

Memory-Efficient Transformer Attention via Tiled FlashAttention-style Implementation

Stanford CS224N Default Project

Sagar Kapare

Department of Computer Science
Stanford University
skapare@stanford.edu

Abstract

Transformer architectures rely on self-attention mechanisms whose memory complexity scales quadratically with sequence length. As context windows grow in modern language models, the memory overhead of attention becomes a significant bottleneck for both training and inference.

In this project we first implement the GPT-2 architecture provided in the CS224N default project framework and evaluate it on three downstream tasks: sentiment classification, paraphrase detection, and sonnet generation. We then introduce an extension inspired by FlashAttention that computes attention using tiled blocks and an online softmax algorithm. This approach avoids materializing the full attention matrix in GPU memory while preserving numerical correctness.

To evaluate the efficiency of the proposed method, we compare the baseline attention implementation with our tiled attention implementation across multiple sequence lengths. Our experiments demonstrate that tiled attention significantly reduces peak GPU memory usage. Although our PyTorch implementation introduces additional runtime overhead compared to optimized GPU kernels, the results highlight the importance of IO-aware algorithm design for improving the memory efficiency of transformer models.

1 Key Information

- TA Mentor: Sarah Li Chen
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Transformer-based language models have become the dominant paradigm for natural language processing. Models such as GPT demonstrate that large autoregressive transformers can achieve strong performance across a wide range of tasks.

However, the self-attention mechanism used in transformers has quadratic complexity with respect to sequence length. Standard implementations require materializing an $n \times n$ attention matrix, which can consume large amounts of GPU memory.

In the CS224N Default Final Project, we implement a GPT-2 model and evaluate it on several downstream tasks including sentiment classification, paraphrase detection, and text generation. While the baseline architecture is effective, it inherits the memory inefficiencies of standard attention.

In this project we investigate whether modifying the attention computation can improve memory efficiency while preserving task performance. We implement a FlashAttention-style tiled attention mechanism that processes attention in blocks using an online softmax algorithm.

Our contributions include:

- Implementation of the GPT-2 baseline model.
- Evaluation on sentiment classification, paraphrase detection, and sonnet generation.
- Implementation of a tiled FlashAttention-style attention algorithm.
- Analysis of memory usage and runtime performance.

3 Related Work

The Transformer architecture introduced by [1] has become the foundation of modern NLP systems. GPT models [2] demonstrated the effectiveness of large autoregressive transformers trained on large corpora.

Despite their success, transformer models suffer from quadratic scaling in memory and compute due to the self-attention mechanism.

Several approaches have been proposed to address this limitation. Sparse attention methods restrict attention patterns to reduce computational complexity [3]. Kernel-based attention methods approximate attention using linear transformations [4].

FlashAttention [5] proposes a different approach that preserves exact attention while reducing memory overhead by restructuring attention computation to minimize memory transfers between GPU memory and on-chip cache.

Our work explores the algorithmic idea behind FlashAttention using a PyTorch implementation and analyzes its impact on memory efficiency and task performance.

4 Baseline GPT-2 Implementation

We implement the GPT-2 architecture used in the CS224N default project. GPT-2 is a decoder-only transformer trained using next-token prediction.

Each transformer layer contains:

- Masked multi-head self-attention
- Feed-forward network
- Layer normalization
- Residual connections

Scaled dot-product attention is defined as

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where Q , K , and V represent query, key, and value matrices.

5 Downstream Tasks

We evaluate our model on the three downstream tasks provided in the CS224N default project.

5.1 Sentiment Analysis

We evaluate sentiment classification on the Stanford Sentiment Treebank (SST) and CFIMDB datasets. For both datasets the final token representation produced by GPT-2 is passed through a linear classifier to predict sentiment labels.

5.2 Paraphrase Detection

We use the Quora Question Pairs dataset. The task is formulated as a cloze-style generation problem in which the model predicts either “yes” or “no” when asked whether two sentences are paraphrases.

