

Attention Modifications for Improved Adaptation

Stanford CS224N Default Project

Michael Jang

Department of Computer Science
Stanford University
jj277@stanford.edu

Jerry Yin

Department of Computer Science
Stanford University
jerryyin@stanford.edu

Abstract

Decoder-only transformers such as GPT-2 are often adapted to downstream tasks, but their practical deployment remains limited by issues in attention behavior, training and inference efficiency, and memory usage. We present a unified empirical study of several lightweight GPT-2 extensions motivated by these issues, including gated attention, selective attention, LoRA, and KV caching. We evaluate these methods on sentiment analysis, paraphrase detection, and Shakespeare-style sonnet generation. Our results show that these methods generally preserve or improve the baseline performance. Furthermore, gated attention qualitatively improves attention sink primarily with small gate value initializations, selective attention based pruning reduces KV cache memory by up to 60% with comparable model performance, LoRA reduces training wall time by roughly 32%, and KV caching improves decoding speed by $1.11\times$, yielding better time and space efficiency for training and inference.

1 Key Information to include

- TA mentor: Caroline Choi; No external collaborators, mentors, or sharing project.
- Late Days (if you are using any):

Team member	Late days remaining
Michael Jang	2
Jerry Yin	2
Total pooled	4
Team size	2
Late days applied for the report	4

2 Introduction

Transformer-based language models [1] have become a standard foundation for natural language processing as they perform well on discriminative and generative tasks. Decoder-only models such as GPT-2 [2] are especially appealing because the same backbone can be adapted to a variety of downstream settings. However, their strong empirical performance comes with other challenges such as quality over long contexts, training and inference time efficiency, as well as memory usage.

Multiple works highlight challenges in Transformer inference and adaptation. Notably, the attention sink phenomenon [3] occurs when the attention mechanism assigns large weight to the first token as an anchor under the softmax constraint. Abruptly masking or evicting this token can significantly degrade generation quality over long contexts. Autoregressive decoding also introduces a trade-off between memory and computation: caching keys and values [4] accelerates inference but increases memory usage. In addition, training and adaptation remain expensive, motivating parameter-efficient methods such as LoRA [5]. These issues raise challenges in attention allocation, efficiency, and model behavior in downstream tasks.

In this work, we study several lightweight extensions to GPT-2 motivated by these concerns. We investigate gated attention as a mechanism for modifying attention allocation and mitigating attention sink behavior, and selective attention [6] as a way to identify more informative tokens during decoding. We further explore whether these signals can be used for cache pruning, while also evaluating LoRA for training efficiency and KV caching for inference speed.

We evaluate these methods on three tasks: sentiment analysis as a sanity check and proof-of-correctness setting, paraphrase detection as a downstream discriminative benchmark, and Shakespeare-style sonnet generation as a generative benchmark for comparing model behavior and decoding strategies. Across these tasks, we find that gated and selective attention generally match or improve baseline GPT-2 performance. In sonnet generation, both attention variants improve generation quality in chrF score under our evaluation setup, and gated attention qualitatively reduces attention sink behavior in attention visualizations. Through our LLM-as-a-judge evaluation, We further find that selective-attention-based cache pruning substantially outperforms simple baselines such as uniform pruning when beginning and most recent tokens are preserved, and can retain comparable generation quality at larger cache budgets while reducing cache size by around 60.27%. By contrast, gated-attention-based cumulative-attention pruning performed worse than the base model. Finally, we observed that LoRA reduces training wall time by roughly 32%, while KV caching improves inference speed by roughly 1.11 times in tokens generated per second. Our results suggest that lightweight attention modifications can preserve and improve performance while also yielding practically useful signals for efficient inference strategies. Since our designated tasks do not directly involve long-context benchmarks, we present these findings as evidence of improved attention behavior and pruning utility rather than definitive proof of performance gains.

3 Related Work

3.1 Transformer Models and Decoder-only Language Modeling

Transformer-based language models have become a dominant paradigm in natural language processing due to their ability to learn contextual token representations through self-attention. Among them, decoder-only models such as GPT-2 [2] are designed for autoregressive language modeling, where each token is predicted conditioned on all previous tokens in the sequence.

The hidden states produced by such models can also serve as contextualized representations of the input sequence. In generation tasks, these representations are used directly for next-token prediction. In discriminative tasks such as sentiment analysis or paraphrase detection, final-layer hidden states can instead be passed to a lightweight classification head for downstream prediction. This flexibility makes GPT-2 a convenient unified backbone for both generative and classification tasks.

Given the wide applicability of decoder-only Transformers, the majority of follow-up work explored ways to improve the attention behavior, training and inference efficiency, and memory usage. Here, we treat the pretrained vanilla GPT-2 model [7] as our baseline and investigate these three directions.

3.2 Methods for Improving Attention Behavior

Sink Attention [8] modifies the softmax denominator by adding a learnable sink parameter such that unused attention weights are directed to other tokens on GPT-OSS family [9]. Gated Attention [10] appends head-specific gate parameters to scale attention outputs to modulate weight across heads of 1.7B dense model and a 15B Mixture of Experts (MoE) model with 3.5T high-quality tokens. Selective Attention [6] reduces attention paid to unneeded context elements through a parameter-free modification by treating an attention head as a "forget" gate, from the observation that irrelevant tokens in the attention context can degrade model quality, and the authors show improvements in both language modeling and downstream tasks across a range of model sizes and context lengths.

While these methods have shown promise in larger-scale settings, it remains unclear how well they transfer to a smaller pretrained GPT-2 model on downstream tasks. We study this setting directly by evaluating whether gated and selective attention preserve or improve performance on extended tasks.

3.3 Methods for Improving Training Efficiency

For improving training efficiency, Parameter-Efficient Fine-Tuning (PEFT) methods are widely considered. For example, adapters [11] add a small bottleneck function consisting of feed-forward

down-projection and up-projection placed after the multi-head attention (MHA) and/or after the feed-forward layer such that the frozen pretrained model can adapt to a downstream task efficiently. Similarly, Prefix-Tuning [12] freezes all pretrained parameters and adds a learnable prefix of parameters, which are sequences of continuous task-specific vectors. Most notably, LoRA [5] adds learnable low-rank matrices to the fixed original weights, building on the finding that model weights have low intrinsic dimensionality [13]. Specifically, fine-tuning occurs in a much smaller parameter subspace that can be learned with low-rank matrices.

