
Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free

Zihan Qiu^{*1}, Zekun Wang^{*1}, Bo Zheng^{*1}, Zeyu Huang^{*2},
Kaiyue Wen³, Songlin Yang⁴, Rui Men¹, Le Yu¹, Fei Huang¹, Suozhi Huang⁵,
Dayiheng Liu^{∞1}, Jingren Zhou¹, Junyang Lin^{∞1}

¹Qwen Team, Alibaba Group ²University of Edinburgh ³Stanford University

⁴MIT ⁵Tsinghua University

Abstract

Gating mechanisms have been widely utilized, from early models like LSTMs [1] and Highway Networks [2] to recent state space models [3], linear attention [4], and also softmax attention [5, 6]. Yet, existing literature rarely examines the specific effects of gating. In this work, we conduct comprehensive experiments to systematically investigate gating-augmented softmax attention variants. Specifically, we perform a comprehensive comparison over 30 variants of 15B Mixture-of-Experts (MoE) models and 1.7B dense models trained on a 3.5 trillion token dataset. Our central finding is that a simple modification—*applying an head-specific sigmoid gate after the Scaled Dot-Product Attention (SDPA)*—*consistently improves performance*. This modification also enhances training stability, tolerates larger learning rates, and improves scaling properties. By comparing various gating positions and computational variants, we attribute this effectiveness to two key factors: (1) introducing non-linearity upon the low-rank mapping in the softmax attention, and (2) applying query-dependent sparse gating scores to modulate the SDPA output. Notably, we find this sparse gating mechanism mitigates ‘massive activation’ [7], ‘attention sink’ [8], and enhances long-context extrapolation performance, and we also release related codes and models to facilitate future research. Furthermore, the most effective SDPA output gating is used in the Qwen3-Next models.

1 Introduction

Gating mechanism is well-established in neural networks. Early architectures, such as LSTMs [1], Highway Networks [2] and GRUs [9], pioneer the use of gating to control information flow across time steps or layers and improve gradient propagation. This principle persists in modern architectures. Recent sequence modeling works, including state-space models [3, 10] and attention mechanisms [11, 12, 4, 13, 14, 15, 16, 17, 5, 6] commonly apply gating, often to modulate the outputs of token-mixer components. Despite its widespread adoption and empirical success, most recent works do not look into the gating mechanisms like the gating scores and their effect on the model’s hidden states.

Insufficient understanding hinders assessing gating’s true contribution, especially when confounded with other architectural factors. For instance, while Switch Heads [18, 19] introduces a sigmoid gating to select top-K attention head experts, our experiments reveal an interesting finding (Appendix A.1): substantial performance gains persist even when reduced to a single expert, where the gate simply modulates the value output. This strongly suggests the gating itself provides significant intrinsic value, separate from the routing mechanism. Similarly, in Native Sparse Attention (NSA) [20], while overall performance improvements are demonstrated, they do not disentangle the contributions of its gating mechanism from the effects of the sparse attention design itself. These considerations underscore the need to rigorously disentangle the effects of gating from other architectural components.