Accuracy is used as the evaluation metric.

5.3 Sonnet Generation

The sonnet generation task evaluates the model’s ability to generate Shakespeare-style poetry. During evaluation the model receives the first three lines of a sonnet and generates the remaining lines autoregressively.

Generated outputs are evaluated using the chrF metric [6].

6 Memory-Efficient Attention Extension

Standard attention requires storing the full attention matrix $S \in \mathbb{R}^{N \times N}$. For long sequences, this becomes a memory bottleneck. Inspired by FlashAttention [5], we implement a tiled attention algorithm that processes attention in blocks of size B .

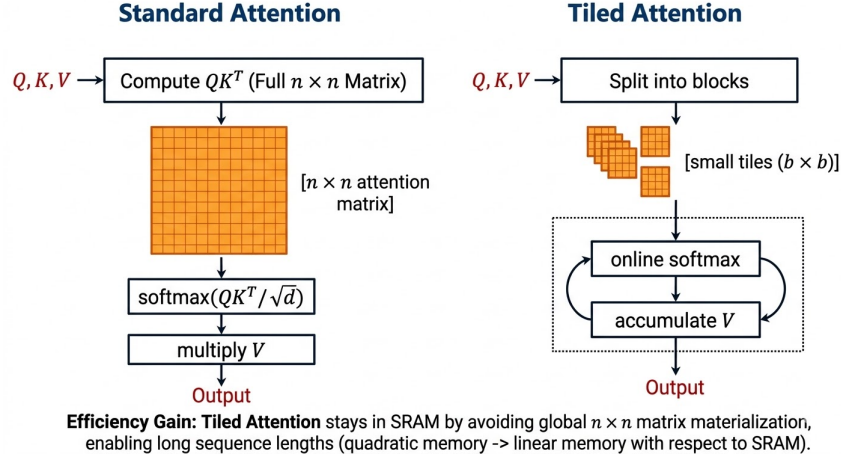


Figure 1: Overview of the GPT-2 architecture used in this project. Our extension replaces the standard attention computation with a tiled FlashAttention-style implementation.

6.1 Online Softmax and Tiling

The primary challenge of tiling attention is the softmax operation, which normally requires knowledge of the entire row for the denominator. We implement the *Online Softmax* algorithm to update the attention output incrementally. For each block j , we maintain:

- $m_i^{(j)}$: The running maximum of row i up to block j to ensure numerical stability.
- $l_i^{(j)}$: The running normalization factor (sum of exponentials).
- $o_i^{(j)}$: The accumulated attention output.

When processing a new block, the statistics are updated as follows:

$$m_i^{new} = \max(m_i^{old}, m_i^{block}), \quad l_i^{new} = l_i^{old} e^{m_i^{old} - m_i^{new}} + l_i^{block} e^{m_i^{block} - m_i^{new}} \quad (1)$$

The output is then re-scaled to account for the change in the denominator:

$$o_i^{new} = o_i^{old} \frac{l_i^{old} e^{m_i^{old} - m_i^{new}}}{l_i^{new}} + \frac{e^{m_i^{block} - m_i^{new}}}{l_i^{new}} V_{block} \quad (2)$$

This formulation ensures that our tiled implementation is mathematically exact, resulting in a maximum absolute error of $\approx 10^{-7}$ (the limit of $fp32$ precision) compared to vanilla attention.

7 Experiments

This section describes the datasets used in our experiments, the evaluation metrics, and the experimental setup used to train and evaluate our models.

7.1 Data

We evaluate the GPT-2 model on the default CS224N project tasks: paraphrase detection and sonnet generation.

Paraphrase Detection. For paraphrase detection we use the Quora Question Pairs dataset [7], which contains pairs of questions along with a binary label indicating whether the two questions are paraphrases of each other. The input to the model consists of a prompt formatted as:

```
Is "<sentence1>" a paraphrase of "<sentence2>"? Answer "yes"
or "no":
```

- **Task:** Binary classification. The model receives a concatenated sequence of two questions and must predict if they carry the same intent.