3.4 Methods for Improving Inference Efficiency

Key-value (KV) caching stores previously computed keys and values during decoding step to reduce redundant computation during autoregressive generation and has become a standard component of efficient Transformer inference [14]. However, caching also increases memory usage as the KV cache scales with the number of processed tokens. This has motivated a large body of work on cache compression, eviction, and pruning. Some methods are based on the observation that certain tokens are consistently more important to retain than others. For example, H₂O retains both recent tokens and “heavy hitters” identified from cumulative attention scores [15], while SnapKV uses an observation window near the end of the prompt to identify important prefix positions for each attention head [16]. More broadly, the work suggests that beginning and recent tokens could be important to preserve, and that attention-derived scores could be useful signals for selecting additional tokens to retain. Selective Attention is also relevant from a space-efficiency perspective, where the special attention head design directly supports an importance signal. Leviathan et al. [6] show that it can support smaller attention buffers during inference while maintaining comparable validation performance. This is appealing because the selective head provides token-importance signals at each layer, making it a natural candidate for cache pruning in a smaller pretrained GPT-2 model.

4 Approach

4.1 Vanilla GPT-2 Setup and Downstream Tasks

We use pretrained GPT-2 [2] as the common backbone for all experiments. GPT-2 is a decoder-only Transformer that produces contextualized hidden states through causal self-attention over the input sequence. Given tokenized input $\mathbf{x} = (x_1, \dots, x_T)$, the model outputs final-layer hidden states $\mathbf{H} \in \mathbb{R}^{T \times d}$, which we fine-tune for each downstream task.

For discriminative tasks, we use the final hidden state of the last token as a sequence representation and feed it to a task-specific linear classification head. For sentiment analysis, this head predicts one of five sentiment labels. For paraphrase detection, we format the two input sentences as a natural-language prompt and use a binary classification head to predict whether the pair is a paraphrase. In both cases, the model is trained with cross-entropy loss.

For sonnet generation, we fine-tune GPT-2 as a standard autoregressive language model without adding a task-specific head. The final hidden states are projected to vocabulary logits using the tied token embedding matrix $\mathbf{E}_{\text{tok}} \in \mathbb{R}^{|\mathcal{V}| \times d}$, and the model is trained with the next-token prediction objective: $P(x_{t+1} \mid x_{\leq t}) = \text{softmax}(\mathbf{H}_t \mathbf{E}_{\text{tok}}^\top)$. At inference, we decode autoregressively using Hugging Face’s `generate()` call with temperature scaling, followed by top- k filtering and nucleus (top- p) sampling. All extensions studied in this work are implemented on top of this shared GPT-2.

4.2 Extension Details

4.2.1 Gated Attention

We implemented the optimal configuration by Qiu et al. [10], a multiplicative head-specific element-wise sigmoid gate after SDPA. After the outputs from the MHA are concatenated, prior to passing through the dense layer, the input Y is modulated using the following formulation: $Y' = Y \odot \text{softmax}(HW_\theta)$, where H is the hidden states after pre-normalization, and W_θ is the learnable weights of the gate. Following our initial analysis on the sentiment classification task, we tested with the gate’s bias initialized to a large positive value to force an initial gate value near 1.0, namely identity initialization. We hypothesized that the model would initially pass all information and suppress the attention sink gradually to stabilize training. To directly target the sink, we tested gating only on the first token with both the identity initialization and uniform random initialization.

4.2.2 Selective Attention

We implement Selective Attention [6], which reuses the logits of one existing attention head to form a selection matrix $S \in \mathbb{R}^{T \times T}$, where $S_{i,j}$ measures how much token x_i masks token x_j . If $A^{(h_0)}$ denotes the pre-softmax attention logits of the designated selective head h_0 , then $S = \text{ReLU}(A^{(h_0)})$. This adds no new parameters - only negligible extra computation. The selection matrix is then accumulated into a masking matrix $F \in \mathbb{R}^{T \times T}$, where $F_{i,j} = \sum_{k \leq i-1} S_{k,j}$, so that $F_{i,j}$ represents how much token x_j has been masked by all preceding tokens up to position i . The masking matrix is subtracted from the standard attention logits before the softmax: $\text{SelectiveAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} - F\right)V$. Intuitively, once a token has been sufficiently attended to by intermediate tokens, its contribution can be reduced for later positions. Since our model is already pretrained and each head might already have retained information, we introduce two new hyperparameters, the number of layers with selective attention and which head such mechanisms are implemented on.

4.2.3 Low-Rank Adaptation

We implement LoRA by adding trainable rank decomposition matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ with the rank $r \ll \min(d, k)$ for query and value projection. Specifically, $W'_Q = W_Q + \Delta W_Q$ and $W'_V = W_V + \Delta W_V$ where $\Delta W = \alpha BA$. α is a hyperparameter that controls the tradeoff between pre-trained knowledge and task-specific knowledge.

4.2.4 KV-Caching and Cache Pruning

At inference time, we use standard key-value (KV) caching to avoid recomputing keys and values from previous tokens at every decoding step. This speeds up autoregressive generation, but the cache grows linearly with context length and can become a significant memory bottleneck.

We implemented two pruning strategies together with a simple retention baseline. First, a cumulative-attention pruning inspired by H₂O [15]. Let $\alpha_{i,j}$ denote the attention assigned to cached token x_j at decoding step i . We define the cumulative attention score of token x_j as $c_j = \sum_i \alpha_{i,j}$ where the sum is taken over decoding steps after x_j enters the cache. Tokens with larger cumulative attention among each layer are treated as more important to retain under a fixed cache budget per layer.

The second strategy uses the masking signal produced by Selective Attention. Recall that Selective Attention defines a masking matrix $F \in \mathbb{R}^{T \times T}$ that measures how much token x_j has been masked before query position i . We convert this into a pruning score by aggregating masking over decoding steps, $m_j = \sum_i F_{i,j}$, and treat more heavily masked tokens as less important to retain. Thus, we evict tokens with larger masking scores under a fixed cache budget. We also construct a simple uniform-pruning baseline that uniformly samples from the cache to satisfy the same cache budget.

5 Experiments

5.1 Data

For sentiment analysis, there are two datasets: 1) Stanford Sentiment Treebank (SST) dataset with 11,855 single-sentence movie reviews with 8,544 training samples, 1,101 validation samples, and 2,210 testing samples, 2) CFIMDB dataset with 2,434 polar movie reviews, with 1,701 training samples, 245 validation samples, and 488 testing samples. For paraphrase detection, the Quora dataset contains 400,000 question pairs with binary labels. There are 141,506 train, 20,215 validation, and 40,431 test samples. For sonnet generation, the Sonnet Dataset contains 154 sonnets by Shakespeare. There are 14 lines per sonnet, with 143 samples for training and 12 samples for testing.

5.2 Evaluation method

5.2.1 Performance Evaluations

For sentiment analysis and paraphrase detection, accuracy is used. For sonnet generation, we utilized the chrF score to measure the character-level similarity. Furthermore, since we observed that sometimes generations with inconsistent themes or collapse formally achieved a high chrF score, we utilize LLM-as-a-judge method using a GPT-5 model to rank the sonnets from different model configurations. Note that both ranking and rating are used for evaluation because generated sonnets

frequently receive uniformly low LLM scores, making rating-based comparisons alone insufficient to distinguish model outputs. See Appendix B for the prompt. This performance evaluation will apply to comparing model configurations for all investigations under sonnet generation.