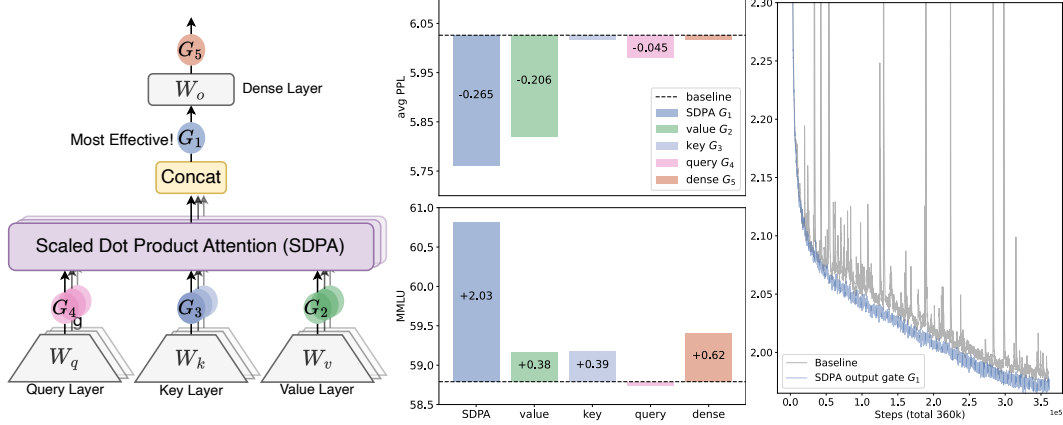


Figure 1: **Left:** Investigated positions for applying gating operations within the self-attention layer.; **Middle:** Performance comparison (Test PPL and MMLU) of 15B MoE models with gating applied at various positions. Gating after SDPA (G_1) yields the best overall results. Gating after the Value layer (G_2) also demonstrates notable improvements, particularly in PPL. **Right:** Training loss comparison (smoothed, 0.9 coeff.) over 3T tokens between baseline and SDPA-gated 1.7B dense models under identical hyperparameters. Gating results in lower final loss and substantially enhanced training stability, mitigating loss spikes. This stability allows for potentially higher learning rates and facilitates better scaling.

In this work, we investigate gating mechanisms in the standard softmax attention [21] (Sec.2.2). Specifically, we introduce gating at distinct positions (Fig. 1): after the query (G_4), key (G_3), and value projections (G_2); following the Scaled Dot Product Attention (SDPA) outputs (G_1); and after the final dense output layer (G_5). Our exploration covers gating variants including elementwise and headwise, head-specific and head-shared, as well as additive and multiplicative forms. We find that: **(i)** applying SDPA output head-specific gating (G_1) yields the most significant performance improvements (e.g., up to 0.2 PPL reduction and 2 points on MMLU); **(ii)** the SDPA output gating also improves training stability, nearly eliminating loss spikes, enabling larger learning rates and enhancing model scalability.

We identify two factors contributing to the efficacy of gating: **(i) Non-Linearity.** The two consecutive linear layers - the value (W_v) and dense (W_o) projections - can be rewritten into one low-rank linear projection. Therefore, introducing non-linearity through gating at positions G_1 or G_2 can increase the expressiveness of this low-rank linear transformation (Sec. 4.1). **(ii) Sparsity.** Although non-linear gating variants consistently enhance performance, we observe that their gains vary. Our analysis further reveals that the pronounced sparsity of the gating scores is another crucial factor, introducing input-dependent sparsity to SDPA outputs (Sec. 4.2). Sparse gating eliminates the *massive activation* [7] and *attention sink* [8]: the initial tokens have large activation values in the corresponding hidden states (Tab. 4) and disproportionately dominate attention scores (Fig. 2, Sec. 4.3). Previous work [8, 7, 22] explains attention sinks as an accumulation of redundant attention due to non-negative softmax normalization. Empirically, we verify that when *query-dependent sparse gating is applied at the SDPA output*, both our dense and MoE models (trained on 3.5T tokens) exhibit no attention sink. Furthermore, these models demonstrate superior performance in length generalization, achieving a gain of over 10 points on RULER [23](Sec.4.4).

Practical Recommendation. For best results, apply *elementwise SDPA gating G_1* (i.e., gating after the attention-weighted value projection) and train with a *moderately increased learning rate*.

2 Gated-Attention Layer

2.1 Preliminary: Multi-Head Softmax Attention

Given an input $X \in \mathbb{R}^{n \times d_{\text{model}}}$, where n is the sequence length and d_{model} is the model dimension, the computation of transformer’s attention layer [21] could be divided into four stages.

