Optimizing Long-Context Understanding with Gated Attention Mechanisms

by Aileen Liao and Ethan Chang
6.7960 MIT Deep Learning

Motivation

Transformer-based models, such as GPT, Llama, and BERT, utilize attention mechanisms where each token in a sequence attends to every other token, generating a full attention matrix. However, as the number of tokens increases, this attention matrix grows quadratically in size. For long sequences, this becomes highly computationally intensive and memory-heavy. The challenge lies in reducing this attention cost while still preserving the model's ability to capture complex dependencies across long sequences.

Prior Work

Much of the research in transformers and attention mechanisms has focused on improving long-range dependency handling while minimizing the quadratic complexity of full self-attention (Vaswani et al., 2017). Methods like Linformer, Longformer, and Reformer tackle the problem by approximating or reducing the attention matrix size using techniques like sparse attention, low-rank approximation, and locality-sensitive hashing.

Linformer (Wang et al., 2020) addressed the quadratic complexity by introducing low-rank approximation to the attention matrix. It approximates the full attention matrix by factorizing it into a low-rank matrix, thus reducing the number of parameters and operations. This approximation assumes that the attention matrix is globally consistent and can be reduced without losing much information. However, the low-rank approximation can be ineffective when the input sequence has varying dependencies, or when capturing fine-grained attention patterns is critical. While efficient, Linformer’s approach lacks the flexibility needed to adapt to dynamic sequence characteristics, as it relies on a fixed low-rank matrix across all sequences.

Longformer (Beltagy et al., 2020) introduced a sparse attention mechanism that enables each token to attend only to a small local window of neighboring tokens, combined with a global attention mechanism to capture long-range dependencies. This reduces the complexity to $O(n \cdot w)$, where n is the sequence length and w is the window size. While Longformer is efficient for long sequences, its reliance on a fixed window size can be limiting. If a sequence contains both very local and very long-range dependencies, the fixed

window size may either miss critical local context or fail to capture distant dependencies effectively. Longformer does not allow for dynamic adaptation of the attention span, which may hinder performance in tasks requiring flexible and diverse contextual information.

Reformer (Kitaev et al., 2020) introduced locality-sensitive hashing (LSH) to approximate the attention matrix. By hashing token pairs into buckets and attending only to tokens within the same bucket, Reformer reduces memory usage and allows for processing of very long sequences. While LSH significantly reduces the space complexity, it sacrifices the precision of attention, as the hashing process introduces randomness and does not always capture the exact relationships between tokens. The loss of precision due to the approximation can impact tasks requiring precise global dependencies, especially in tasks like language modeling and machine translation where token interactions are highly nuanced.

One promising approach to address the issues was Duo Attention (Xiao et al., 2023), which introduced a hierarchical attention head mechanism. In Duo Attention, the attention mechanism is divided into two components:

- Retrieval Heads: These heads compute the full attention across all tokens, which captures long-range dependencies.
- Streaming Heads: These heads focus only on the most recent tokens, optimizing short-term computational efficiency.

A learnable hyperparameter decides the weighting between these two types of heads, allowing the model to choose between the full attention (global context) and a more local, efficient attention mechanism. This value is determeined for each KV head at a global value based on the KV hash. While this approach improves efficiency, there remains an opportunity to further optimize the attention matrix through the use of learned masks, which can simplify the full attention calculation.

Our Method

To merge the benefits of both local and global attention, we introduce the Gated Attention mechanism that extends Duo Attention. The motivation is that if provided the input token, it is possible to infer a metric based on that token at compute time that decides if the model requires long or short context to predict the next token. Here, the final attention mask is computed by interpolating between two distinct types of attention distributions for each of the token input:

- Local Attention: Focuses on a sliding window of tokens, allowing each token to attend only to a fixed number of neighboring tokens.
- Global Attention: Respects the causal constraints, ensuring that each token can attend only to itself and earlier tokens (in autoregressive models).

The mask is dynamically adjusted based on a gating signal, which determines the relative importance of local versus global attention at each step. This dynamic nature ensures that the model can efficiently handle both short-range and long-range dependencies. The gating signal is determined by the input token through an mlp from d_hidden to 1, which makes each token able to decide how far does the output need to attend to. Our method is different from DuoAttention as instead of calculating one single parameter that will determine if the current KV head will be using the full attention or the local attention for a forward path, the attentions are masked and combined locally based on each individual input token to futher scrutinize the detail in each attention masks.

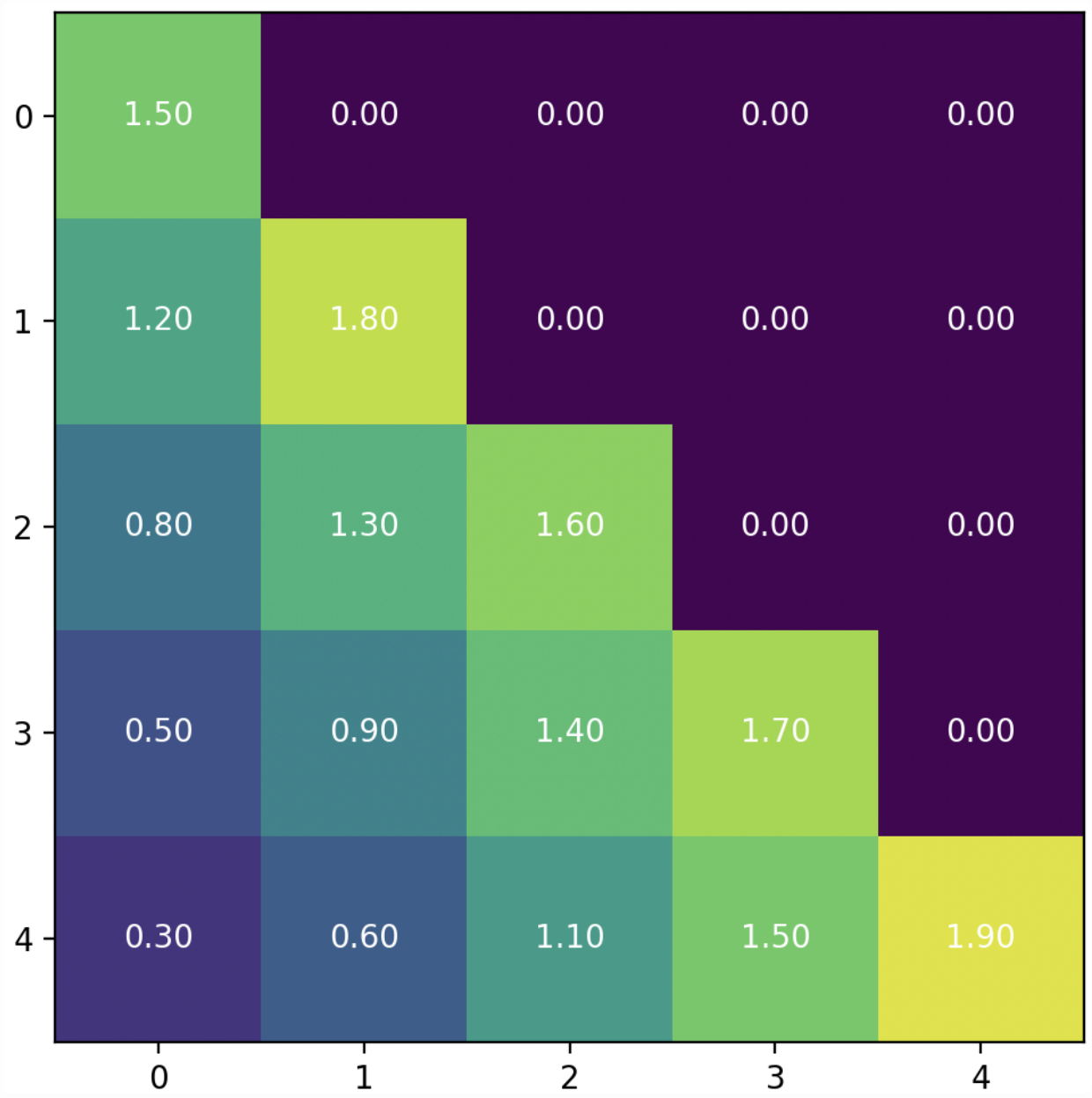
We hypothesize that this might reduce the inference time needed due to the increase sparsity in the attention matrix. It will also help decrease the loss of next token prediction to remove unesscesary attention inputs from tokens far from the current token. We think this will also work well with small models that cant afford to have multiple attention heads.

