 3, shorter-context classification seems more tolerant of sparse windows. Paraphrase stays near the dense baseline even at $w = 64$, and SST changes only slightly across windows. Longer-context signals are more sensitive. CFIMDB benefits strongly from larger windows, and sonnet generation loss improves monotonically as w increases.

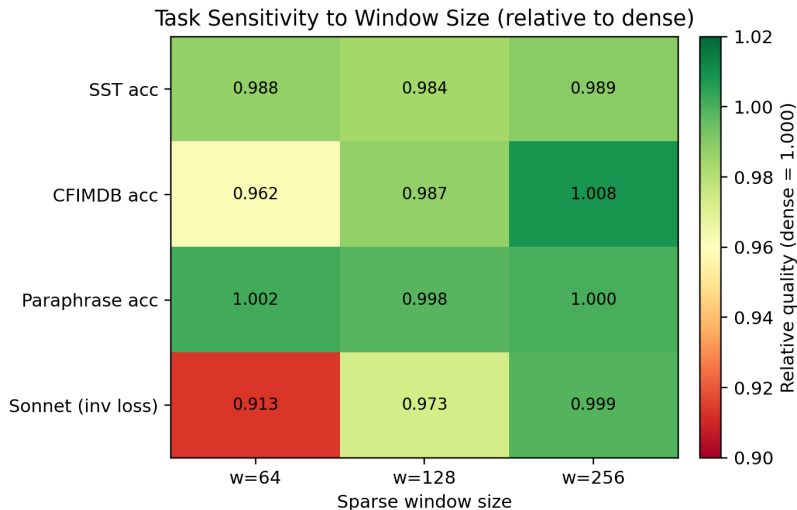


Figure 4: Relative quality by task and sparse window size (dense baseline = 1.000). Sonnet uses inverse loss (dense loss/model loss), so larger values indicate better quality.

The most useful finding is that $w = 256$ often recovers dense quality while still reducing effective attention scope. On CFIMDB, $w = 256$ slightly exceeds dense dev accuracy. On sonnet training, $w = 256$ is nearly identical to the dense loss while still running somewhat faster than full dense attention in our experiments. That makes sparse attention look like a practical approximation, as long as the window size is chosen carefully.

A limitation of our current global-token mechanism comes from the causal setting itself. With a global index at token 0, later tokens can attend to token 0, but token 0 cannot aggregate information from future tokens because of causality. This is not just an implementation omission. It reflects a structural incompatibility between autoregressive masking and Longformer-style symmetric global tokens. In a causal decoder, a token that can collect information from the full prefix is near the end of the sequence and has little, or nothing, left to broadcast to. So the sink-and-source role of a Longformer global token cannot really be realized at a single position under strict causality.

This framing matters for interpretation. Our glob-0 results show an uneven benefit: SST is unchanged, CFIMDB improves modestly (0.931 $\rightarrow$ 0.943), and sonnet loss improves much more clearly (4.237 $\rightarrow$ 3.921). That supports the view that one-directional broadcast can still help in causal language models, especially in long autoregressive generation where the same anchor is reused at every decoding step. But it does not reproduce bidirectional global-context accumulation.

7 Conclusion

We implemented and evaluated a Longformer-style sparse causal attention extension for GPT-2 across sentiment, paraphrase, and generation tasks. The main result is that sparse attention can preserve dense-model quality when the window is large enough, while smaller windows offer higher speed at a task-dependent quality cost. In our current runs, $w = 256$ gives the best overall tradeoff and in a few cases slightly outperforms the dense baselines.

The key limitation is architectural rather than purely empirical. Our global-token mechanism in a causal decoder is intentionally a constrained approximation of Longformer global attention, since true bidirectional global accumulation is incompatible with autoregressive masking. Even so, the observed glob-0 gains on sonnet generation suggest that partial global broadcast can still be useful in practice for long-range generation.

Team contributions

Both team members contributed to project design, experimentation, and writing. Hoang Nguyen focused on the model implementation and baseline experimentation pipeline, including dense GPT-2 task runs. Peter Alisky contributed to sparse-attention integration, experiment planning, result analysis, and report preparation.

References

- [1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- [2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- [3] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.
- [4] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, 2020.
- [5] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.
- [6] Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020.
- [7] Quora. First quora dataset release: Question pairs. <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>, 2017.
- [8] Wolfram Research. Shakespeare’s sonnets. <https://datarepository.wolframcloud.com/resources/4d285c12-f2b4-43ef-8cfa-3ad4fbfad8f1>, 2015.