5.2.2 Efficiency Evaluations

For speed, the effect of LoRA will be compared via wall time, and KV-caching will be evaluated by the difference between the number of tokens generated with and without caching. We also investigate memory saving by reporting peak and average cache size for different cache token budgets in MB.

5.2.3 Attention Sink Evaluation

For gated attention, we conduct a qualitative analysis of the model’s layer and head-wise attention weight heatmaps to monitor the attention sink phenomenon under the three tasks.

5.3 Experimental details

5.3.1 Sentiment Analysis

We performed a grid search for hyperparameters tuning with five different learning rates with a fixed batch size of 128 and dropout rate of 0.3 for 20 epochs: 10^{-3} , 5×10^{-4} , 10^{-4} , 5×10^{-5} , 10^{-5} . The vanilla GPT-2 model was trained with both last layer and full model training, and we also tested multiplicative head-specific elementwise sigmoid gate (HS) after the SDPA output.

5.3.2 Paraphrase Detection

We tuned hyperparameters for paraphrase detection over learning rate, batch size, and training configuration, subject to available compute. For the vanilla GPT-2 model, we searched over learning rates 10^{-5} , batch sizes 16, 32, 64, and trained up to 20 epochs with early stopping. For LoRA, we searched over ranks 4, 8, 16 (with $\alpha = 2r$) and learning rates 10^{-4} , 3×10^{-4} , 5×10^{-4} at batch size 16. For selective attention, we varied the number of converted layers (all 12 vs. top 6) and the number of selection heads ($h \in 0, 10, 12$). Based on validation performance, we selected a learning rate of 10^{-5} , batch size 16, and 20 training epochs for the full fine-tuning baseline. We also choose the selective attention extension with head 0 and full layers.

Due to computational limits, we did not perform a separate full grid search for every extension. Instead, for gated-attention variants we reused the best hyperparameter setting found for vanilla GPT-2 on this task: learning rate 10^{-5} , batch size 16, and 20 epochs. We then evaluated multiple gated-attention configurations including multiplicative head-specific elementwise sigmoid gate with identity initialization, first token identity initialized gating, and first token uniformly random initialized gating. All experiments were run in parallel on an NVIDIA L4 24GB and a shared NVIDIA A100 80GB, with the L4 primarily used for gated-attention runs and the A100 used for vanilla and selective-attention runs. All the training curves are shown in Figure 9 in Appendix B.

5.3.3 Sonnet Generation

For sonnet generation, we chose a default set of hyperparameters of learning rate 10^{-5} , batch size 16, and 30 epochs with early stopping by cross-entropy loss on the full validation set sonnets. We investigated different configurations of selected attention similar to the paraphrase task, and chose the best performing (by chrF) configuration with selective head 10 on the top 6 layers. We also experimented with first token identity initialized gating. These three serve as the models we experiment with for the following paragraphs. We also experimented with the generation configurations with different top- k filters and p nucleus sampling values and determined the best configuration through chrF and human evaluations, which leaves out the top- k filtering (possibly due to the creative nature of the task) and $p = 0.9$. Besides the chrF score, we also perform an LLM as a judge ranking.

To analyze the effect of KV-caching, we generated validation set sonnets with and without our KV-cache implementation on all three models fixing other configurations while recording the wall time during each generation. We prove correctness of our implementation from the identical trajectories for each model and count the number of tokens generated to calculate the generation speed (num_tokens/s) with and without caching. This experiment runs on the NVIDIA A100 80GB GPU.

We then conduct four experiments to investigate the effect of cache pruning (see 4.2.4 for details) using the fixed generation method mentioned above and an LLM judge. First, we show the effectiveness of selective head pruning against the baseline for cache token budgets 48, 64, and 96 for our selective attention model. Next, we compare the generations with different cache budgets (32, 48, 64, 96, no_limit). Third, we experiment the effect of preserving different beginning (0, 8, 32 while preserving the last token in the end) or most recent (1, 8, 32 while preserving 16 tokens in the beginning) tokens on a cache budget of 64. Finally, we apply the cumulative attention pruning method and compare the performance between the base and gated attention model.

6 Results

6.1 Sentiment Analysis

Dataset	Model	Setting	Loss	Train Acc	Dev Acc
SST	Vanilla GPT-2	Last Layer	1.307	0.493	0.468
		Full FT	1.045	0.595	0.510
	Gated (HS)	Full FT (10^{-5} LR)	1.150	0.560	0.481
CFIMDB	Vanilla GPT-2	Last Layer	0.398	0.874	0.873
		Full FT	0.033	0.998	0.980
	Gated (HS)	Full FT (10^{-5} LR)	0.119	0.985	0.963

Table 1: Sentiment Analysis: Vanilla GPT-2 vs. Head-Specific (HS) Gated Attention.

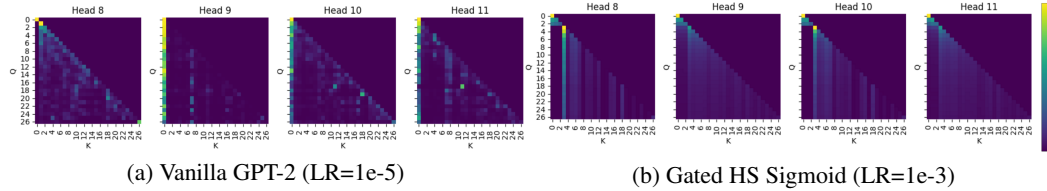


Figure 1: Attention weights for Head 8-12 at Layer 12 during full-model training on CFIMDB. All heads are shown in Figure 3 in Appendix A.

For the sentiment task, layer-12 attention weights trained on the CFIMDB dataset are visualized in Figure 1. The vanilla GPT-2 baseline (left) shows a clear attention sink, with significant attention mass allocated to the first token. With a head-specific sigmoid gate trained with a larger learning rate, the attention is no longer concentrated on the first token. An interesting phenomenon is observed, where the attention is allocated to specific tokens instead of the first. However, it causes severe representation collapse, with each head showing smoothed attention across all heads. The empirical results also reflect these results, with a decrease in validation accuracy from 0.963 to 0.882 for 10^{-3} LR. Lower learning rate maintains comparable accuracy, but the attention sink was not resolved.

6.2 Paraphrase Detection

In terms of performance, the gated attention model slightly improves over the baseline on the test set. In contrast, models incorporating selective attention achieve higher validation accuracy but exhibit reduced performance on the test set, potentially indicating overfitting. Regarding training efficiency, LoRA reduces training time to approximately 15 minutes per epoch compared to 22 minutes for full-parameter fine-tuning. However, due to the gap in validation accuracy and the relatively small size of the other downstream tasks, we adopt full-parameter fine-tuning for the remaining experiments.

Table 2: Paraphrase detection model performance on Dev and Test sets in accuracy.