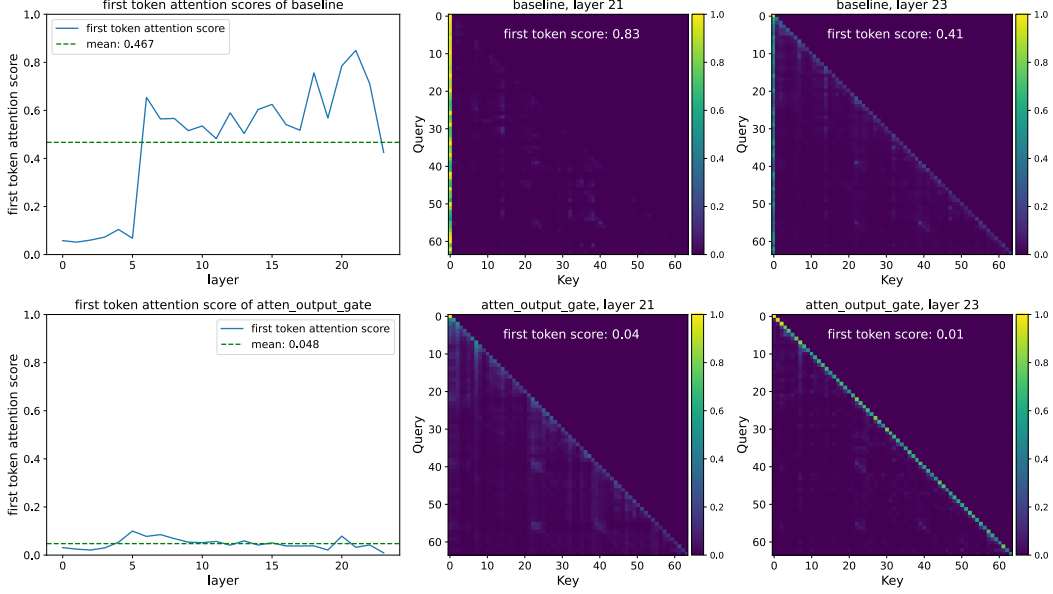


Figure 2: **Left:** Proportion of attention allocated to the initial token per layer (test perplexity dataset). The baseline model suffers from a significant attention sink, with an average of 46.7% of attention scores across layers directed towards the first token. Introducing a gate effectively alleviates this, reducing the proportion to 4.8%. **Right:** Average attention map weights for each head. Layer 21 in the baseline model demonstrates a strong attention sink (83% on the first token), which is substantially reduced by the gate (4%). In the final output layer, the gate amplifies the existing tendency for the model to attend to individual tokens within the sequence.

QKV Linear Projections: The input X is linearly transformed into queries Q , keys K , and values V using learned weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $Q, K, V \in \mathbb{R}^{n \times d_k}$:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \quad (1)$$

Scaled Product Dot-Product Attention (SDPA): computes attention scores between queries and keys, followed by a softmax normalization. The output is a weighted sum of the values:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V, \quad (2)$$

where $\frac{QK^T}{\sqrt{d_k}} \in \mathbb{R}^{n \times n}$ represents the scaled dot-product similarity matrix, and $\text{softmax}(\cdot)$ ensures the attention weights are non-negative and sum to 1 across each row.

Multi-Head Concatenation: In multi-head attention, the above process is repeated for h heads, with each head having its projection matrices W_q^i, W_k^i, W_v^i . All heads' outputs are concatenated:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h), \quad (3)$$

where $\text{head}_i = \text{Attention}(QW_q^i, KW_k^i, VW_v^i)$.

Final Output Layer: The concatenated SDPA output is passed to the output layer $W_o \in \mathbb{R}^{hd_k \times d_{\text{model}}}$:

$$O = \text{MultiHead}(Q, K, V)W_o. \quad (4)$$

2.2 Augmenting Attention Layer with Gating Mechanisms

The gating mechanism is formalized as:

$$Y' = g(Y, X, W_\theta, \sigma) = Y \odot \sigma(XW_\theta), \quad (5)$$

where Y is the input to be modulated, X is another input used to compute the gating scores¹, W_θ refers to the learnable parameters of gate, σ is an activation function (e.g., sigmoid), and Y' is

¹We adopt the hidden states after pre-normalization as X .

the gated output. The gating score, $\sigma(XW_\theta)$, effectively acts as a dynamic filter, controlling the information flow from Y by selectively preserving or erasing its features.

This work comprehensively investigates variants of gating mechanisms within the attention layer. Our exploration focuses on five key aspects: **(1) Positions.** We study the effect of applying gating at different positions, as illustrated in Fig. 1(left): (a) after the Q, K, V projections (Equ. 1), corresponding to positions G_2, G_3, G_4 in Fig. 1(left); (b) following the SDPA (Equ. 3) outputs (G_1). (c) after the final concatenated multi-head attention outputs (Equ. 4, G_5). **(2) Granularity.** We consider two levels of granularity for the gating score: (a) Headwise: A single scalar gating score modulates the entire output of an attention head. (b) Elementwise: Gating scores are vectors with the same dimensionality as Y , enabling fine-grained, per-dimension modulation. **(3) Head Specific or Shared.** Given the multi-head nature of attention, we further consider: (a) Head-Specific: each attention head has its specific gating scores, enabling independent modulation for each head. (b) Head-Shared: W_θ and gating scores are shared across heads. **(4) Multiplicative or additive.** For applying gating score to Y , we consider (a) Multiplicative Gating: The gated output Y' is computed as: $Y' = Y \cdot \sigma(X\theta)$. (b) Additive Gating: $Y' = Y + \sigma(X\theta)$. **(5) Activation Function.** We mainly consider two common activation functions: SiLU [24] and sigmoid. We only use SiLU for additive gating due to its unbounded output range, and sigmoid only gives scores in $[0, 1]$. Additionally, to further dissect the mechanisms underlying gating’s effectiveness, we also consider Identity Mapping or RMSNorm [25] (detailed in Sec 4.1). Unless otherwise specified, we employ head-specific, multiplicative gating utilizing the sigmoid activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$).