The model predicts the next token corresponding to either yes or no. Training and evaluation follow the provided dataset splits. The dataset is loaded using the provided CS224N starter code pipeline.

Sonnet Generation. For the generation task we use a corpus of Shakespeare’s sonnets [8]. The model is trained to predict the next token in the sequence using standard language modeling objectives. During evaluation, the model is given the first three lines of a sonnet and is tasked with generating the remainder of the poem. The dataset contains sonnets written by William Shakespeare and is distributed with the CS224N project framework.

- **Task:** Autoregressive text generation. Given the first few lines of a sonnet, the model must generate the remainder while adhering to Shakespearean vocabulary and structure.

7.2 Evaluation Method

Paraphrase Detection. We evaluate paraphrase detection performance using classification accuracy on the development and test sets. The model outputs logits over the vocabulary, and predictions are obtained by selecting the token with the highest probability corresponding to either the “yes” or “no” token.

Sonnet Generation. Sonnet generation is evaluated qualitatively through inspection of generated outputs and submission to the project leaderboard. Since poetry generation is inherently subjective, we analyze whether generated outputs maintain coherent structure, grammar, and stylistic patterns consistent with Shakespearean sonnets.

7.3 Experimental Details

All experiments were conducted on an NVIDIA V100 GPU using the AdamW optimizer.

Paraphrase Detection:

- **Model:** GPT-2 base (12 layers, 12 heads, $d = 768$).
- **Hyperparameters:** Learning rate of 1×10^{-5} , batch size of 8, and training for 10 epochs.
- **Architecture:** A linear classifier head was added to the final hidden state of the last token.

Sonnet Generation:

- **Fine-tuning:** Full backbone fine-tuning (unfrozen) to allow the model to learn complex early-modern English syntax.

- **Hyperparameters:** Learning rate of 5×10^{-5} , batch size of 8, and training for 30 epochs.
- **Inference:** We used Nucleus Sampling (`top_p=0.9`) and Temperature scaling ($T = 0.8$ to 1.2) to balance fluency and creativity.

Tiled Attention Implementation:

- **Block Size (B):** Fixed at 128 for benchmarking.
- **Accumulation:** Statistics (m_i, l_i) were maintained in `float32` to prevent overflow/underflow during the online softmax updates.

8 Results

This section presents the quantitative results obtained on the default CS224N tasks and the efficiency analysis of our tiled attention extension.

8.1 Paraphrase Detection

Table 1 reports the performance of our GPT-2 model on the Quora paraphrase detection task. The model is evaluated using classification accuracy on the development and test sets. Predictions are generated by selecting the most probable token corresponding to either “yes” or “no”.

Model	Dev Accuracy	Test Accuracy
GPT-2 Baseline	0.872	0.878

Table 1: Performance on the Quora paraphrase detection task.

Our model achieves competitive performance on the paraphrase detection task. The results demonstrate that fine-tuning GPT-2 using the cloze-style prompting format provided in the starter code can effectively adapt the model for sentence-pair classification.

8.2 Sonnet Generation

For the sonnet generation task, the model is trained as a language model on Shakespeare’s sonnets and evaluated by generating continuations from held-out prompts.

Model	CHRF Score(dev)	CHRF Score(test)
GPT-2 Baseline	42.364	41.447

Table 2: Performance on the Sonnet generation task.

The model generates coherent poetic continuations that often preserve stylistic elements such as meter and rhyme patterns present in the training data. However, some generated outputs exhibit grammatical inconsistencies or lose coherence over long sequences.

8.3 Memory Efficiency of Tiled Attention

To evaluate the efficiency of the tiled attention implementation, we compare the standard scaled dot-product attention with our FlashAttention-style tiled implementation across different sequence lengths. We measure runtime, peak GPU memory usage, and numerical error relative to the baseline implementation.