Gated Attention Implementation

Causal Attention

Designed for autoregressive models, the causal attention ensures that each token can attend only to itself and earlier tokens in the sequence, maintaining the temporal order of dependencies. This is normally done by masking out the upper right triangle to -inf. The mask is represented as a lower triangular matrix, where values above the diagonal are set to -inf, effectively preventing future token attention. This, after softmaxxing often results in a attention matrix that only has the lower triagular values:

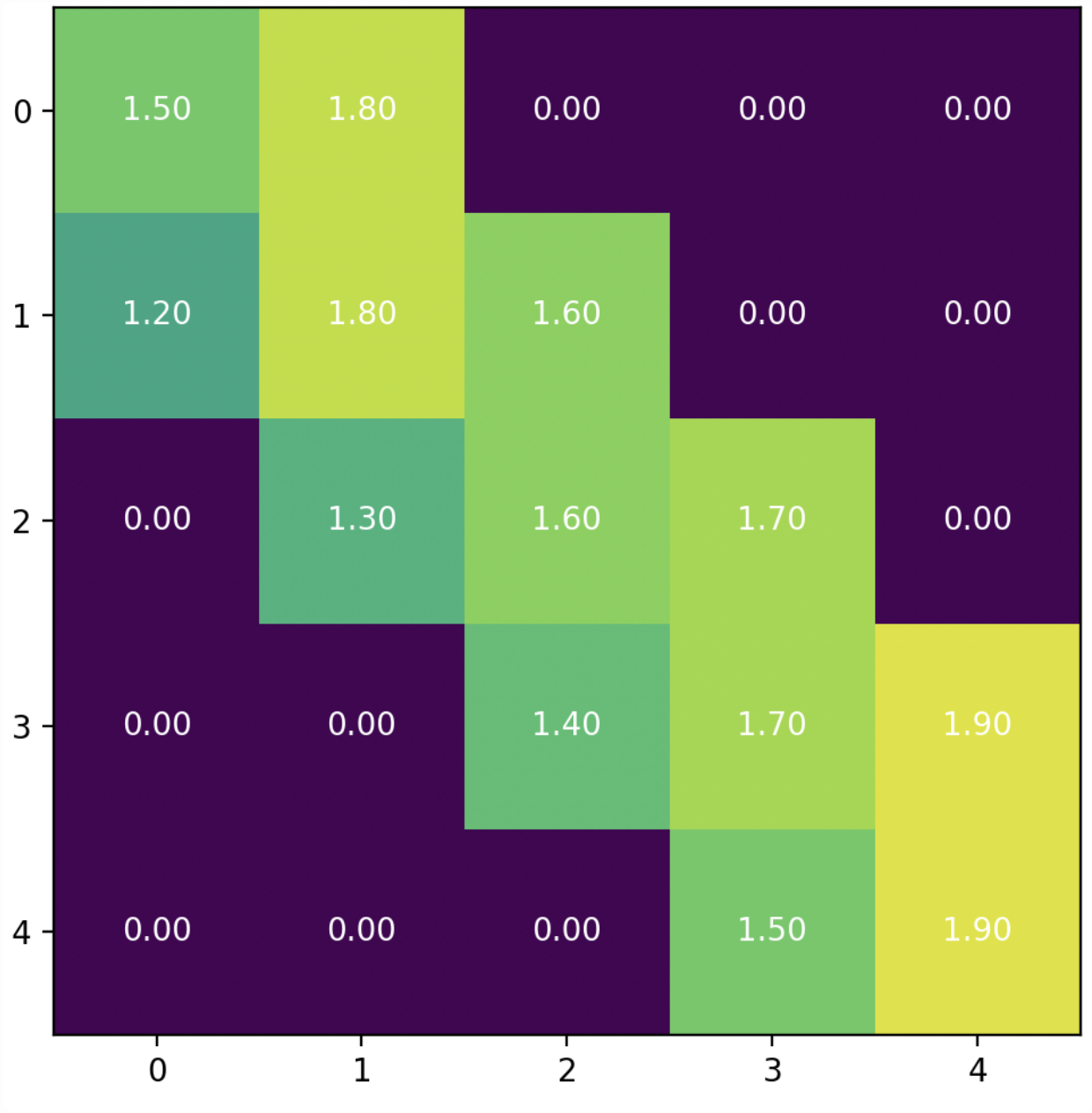
Example:



Local Window Attention

The local window mask restricts attention to a fixed window size around each token. For a window size of 3, each token attends only to itself and its immediate neighbors. This mask is efficient for capturing short-range dependencies, focusing on local context while reducing unnecessary computation. After applying this mask, the attention values normally looks like the following

Example (window size 3):