3 Experiments

3.1 Experimental Setups

Model Architecture and Training Settings We conduct experiments on both MoE models (15B total parameters with 2.54B activated, 15A2B) and dense models (1.7B total parameters). The 15A2B MoE models utilize 128 total experts with top-8 softmax gating, fine-grained experts [26], global-batch LBL [27], and z-loss [28]. We adopt group query attention (GQA) [29] for the attention part. More detailed model architecture configurations are discussed in Appendix A.2. We train the models on subsets of a 4T high-quality tokens, encompassing multilingual, math, and general knowledge content. A sequence length of 4096 is used. More detailed configurations, such as learning rate and batch size (bsz), will be introduced in each part. Other hyperparameters follow the default values of the AdamW optimizer. Since the parameters and flops introduced by the gating are relatively small, *the wall-time latency introduced by gating is less than 2%*. **Evaluation** We test the few-shots results on popular benchmarks, including, Hellaswag [30] for English, MMLU [31] for general knowledge, GSM8k [32] for math reasoning, HumanEval [33] for coding, C-eval [34] and CMMLU [35] for Chinese proficiency. We also test the perplexity (PPL) on diverse held-out test sets, including domains like English, Chinese, Code, Math, Law and Literature.

3.2 Main Results

3.2.1 Gated Attention for MoE models

We first compare different gatings on the training-efficient MoE-15A2B models. All models use a scheduler that warms up to a maximum LR of $2e-3$ in 1k steps and decays using cosine to $3e-5$. We use a global bsz of 1024, comprising 100k optimization steps. The results are summarized in Tab. 1. To provide a fair comparison, we supplement the vanilla MoE baseline (row 1) with parameter expansion methods, including increasing the number of key-value heads (row 2), increasing the number of query heads (row 3), and increasing both the total and activated number of experts (row 4). These methods introduce a comparable or greater number of parameters than the gating mechanisms.

From Tab. 1, we observe: **(i) SDPA and value output gating are effective.** Inserting gates at the output of SDPA (G_1) or the value map (G_2) is the most effective, achieving lower PPL and better overall benchmark performance than other variants. We will further investigate why gating at these two positions is effective in Sec 4.2. **(ii) Head-Specific Gating Matters.** Applying headwise gating at G_1 and G_2 introduces very few additional parameters (less than 2M for the MoE-15A2B model) but still delivers substantial improvements (rows 10 and 11). When sharing gating scores across different attention heads (we average over the query head dimension q to obtain an $n \times d_k$ score from

Table 1: Gating variant performance and results. We train the **15A2B** MoE models on **400B** tokens. d_k is the head dim, d_{model} is the model’s hidden dim, and n is the number of tokens. q refers to the number of query heads, k refers to the number of key-value heads. ‘Act Func’ is the activation function in Eq 5. ‘Score Shape’ is the gating score shape for an input $X \in \mathbb{R}^{n, d_{\text{model}}}$. ‘added param’ indicates added parameters.