	Base	Gated-Attn	Sel-Attn	Gated+Sel-Attn
Dev Accuracy	0.898	0.899	0.901	0.897
Test Accuracy	0.871	0.881	0.861	0.796

Specifically for gated attention extensions, we observe that first-token gating with uniform initialization achieves the best performance (dev accuracy 0.899, test accuracy 0.881), outperforming both dev accuracies of head-specific gating (0.893) and first-token gating with identity initialization (0.895). The vanilla GPT-2 baseline shows a clear attention sink. However, identity initialization for both the standard gate and the first token gate experiments produces near-identical attention weights as shown

in Figure 5 in Appendix A. With uniformly random initialization, the attention sink is resolved for the heads that started training with a low gate initialization. Importantly, we identified that the gate value does not change over epoch, as shown in Figure 8.

6.3 Sonnet Generation

Model performance is evaluated using both chrF scores and LLM-judge rankings. The proposed extensions match or improve performance on both metrics compared to the baseline shown in 3, with the combined model achieving the best results.

Table 3: LLM judge eval on dev set (mean $\pm$ sd) and chrF on test set

Config	Score	Rank	chrF
Baseline	2.58 ± 1.32	2.25 ± 1.09	41.60
Sel-Attn (top6, $h=10$)	2.50 ± 1.12	2.25 ± 1.16	42.18
Gated (First Token Identity)	2.83 ± 1.21	2.50 ± 1.12	41.58
Gated+Sel (top6, $h=10$)	3.33 ± 1.11	3.00 ± 0.91	42.51

Due to the sparsity of attention weights, visualizations of the attention plots are shown in Appendix A Figure 6. Qualitatively, first token identity initialized gating results in a better distribution of attention weights, especially for heads 0, 1, 3, and 10, with brighter activation away from the first column.

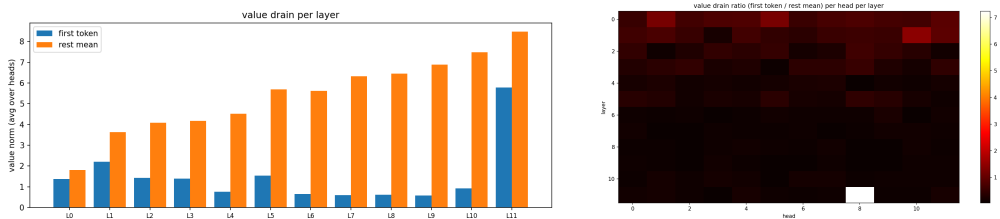
For KV-caching, we observe an average $1.11\times$ increase in inference speed, with 92.9 tokens/s compared to 83.7 tokens/s without caching.

For pruning, we observe that selective-attention pruning consistently outperforms random eviction across all evaluated cache budgets (Table 5). Furthermore, generation quality under different cache budgets is summarized in Table 6. As the cache budget decreases, generation quality declines. To our surprise, budget 96 outperforms while reducing peak cache usage by 40.4% relative to the no-pruning configuration. Third, the effect of protecting prefix and suffix tokens during pruning is shown in Tables 7 and 8. Protecting fewer prefix tokens yields higher scores, while increasing the number of protected suffix tokens results in improved generation quality. Lastly, we find the LLM judge consistently preferring the base model over the gated attention variation for all cache budgets.

7 Analysis

7.1 Gated Attention

Our experiments suggest that initializing some gates with small values introduces a forcing mechanism that can suppress attention sink behavior and redistribute attention mass during training, leading to a slight performance gain. However, we claim this mechanism is significantly less effective and thus does not resolve the sink for every head when applied to a pretrained model. Research



(a) Value ratio (1st token to rest mean) across layers (b) Value ratio (1st token to rest) across heads and layers

Figure 2: Value drain of Pretrained Vanilla GPT-2 Model

shows that the model learns to mitigate the effect of attention sink by assigning a small norm to the value vector of the sink token during pretraining, referred to as value drain [17] [18]. Thus, sink contributes significantly less to the layer output, but the remaining attention to the meaningful tokens is still lacking due to the sink. Such implicit gating mechanism for pretrained LLMs acts as a stable equilibrium state, which assigns high attention weight to the sink token but contributes negligible signal to the residual stream. Therefore, gating the first token for a pretrained model does not noticeably change the model output. This incentivizes the model to consider alternative paths, such as the MLP layer, to adjust the weights for the downstream task, contributing to unchanging gate values from initialization. In other words, the fundamental issue lies in the fact that value drain

emerges as a learned mitigation to the attention sink. Figure 2 shows that the vanilla GPT-2 model exhibits value drain except head 8, which is the only head that does not have any attention sink, providing further evidence that value drain acts as an implicit gate for models with attention sink as shown in Figure 4. We also observe such phenomenon for the fine-tuned gated model as shown in Figure 7, showing that value drain cannot be resolved even after gating for downstream tasks.

Furthermore, we hypothesize that even though the attention sink issue is well resolved for sonnet generation, changing how much an attention head contributes to the residual stream does not necessarily suggest whether its attention weights are a reliable importance signal for cache retention. On the contrary, since the GPT-2 model was pretrained with the sink-related behavior and value drain already established, gating may have weakened the connection between cumulative attention and actual token utility and created adversary effects on cumulative-attention pruning.

7.2 Selective Attention

First, we observe that selective attention appears more effective in generative settings than in paraphrase detection. This may imply that stronger context filtering may not generalize as well to tasks requiring fine-grained semantic comparison across the full input. Furthermore, extending the model with both gated and selective attentions might have induced too much complexity and therefore resulted in under-performance on the test set.

The pruning experiments provide evidence that the selective heads are meaningful rather than arbitrary. The first experiment against the baseline indicates that the masking signal correlates with which tokens can be removed with limited quality degradation. On the second experiment, a moderate cache budget outperforming the no-pruning setting suggest that selective head may be helpful removing redundant context on a moderate cache budget. The prefix/suffix ablations further indicate that selective attention is better at identifying redundancy in older context, since preserving suffix tokens is consistently more beneficial than preserving additional prefix tokens.

Furthermore, figure 10 also visualizes different layers of the first 20 token positions for sonnet 3 to show the effectiveness. It can be observed that majority of the heads tend to suppress only a small subset of early tokens, which might have been absorbed well into context. Furthermore, positions at less semantically meaningful words such as prepositions are more likely to receive strong signals.

8 Conclusion

We studied GPT-2 modifications for improving adaptation, efficiency, and attention behavior across sentiment analysis, paraphrase detection, and sonnet generation. Gated attention improved performance on paraphrase detection and sonnet generation, and sometimes mitigated attention sink qualitatively. Besides boosting the performance on sonnet generation, selective attention produced useful signals for substantially reducing cache usage while maintaining, if not improving, generation quality. Furthermore, LoRA and KV caching improved training and inference speed respectively.