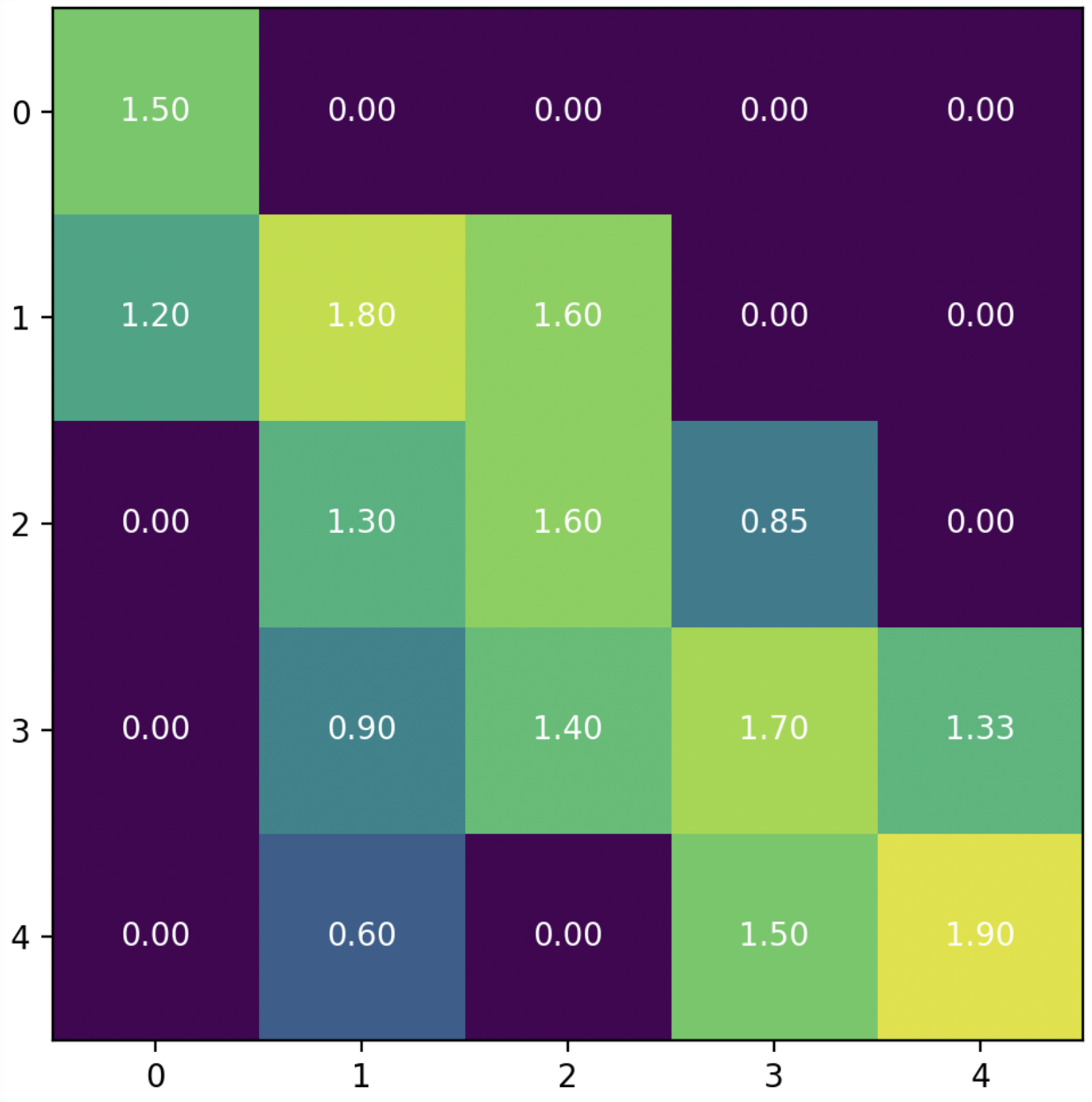
Gated Attention

The gated attention mechanism interpolates between the local and global attentions, dynamically adjusting the attention based on the gating signal based on each token. The final attention probabilities are computed as:

attn_probs=(1-gates)×local_attn_probs+gates×global_attn_probs

For a ceratin input token streams, the gate nlp applies to each of the incoming token and decide a value based on 0-1 to use a masking for that column of attention. Then local and

global attention is combined accordingly:



How the Gated Attention Calculation Works

1. For an input stream with tokens [A,B,C,D,E], as they come into an attention layer, we individually calculate a gate score using an MLP layer from n_hidden to 1 and a layer of softmax. This gives a query_length long stream of gate values, assumed to be [1, 0, 1, 0.5, 0.7] for the above figure.
2. We then individually calculate the causal attention and the local attention by using individual -inf masks and soft masking.
3. As the last step we interpolate between the two attentions with regards to the gate values for each token. In this example the first column will be the same as the causal attention, the second column will be the same as the local window attention. The rest will be something in between the two masks with regards to each gate value.

This will then be sent through the next layers.

Complexity Analysis

Causal Attention

For a sequence of length n, the causal attention is a lower triangular matrix, meaning there are approximately n(n−1)/2 valid attention weights (ignoring the diagonal).
This results in a relatively sparse attention matrix, but for long sequences, the size of the matrix grows quadratically. Each token can attend to every token before it, leading to a large number of operations.

Local Window Attention

In a local window mask, each token attends only to its neighbors within a fixed window size, w. This leads to a matrix that is sparse (only non-zero values for positions within the window around each token).
For each token in a sequence of length n, it will attend to approximately w tokens (with the assumption that the window is fixed and symmetric). This reduces the number of valid attention weights significantly when w is much smaller than n, making the computation more efficient than the causal mask for long sequences.
If we use a window size of 3, each token will attend to itself and its two neighbors, reducing the complexity compared to the causal mask.

Gated Attention

The gated attention combines local and global attention by interpolating between the two resulting attentions. The computation for each token is the weighted sum of the local and global attention probabilities.
If the gate values are close to 0 or 1 (local for short-range attention and causal for autoregressive sequence generation), the gated mask might not require more operations than computing the attention for each of the individual masks separately, especially if both local and global attention matrices are sparse. However some value in the middle will require extra averaging of the existing attentions.

Training

Setup

We compared the gated attention versus the normal attention network by using the llama3.2-1B model. In this model, there were a total of 16 attention layers. We added the aforementioned gated attention mechanism to all these 16 layers (we call this model the **gated model**), and compared it to the original model where we also swapped out the orignal attention layers and performed the simplest causal attention mask (we call this the **baseline model**). This was to prevent extra rotational encoding calculation and multi-head queries that was implemented in the llama3.2 model to effect the experiment.

The model was trained using autocast, which is a feature in PyTorch that automatically handles mixed precision training by dynamically choosing between single-precision (FP32) and half-precision (FP16) operations to optimize performance and memory usage while maintaining numerical stability.

The dataset used was the "wikitext" dataset, specifically the "wikitext-103-raw-v1" version. 1000 samples were used in the training dataset and next token prediction was performed. A linear learning rate scheduler was employed, which included a warmup phase (500 steps) and gradually decayed the learning rate over the course of training. Loss was calculated using a cross entropy loss for next token prediction. Both the gated and the baseline model were trained for 4 epochs with 1000 samples each.