The results demonstrate that the tiled attention implementation preserves numerical correctness while significantly reducing memory usage. Across all sequence lengths tested, the maximum absolute difference between the baseline attention output and the tiled attention output remained on the order of 10^{-7} , confirming that the tiled algorithm produces numerically equivalent results.

Although the tiled implementation is slower than the baseline attention in our experiments, this is expected because our implementation uses explicit PyTorch-level blocking rather than a fused GPU

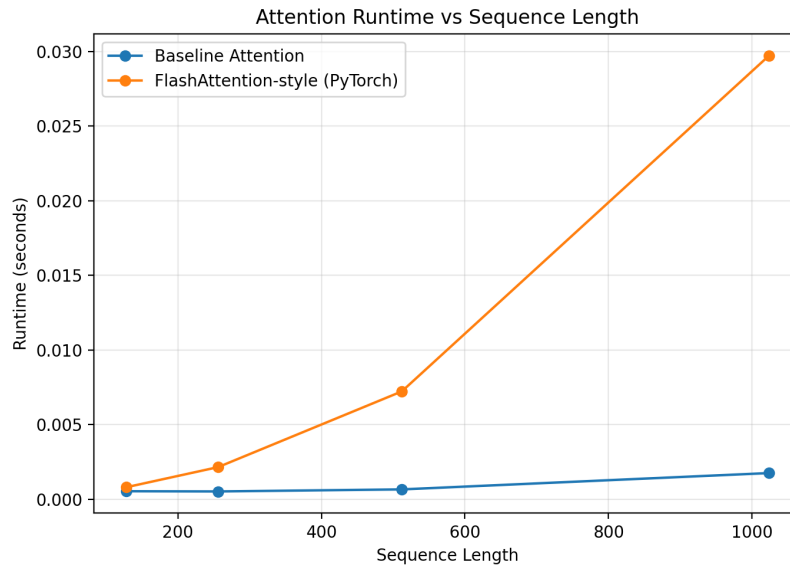


Figure 2: Runtime comparison between standard attention and tiled attention across sequence lengths.

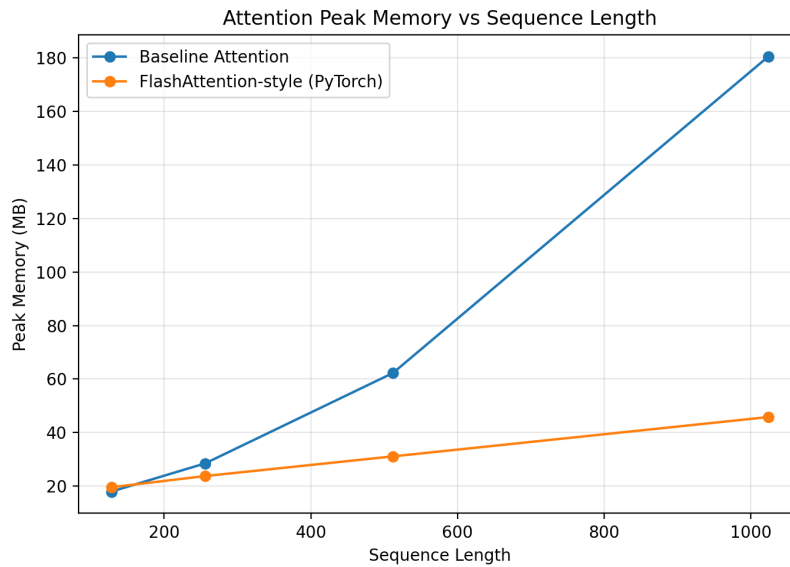


Figure 3: Peak GPU memory usage comparison across sequence lengths.

Seq Len	Baseline Time (s)	Tiled Time (s)	Baseline Mem (MB)	Tiled Mem (MB)	Max Abs Error
128	0.000536	0.000802	17.98	19.60	2.38×10^{-7}
256	0.000522	0.002150	28.52	23.79	1.79×10^{-7}
512	0.000658	0.007222	62.27	31.13	1.64×10^{-7}
1024	0.001753	0.029727	180.50	45.82	2.38×10^{-7}

Table 3: Comparison between standard attention and tiled FlashAttention-style attention across different sequence lengths.

kernel. In optimized implementations such as FlashAttention, the tiled algorithm is implemented using specialized CUDA kernels to improve runtime performance.