However, the gains are not consistent across tasks. Gated attention is less effective in pretrained GPT-2, likely due to value drain, and selective attention performs better in generation than in paraphrase detection. Due to the nature of our downstream tasks, the results suggest practical utility that could be evaluated by future works with longer context and more complex tasks.

Team contributions (Required for multi-person team)

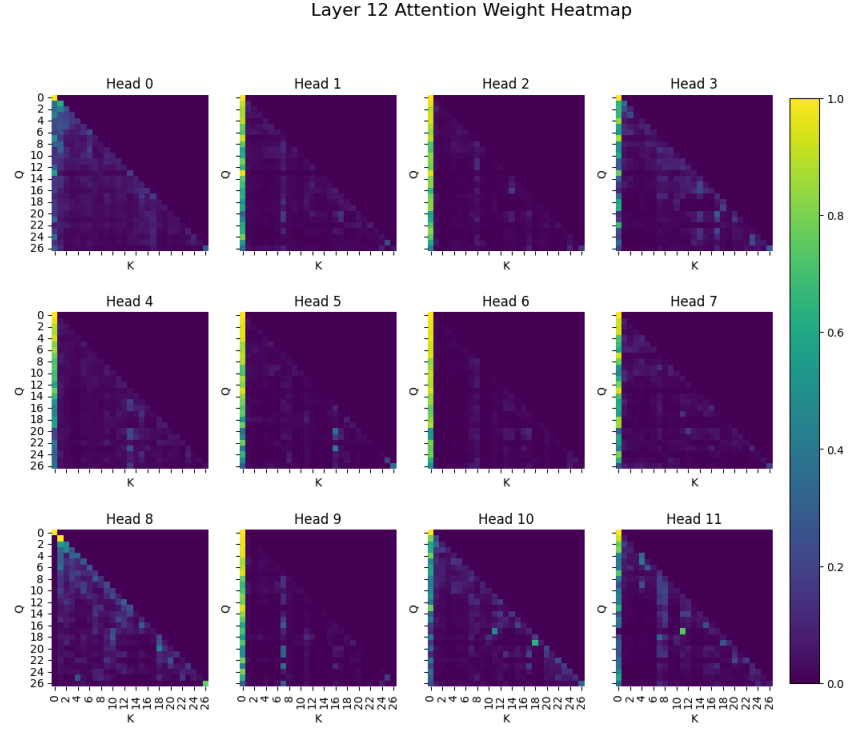
Michael: GPT base, sentiment analysis, gated attention implementation, experiments, and analysis.
 Jerry: paraphrase hyperparameter search, sonnet generation, LoRA, selective attention, KV-cache, cache pruning implementations, experiments, and analysis.

References

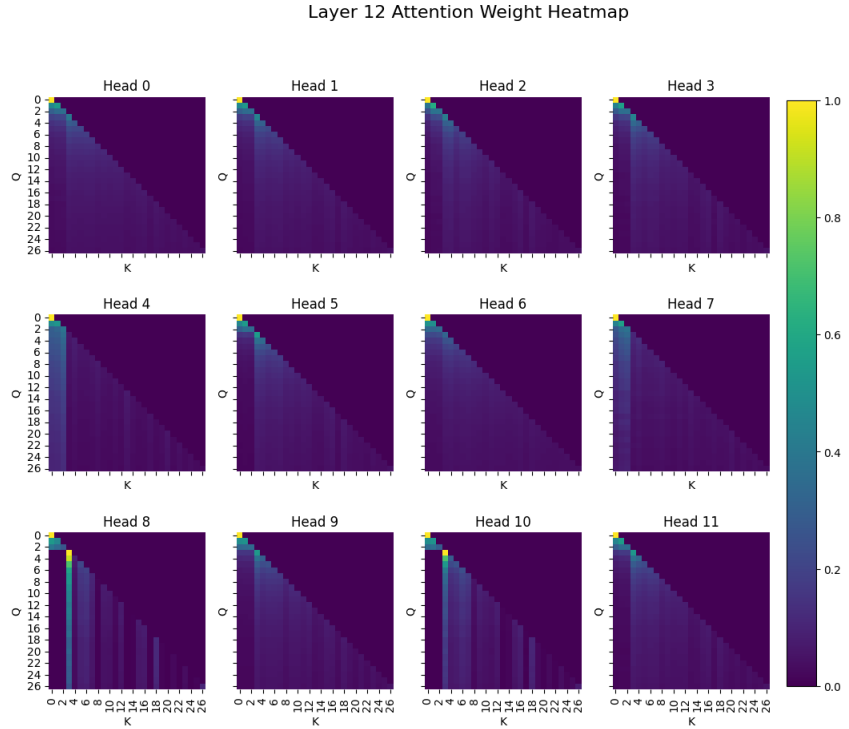
- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. Accessed: 2024-11-15.

- [3] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.
- [4] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference, 2022.
- [5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [6] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Selective attention improves transformer, 2025.
- [7] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.
- [8] Rishabh Agarwal, Mark Chen, Aditi Gupta, et al. Gpt-oss: Eliminating attention sink with sink-aware softmax. *arXiv preprint arXiv:2503.XXXXX*, 2025.
- [9] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, et al. gpt-oss-120b & gpt-oss-20b model card, 2025.
- [10] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.
- [11] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mohammad Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning (ICML)*, 2019.
- [12] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- [13] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 7319–7328. Association for Computational Linguistics, 2021.
- [14] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *ArXiv*, abs/2211.05102, 2022.
- [15] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H₂O: Heavy-hitter oracle for efficient generative inference of large language models, 2023.
- [16] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation, 2024.
- [17] Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. *arXiv preprint arXiv:2410.13835*, 2024.
- [18] Zizhuo Fu, Wenxuan Zeng, Runsheng Wang, and Meng Li. Attention sink forges native moe in attention layers: Sink-aware training to address head collapse. *arXiv preprint arXiv:2602.01203*, 2026.