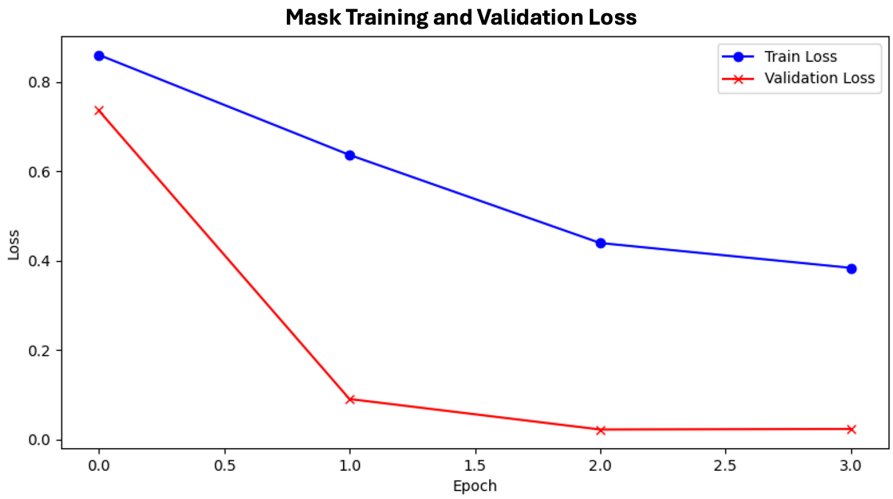


Fig 1: Training curves for gated model.

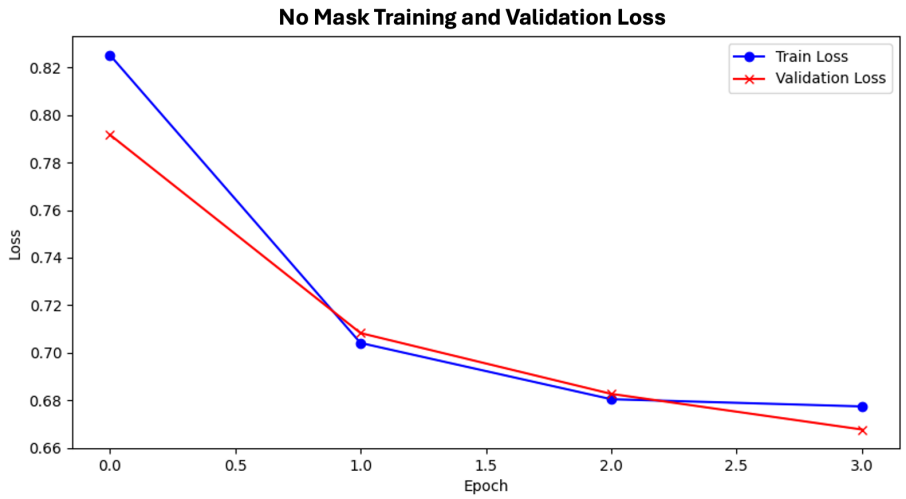


Fig 2: Training curves for baseline model. (try to generate on the same scale as pic 1)

Both the gated and baseline models were trained to provide comparison. The gated method seemed to allow for faster convergence as the validation loss dropped drasticly within the first epoch and achieved lower losses compared to the unmasked training.

On training the gated model, additional loses were added to ensure convergence of the gate_value layer. On top of the cross entropy loss that was applied to the baseline model, we added a sparsity loss, defined as \(-gate*(1-gate)\), to push the gate value towards 0 or 1. Additionally, a mean loss defined as mean(gate_values) was added to push the gate value closer to 0 and encourage the use of the local mask. Each of the 16 attention layers had their own gate MLP. For the sake of simplicity, the training and evaluation models operate identically; both mix attentions based on gate_values after the causal attention and sliding attention is calculated.

Evaluation

We used 1000 samples from the wikitext evaluation dataset and evaluated next token prediction. Specifically, we looked at the following metrics:
Perplexity: \[\exp\left(\frac{1}{N} \sum_{i=1}^N \log(p_i)\right) \]
where \(p_i\) is the probability and N is tokens it is averaged over
Loss: Cross entropy loss of the log probs and the correct next token.
Inference Time: Average time to infer the \(log(p_i)\) from a a new input token
Mask gating values: the values that decide the mask by inputing the token to a gate MLP and a softmax afterwards

These metrics were evaluated across different context lenght after both models were trained.

Results

We found that for context length of 512, the fine-tuned (masked) model had a lower perplexity and loss as seen in Figures 4, 5. However, the inference time did not decrease for the fine-tuned model compared to the baseline (512). 512 tokens, if counted around 4 characters per toker, is around 2000 characters long. This is around the size of a short length paragraph.

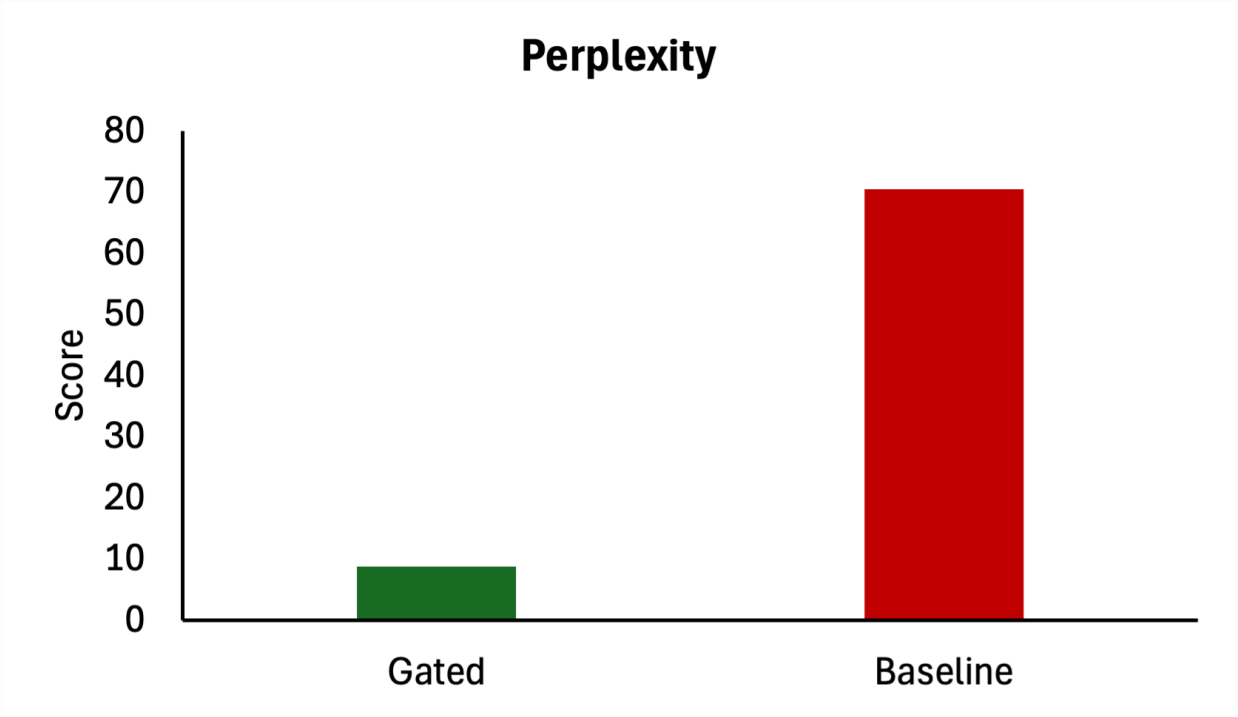


Fig 3: Perplexity results between gated and baseline models.