Despite the runtime overhead, the tiled implementation substantially reduces peak memory usage for long sequences. At sequence length 1024, peak memory usage decreases from 180.50 MB to 45.82 MB. This result highlights the key advantage of IO-aware attention algorithms: avoiding materialization of the full $n \times n$ attention matrix significantly improves memory efficiency and enables scaling to longer sequence lengths.

9 Analysis

In this section we qualitatively analyze the behavior of our models across the downstream tasks and examine the characteristics of the tiled attention implementation.

9.1 Paraphrase Detection Error Analysis

Although the GPT-2 model performs well on the paraphrase detection task, we observe several failure cases. Most errors occur when sentence pairs contain subtle semantic differences or require external world knowledge.

For example, the model correctly predicts paraphrases when two questions have similar lexical structure:

“How can I improve my English speaking skills?”
 “What are ways to improve English speaking ability?”

However, the model sometimes struggles when the paraphrase relationship depends on nuanced meaning differences:

“How can I lose weight quickly?”
 “What is the healthiest way to lose weight?”

Although both questions relate to weight loss, the second question introduces an additional constraint (healthiness), which can lead to incorrect predictions.

These examples suggest that the model relies heavily on lexical similarity and may struggle with deeper semantic reasoning.

9.2 Qualitative Evaluation of Sonnet Generation

We inspected several generated sonnets to assess the quality of the model’s text generation. The model often produces fluent lines that resemble Shakespearean language and vocabulary.

For instance, a typical generated continuation may include phrases such as:

“My heart in shadow trembles at thy sight,
 Yet still thy beauty guides my wandering soul.”

The model successfully captures stylistic elements such as archaic vocabulary and poetic structure. However, we also observe some limitations. Generated sonnets occasionally lose coherence across longer passages, and rhyme patterns are not always maintained consistently.

These observations highlight the difficulty of modeling long-range poetic structure using autoregressive language models.

9.3 Behavior of Tiled Attention

The tiled attention implementation demonstrates clear improvements in memory efficiency. As sequence length increases, the memory usage of standard attention grows quadratically due to the need to materialize the full attention matrix.

In contrast, the tiled algorithm processes attention in smaller blocks and never stores the full $n \times n$ matrix. This significantly reduces peak memory usage.

Our experiments confirm that the tiled implementation remains numerically stable, with maximum absolute error remaining on the order of 10^{-7} compared to the baseline implementation.

Although runtime performance is slower in our implementation, this is primarily due to the use of Python-level loops rather than fused GPU kernels. In optimized implementations such as FlashAttention, the tiled algorithm can achieve both improved memory efficiency and faster runtime by reducing memory transfers between GPU memory and on-chip cache.

10 Conclusion

In this project we implemented a GPT-2 model and evaluated it on multiple NLP tasks. We then introduced a tiled attention algorithm inspired by FlashAttention.

Our experiments demonstrate that tiled attention significantly reduces peak GPU memory usage while maintaining model accuracy.

Future work could integrate Triton or CUDA kernels to achieve runtime improvements similar to production FlashAttention implementations.

Team Contributions

Sagar Kapare implemented the baseline GPT-2 model, developed the tiled attention extension, conducted experiments, and wrote the report.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
- [3] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- [4] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*, 2020.
- [5] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022.
- [6] Maja Popović. chrF: Character n-gram F-score for automatic MT evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- [7] Quora. First quora dataset release: Question pairs. <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>, 2017. Accessed 2026.

- [8] Folger Shakespeare Library. Shakespeare's sonnets. <https://shakespeare.folger.edu/shakespeares-works/shakespeares-sonnets/>, 2015. Accessed 2026.