A Gated Attention



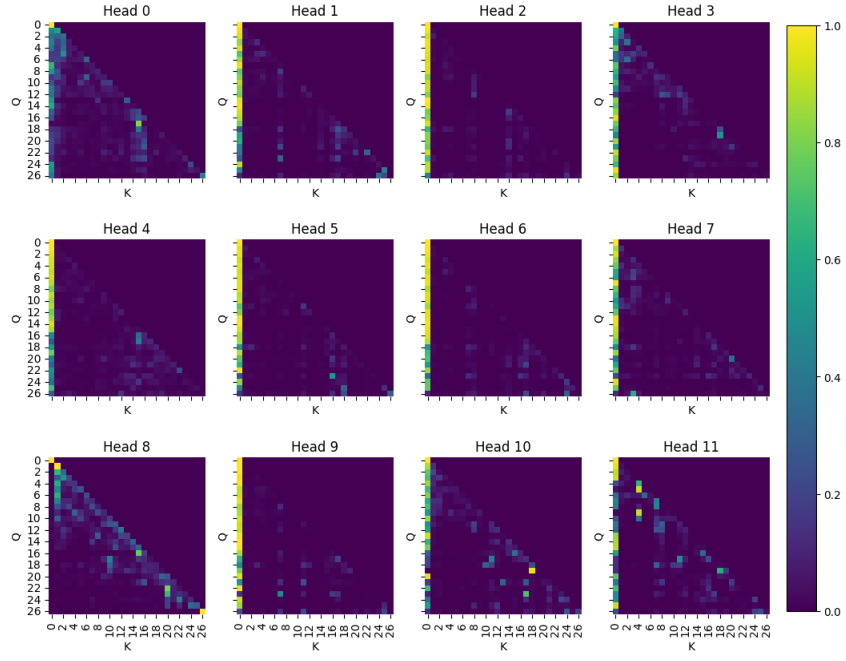
(a) Vanilla GPT-2 (LR=1e-5)



(b) Gated HS Sigmoid (LR=1e-3)

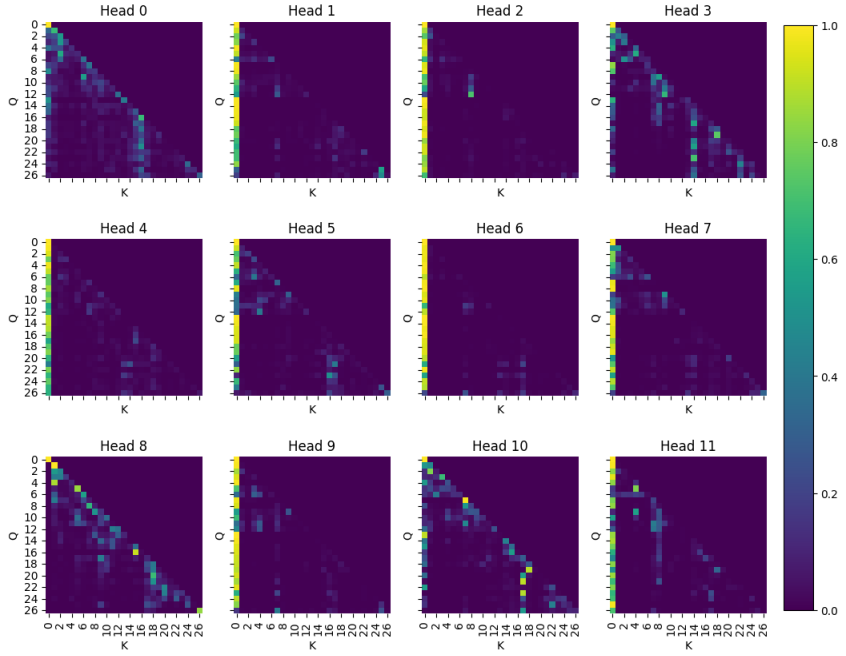
Figure 3: Attention weights at Layer 12 during full-model training on CFIMDB. A multiplicative head-specific elementwise sigmoid gate is used for Figure (b). We see that attention sink is mitigated for most of the heads (with some sinks moving location to token 4 as in head 8 and 10), but the attention collapses across the heads, leading to an over-smoothed attention.

Layer 12 Attention Weight Heatmap



(a) Vanilla GPT-2

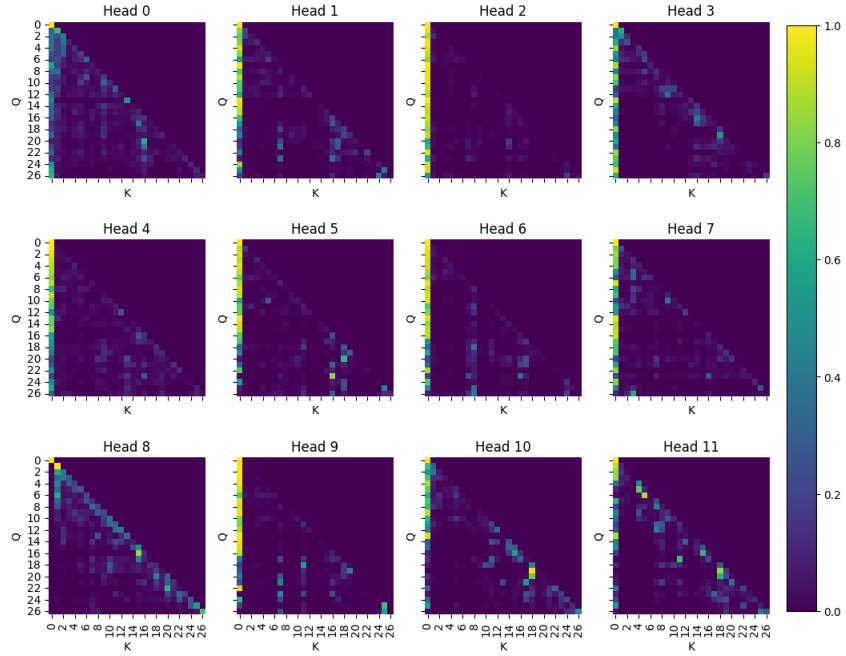
Layer 12 Attention Weight Heatmap



(b) Gated First Token, Uniform Random Initialization

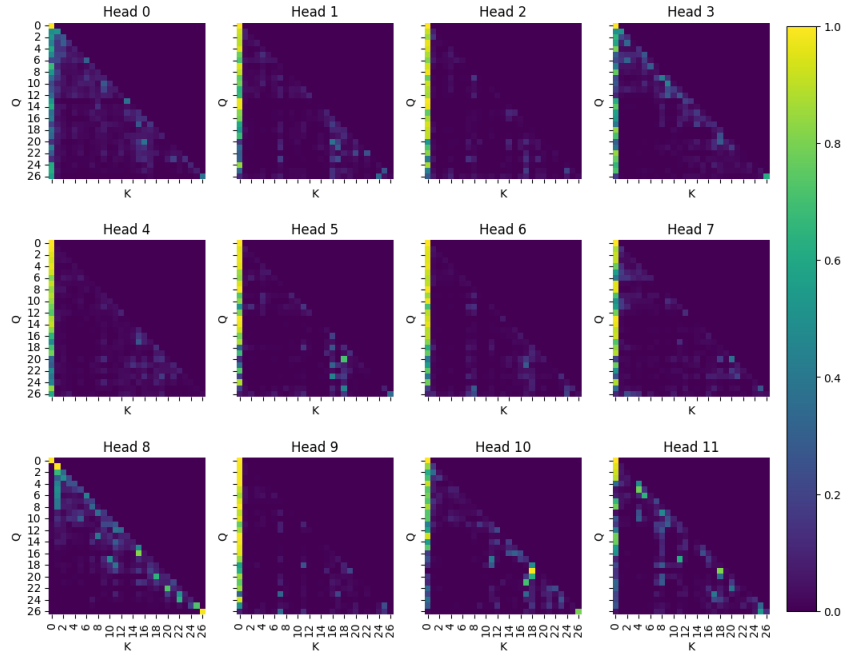
Figure 4: Attention weights at Layer 12 during full-model training on Quora Dataset. Only head 8 does not have attention sink for the vanilla model. First token gate values for (b) are in Table 4.