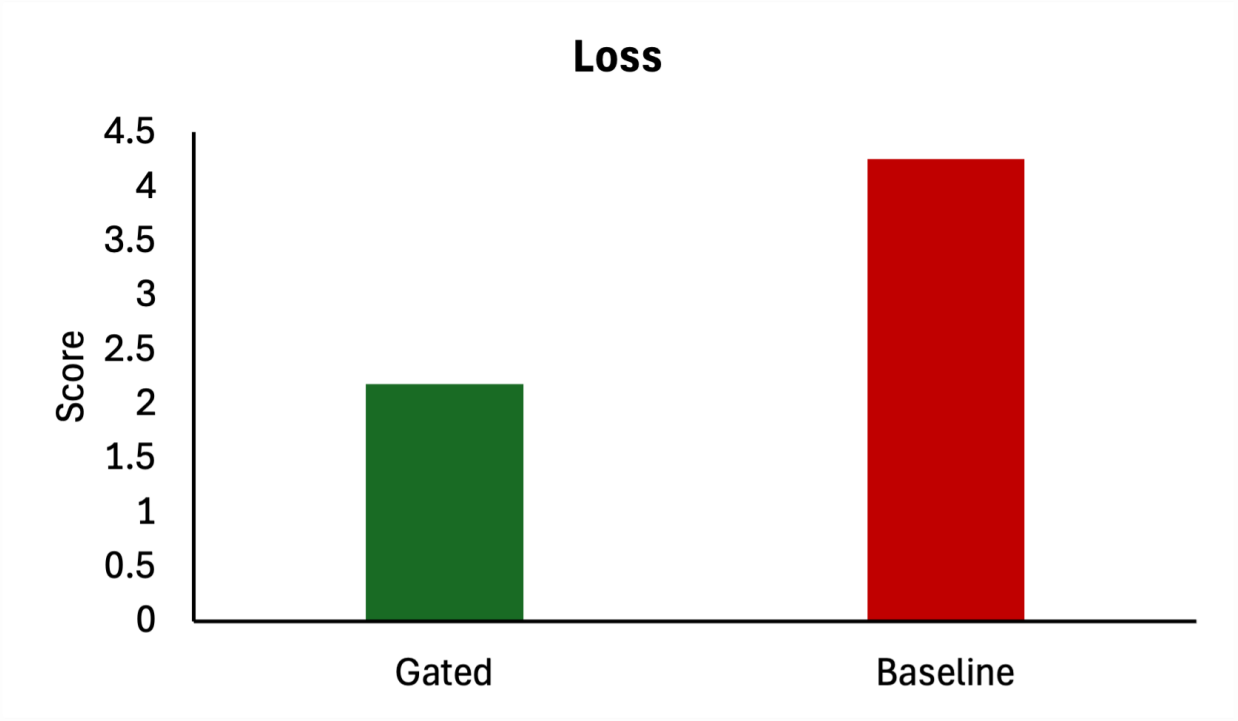


Fig 4: Loss between gated and baseline models.

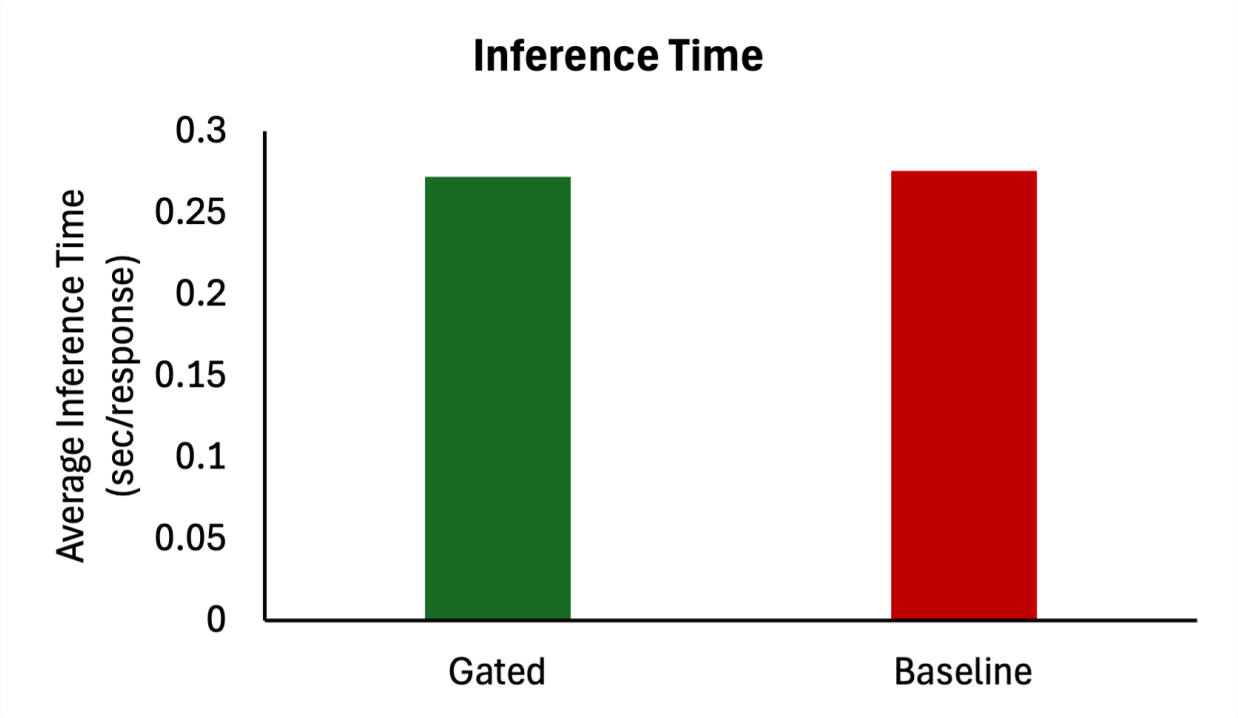


Fig 5: Inference time bewteen gated and baseline models.