Layer 12 Attention Weight Heatmap



(a) Gated HS, Identity Initialization

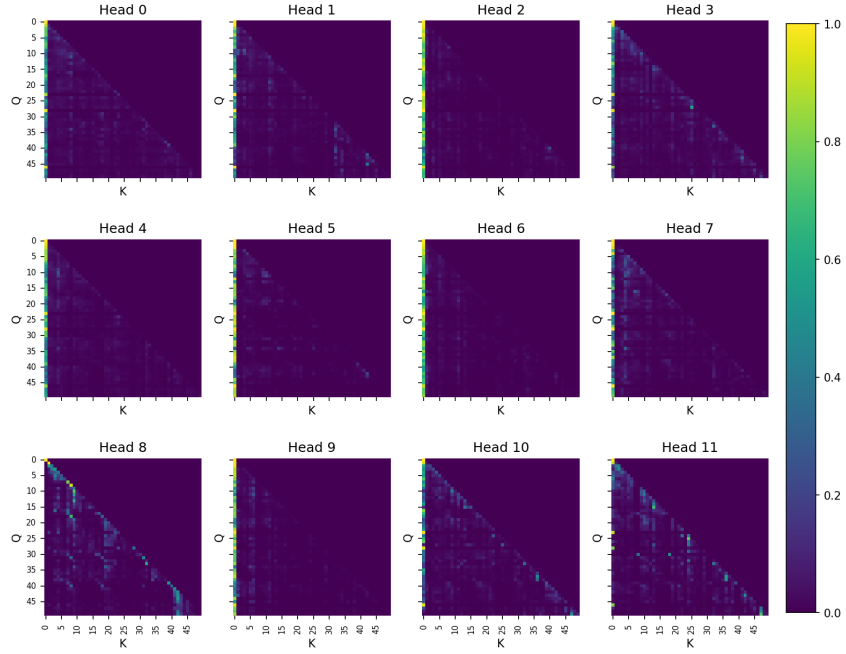
Layer 12 Attention Weight Heatmap



(b) Gated First Token, Identity Initialization

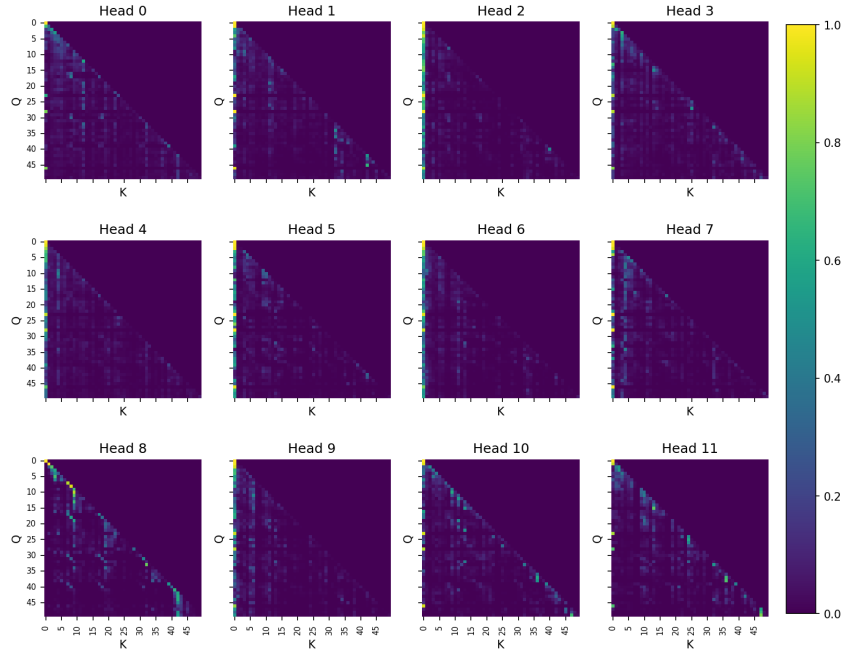
Figure 5: Attention weights at Layer 12 during full-model training on Quora Dataset. We see that identity initialization does not solve the attention sink issue. Only head 8 showing no attention sink, which is the exact same case for the vanilla GPT-2 model trained on the same dataset in Figure 4.

Layer 11 Attention Weight Heatmap



(a) Vanilla GPT-2

Layer 11 Attention Weight Heatmap



(b) Gated First Token, Identity Initialization

Figure 6: Attention weights at Layer 12 during full-model training on Sonnet Dataset. Comparing the two plots, we see that identity initialization on the first token leads to a better distribution of token weights by mitigating attention sink especially for heads 0, 1, 3, and 10.

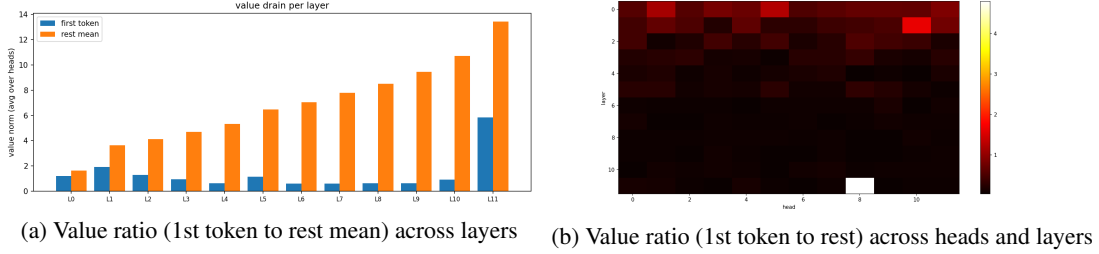


Figure 7: Value drain of Finetuned First Token Gated GPT-2 Model

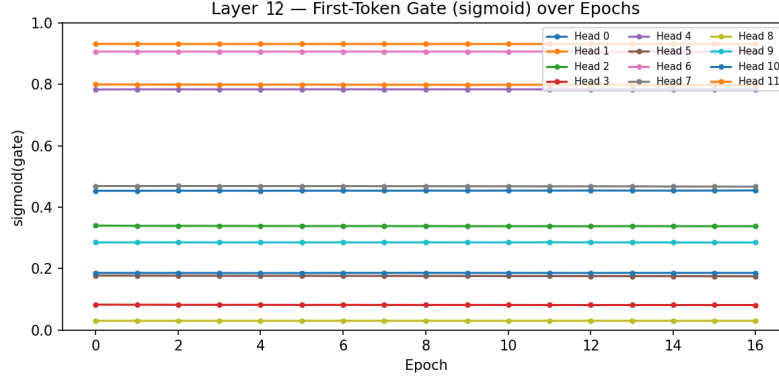


Figure 8: First Token Uniform Random Initialized Gate. The model barely changes the gate behavior over epochs.