A sweep of metrics vs context lengths were performed. The baseline had a consistently higher loss and perplexity than the fine-tuned model. However, the inference time was similar between the two, with the baseline having slightly faster performance.

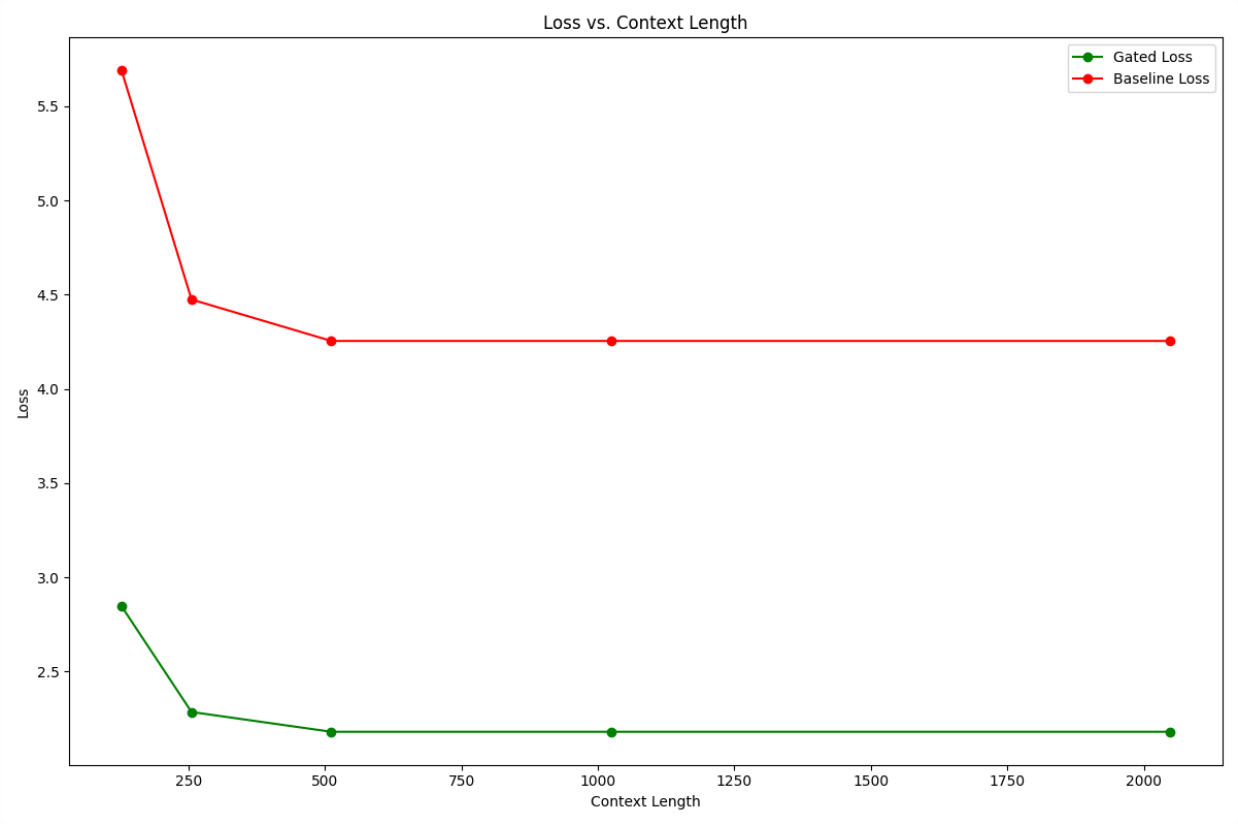


Fig 6: Loss of fine-tuned and baseline models vs context lengths.

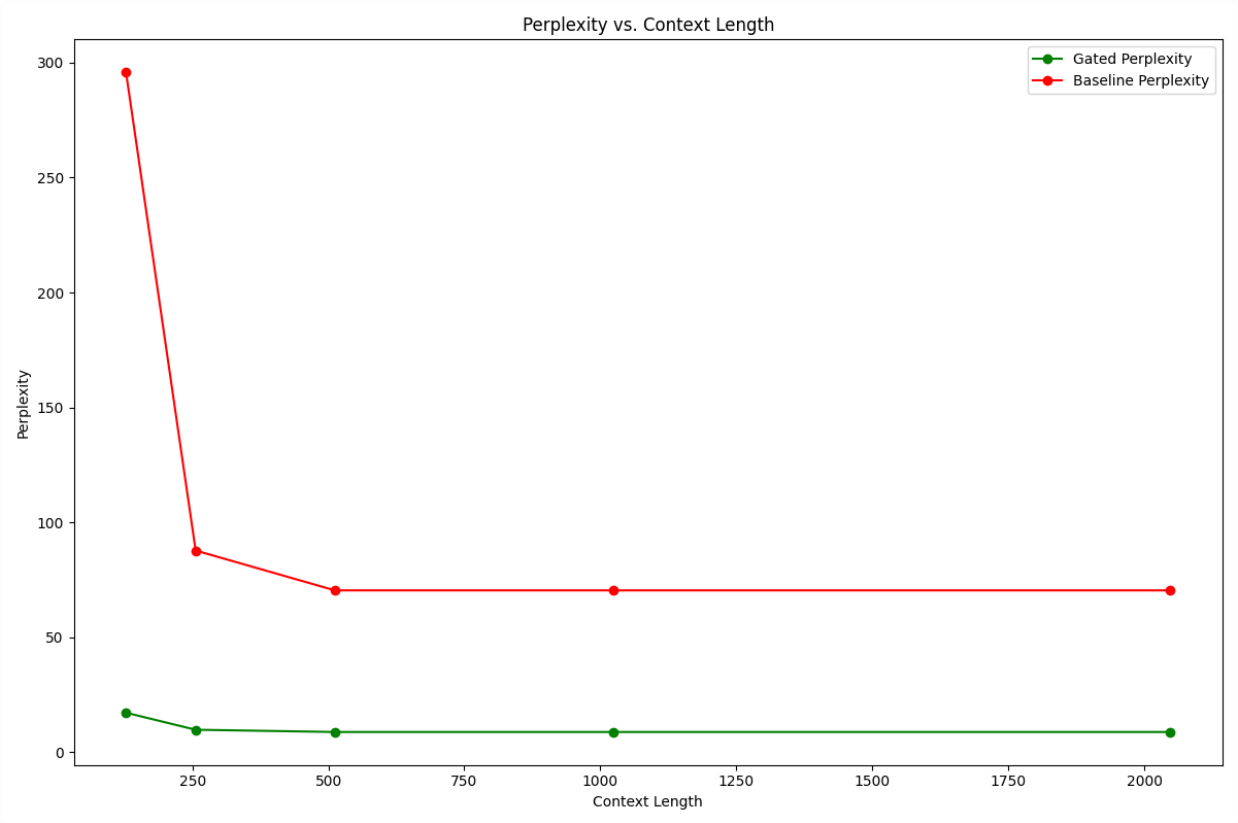


Fig 7: Fine tuned and Baseline perplexities vs context lengths.

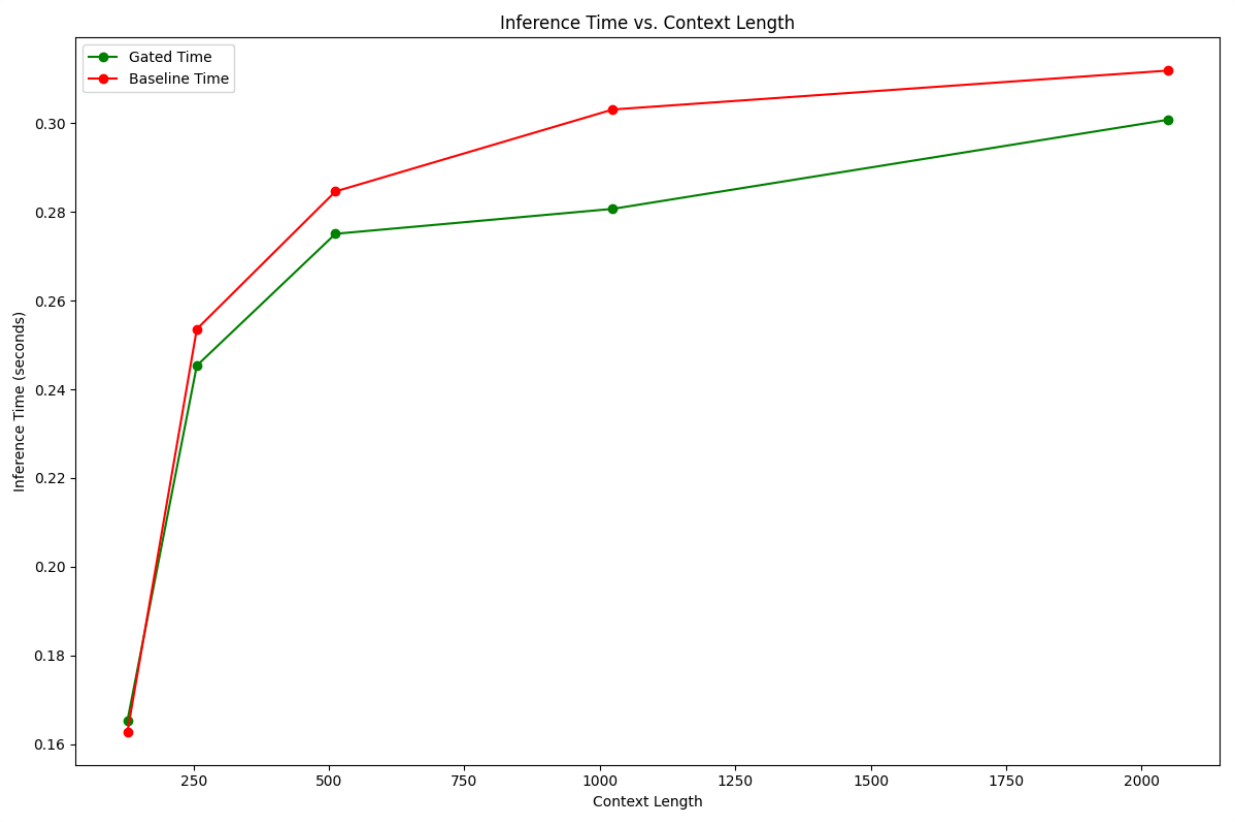


Fig 8: Inference times for fine-tuned and baseline models vs context lengths.

To analyze possible reasons why loss and perplexity were improved in the fine-tuned model but inference did not increase, the gate means, standard deviations (Fig 9), and sparsity (Fig 10) were plotted for each of the 16 layers. Various layers had different gating values, but there was no clear trend between layers and gating values or standard deviations. The fine-tuned model had a consistent sparsity due to the masking of approximately 50% for all layers. However, the baseline had varying sparsities for different layers. The averaged sparsity of all layers in baseline is similar to the fine-tuned, leading to similar inference times.

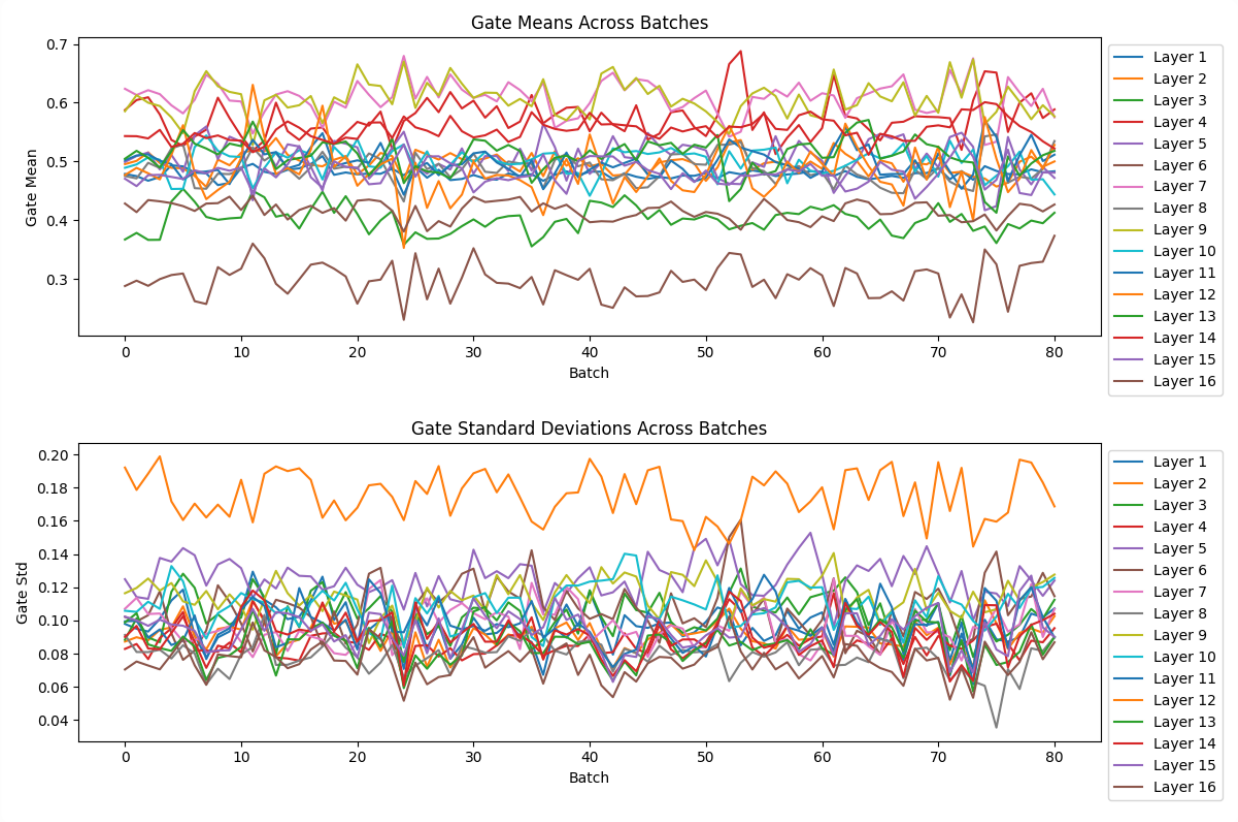


Fig 9: Gate means and standard deviations for each layer were shown for the masked finetuned model.