Head	Gate	Head	Gate	Head	Gate
0	0.46	4	0.78	8	0.03
1	0.80	5	0.18	9	0.29
2	0.34	6	0.90	10	0.18
3	0.08	7	0.47	11	0.93

Table 4: Gate values for the first token per attention head in Layer 12.

B LLM Prompt

The following prompt was used to instruct the LLM judge when evaluating generated sonnets.

System Prompt

You are a literary critic and poetry expert.
 You will evaluate machine-generated Shakespeare-style sonnets.
 Each sonnet was generated by a language model conditioned on the first three lines of an original Shakespeare sonnet.
 The model then generated the remaining lines to complete the sonnet.

For each sonnet:

- Give an overall score from 1-10 (may use decimal points to distinguish the quality and use full range of scores), considering coherence (foremost), adherence to sonnet format, and poetic value.
 - Provide brief reasoning for the score.
 - Do not assign the same score to any two sonnets.
- Also provide a summary comparing all sonnets, highlighting key differences.

User Prompt Template

Shakespeare Sonnet #{sonnet_num}

Original first three lines (given as prompt to the model):

```
"""
{first_three_lines}
"""
```

Below are {N} different model-generated completions.
Please compare and score each one.

[A] {sonnet_A}

[B] {sonnet_B}

[C] {sonnet_C}

C Paraphrase Detection Hyperparameters Tuning

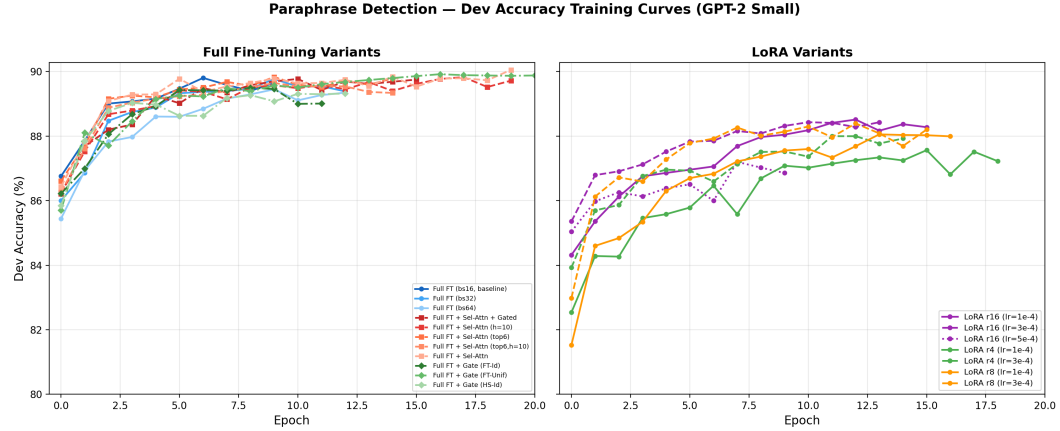


Figure 9: All training curves for the paraphrase detection task.

D Sonnet Generation results

Table 5: Selective-attention pruning vs. random eviction

Budget	Score (1–10)		Rank (1–2)	
	Sel-Attn	Random	Sel-Attn	Random
48	2.00 ± 0.82	1.42 ± 0.49	1.67 ± 0.47	1.33 ± 0.47
64	2.00 ± 0.71	1.33 ± 0.47	1.75 ± 0.43	1.25 ± 0.43
96	2.00 ± 0.41	1.25 ± 0.60	1.83 ± 0.37	1.17 ± 0.37

Table 6: Selective-attention pruning at different cache budgets

Configuration	Score	Rank	Peak Cache (KB)	Reduction
No pruning	3.50 ± 1.71	3.25 ± 1.42	11624	—
Budget 96	3.75 ± 1.69	3.42 ± 1.38	6926	40.4%
Budget 64	3.33 ± 1.37	3.17 ± 1.28	4617	60.3%
Budget 48	2.83 ± 1.21	2.75 ± 1.23	3463	70.2%
Budget 32	2.50 ± 1.44	2.42 ± 1.50	2477	78.7%

Table 7: Prefix protection

b	Score	Rank
0	2.67 ± 0.94	2.33 ± 0.75
8	2.08 ± 0.76	1.83 ± 0.90
32	2.17 ± 1.07	1.83 ± 0.69

Table 8: Suffix protection

e	Score	Rank
1	1.17 ± 0.37	1.08 ± 0.28
8	2.67 ± 0.75	2.42 ± 0.49
32	2.75 ± 0.72	2.50 ± 0.65

E Selective Head Visualization

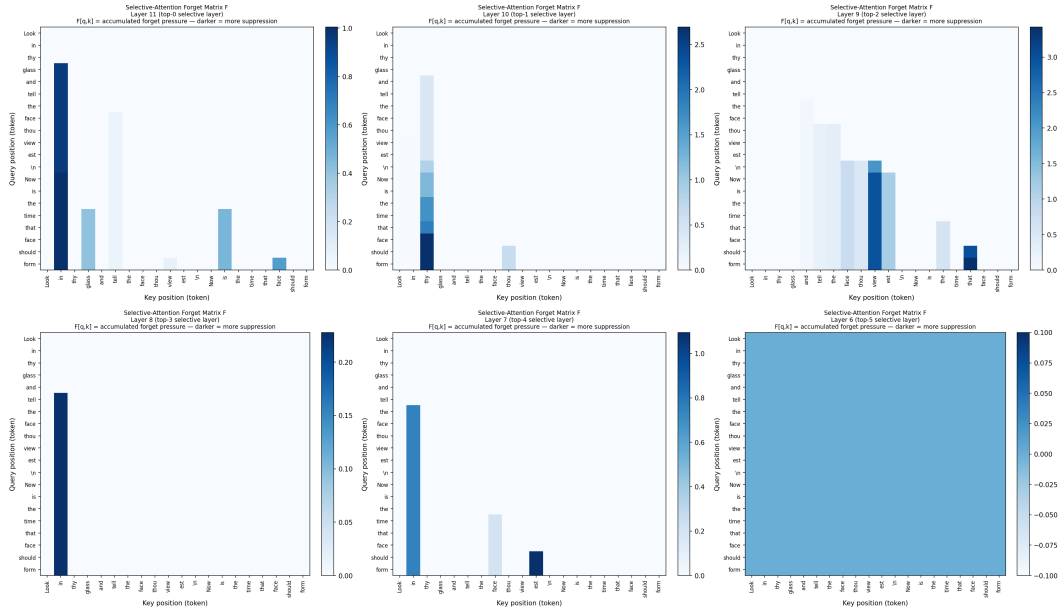


Figure 10: Selective Head across all 6 upper layers for Sonnet Generation