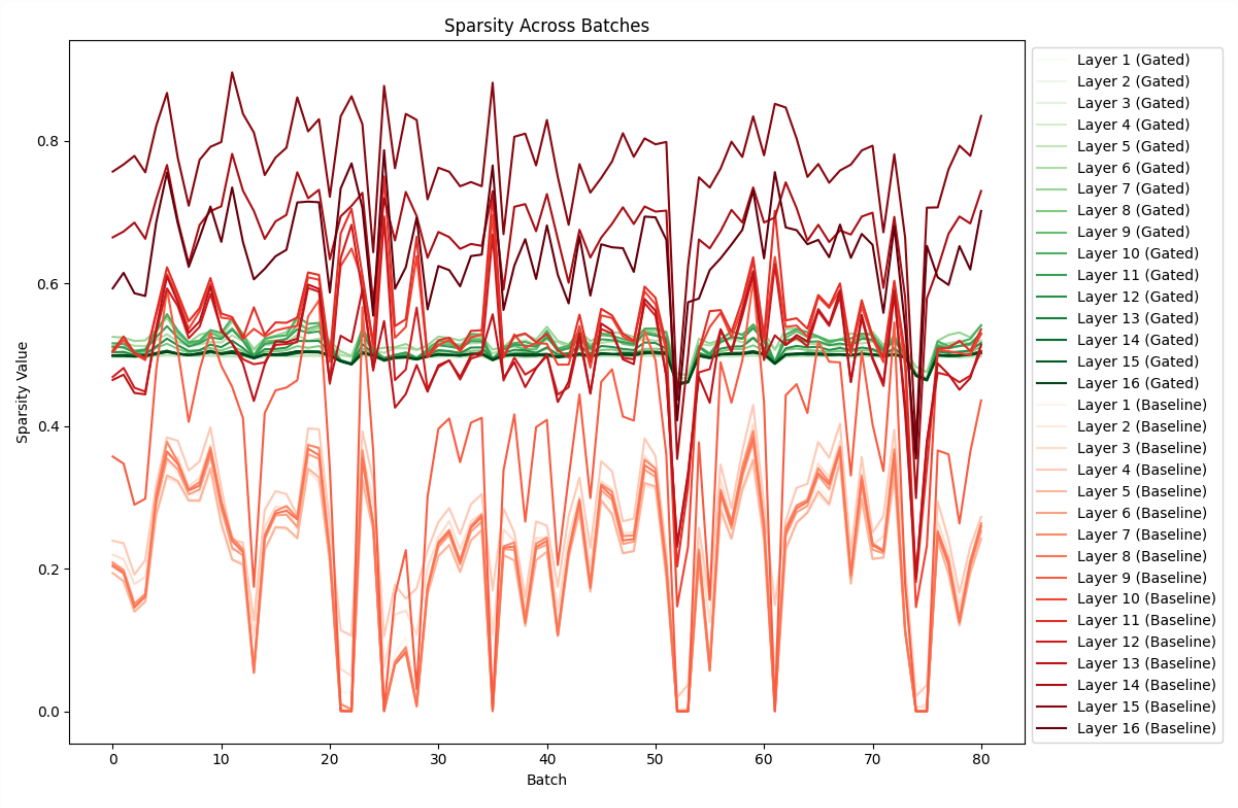


Fig 10: The ratio of sparsity of the attention matrices were plotted for each layer of the

baseline vs gated
models. Baseline models had higher variability in sparsity between layers.

Discussions and Limitations

For this small model, the gated attention mechanism decreased the prediction loss for long context generation. However, the gated mechanism did not reduce inference time. From the plots of the sparsity of different layers, the model itself learned to lower the sparsity of early on layers, and increased the sparsity of later layers that were closer to the output. We hypothesize that this may be due to the generation method for next token prediction as earlier layers capture more short term behaviors (i.e. phrases). So, in early layers the gate values could be lowered to only look at sliding window context; while later layers were responsible for global featured where long-term full attention matrix was needed.

For short context lenghts, the gated mechanism did not cause a drastic difference in inference time. As context length increased, the difference caused by calculating the attention layer twice was more apparent even with the sparsity induced by the gating mechanism. This could be possibly resolved by using certain weight dropping mechanisms in the evaluation. For example, using gumble softmax or applying a certain threshold to decide which mask to use could be implemented to bypass the muiltiple O(n) averaging operations.

The study was only done on a small 1B model. The gated mechanism may be redudant for larger models that have multiple attention heads. As the Duo-Attention Paper suggests, applying a global parameter that decides whether an attention head uses a causal mask or a sliding window mask is a great way to reduce inference time. Further investigation is needed to understand whether the mechanism is needed as models grow since the gating is already causing slight overhead. The calculculations arise from both computing attention probabilities and mixing them regarding each input token.

Additionally, there were limitations in dataset size and compute resources for training. Larger models with longer training and more comprehensive datasets could yield faster inference. Future work could be done on different masking/gating mechanisms for multi-level cascading masks. GPU optimization methods and evaluating tensor storage could also speed up inference times.

Conclusion

The Gated Attention with Hierarchical Masking mechanism improved long-context understanding in LLMs. By combining local window attention, global causal attention, and a dynamic gating mechanism, the model achieved better performance in terms of perplexity and loss for longer contexts.

References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. A., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Proceedings of NeurIPS 2017, 30, 5998-6008.

Wang, S., Liu, J., & Yu, M. (2020). Linformer: Self-attention with linear complexity. In Proceedings of ICLR 2020.

Beltagy, I., Lo, K., & Cohan, A. (2020). Longformer: The long-document transformer. In Proceedings of arXiv preprint arXiv:2004.05150.

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. In Proceedings of ICLR 2020.

Xiao, G., Tang, J., Zuo, J., Guo, J., Yang, S., Tang, H., Fu, Y., & Han, S. (2023). DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. arXiv preprint arXiv:2303.06153.