Method	Act Func	Score Shape	Added Param	Avg PPL	Hellaswag	MMLU	GSM8k	C-eval
Reference Baselines (Baseline uses $q = 32, k = 4$. All methods use $d_k = 128$.)								
(1) Baseline	-	-	0	6.026	73.07	58.79	52.92	60.26
(2) $k = 8$	-	-	50	5.979	73.51	59.78	52.16	62.26
(3) $q = 48$	-	-	201	5.953	73.59	58.45	53.30	59.67
(4) Add 4 Experts	-	-	400	5.964	73.19	58.84	52.54	63.19
Gating Position Variants								
(5) SDPA Elementwise G_1	sigmoid	$n \times q \times d_k$	201	5.761	74.64	60.82	55.27	62.20
(6) v Elementwise G_2	sigmoid	$n \times k \times d_k$	25	5.820	74.38	59.17	53.97	61.00
(7) k Elementwise G_3	sigmoid	$n \times k \times d_k$	25	6.016	72.88	59.18	50.49	61.74
(8) q Elementwise G_4	sigmoid	$n \times q \times d_k$	201	5.981	73.01	58.74	53.97	62.14
(9) Dense Output G_5	sigmoid	$n \times d_{\text{model}}$	100	6.017	73.32	59.41	50.87	59.43
Gating Granularity Variants								
(10) SDPA Headwise G_1	sigmoid	$n \times q$	1.6	5.792	74.50	60.05	54.44	62.61
(11) v Headwise G_2	sigmoid	$n \times q$	0.2	5.808	74.38	59.32	53.53	62.61
Head-Specific v.s. Head-Shared Gating								
(12) SDPA Head-Shared G_1	sigmoid	$n \times d_k$	201	5.801	74.34	60.06	53.15	61.01
(13) v Head-Shared G_2	sigmoid	$n \times d_k$	25	5.867	74.10	59.02	53.03	60.61
Multiplicative v.s. Additive								
(14) SDPA Additive G_1	SiLU	$n \times q \times d_k$	201	5.821	74.81	60.06	53.30	60.98
Activation Variants								
(15) SDPA Elementwise G_1	SiLU	$n \times q \times d_k$	201	5.822	74.22	60.49	54.59	62.34

the original $n \times q \times d_k$), the benchmark improvements are smaller than those achieved by headwise gating (row 12 v.s. 10, 13 v.s. 11). This underscores the importance of applying distinct gating scores for different attention heads. **(iii) Multiplicative Gating is Preferred.** Additive SDPA output gating underperforms the multiplicative one, although it shows improvements over the baselines. **(iv) Sigmoid Activation is Better.** Replacing the activation function in the most effective gating configuration (row 5) with SiLU (row 15) leads to less improvement.

Overall, adding gating at the value layer (G_2) and SDPA output (G_1) reduces PPL by more than 0.2, outperforming various parameter-expanding baselines. However, gating at G_1 achieves better PPL and benchmark results. As long as different heads receive distinct gating scores, the granularity of gating and the choice of activation function have relatively minor impacts. We will further analyze the reasons behind these observations in Analysis (Sec 4.2).

3.2.2 Gated Attention for Dense Models.

We also conduct experiments on dense models following [36] to validate SDPA output sigmoid gating. When using gating, we reduce the width of FFN to maintain the parameter size. Most experiments use optimized hyperparameters for the baseline. For instance, for the 1.7B model trained on 400B tokens, we use a maximum LR of 4e-3 and a bsz of 1024. For training on 3.5T tokens, we increase the maximum LR to 4.5e-3 and the bsz to 2048. Prior work has established that while increased network depth, large learning rates, and large batch sizes can significantly improve model performance [37, 38, 39] and distributed training efficiency, they often introduce training instabilities [38, 40, 41]. We observe that applying gating largely reduces the loss spikes [42, 41] during training (Fig. 2 right), suggesting a promising role for gating in enhancing training stability. Therefore, we introduce another setting characterized by an increased number of layers, a higher maximum learning rate, and a larger batch size to further probe gating’s stabilizing effects.

Tab. 2 reveals that: **(i) Gating is effective across various settings** Across various model configurations (row 1 v.s. 2, 5 v.s. 8), training data (row 3 v.s. 4), and hyperparameters (row 11 v.s. 13), SDPA output gating consistently yields benefits. **(ii) Gating improves stability and facilitates scaling.** Under the 3.5T token setting, gating improves training stability, largely reducing the loss spike (Fig. 1, right). When increasing the maximum LR, baselines encounter convergence issues (row 6, 12). While adding sandwich norm [43] restores convergence, the improvement is negligible. In contrast, increasing the maximum LR in models with gating results in a noticeable improvement.

Table 2: Performance of different methods with varying learning rates, batch sizes and model configurations. ‘SDPA’ refers to the sigmoid gating after SDPA in Eq 3, and ‘sandwich norm’ [43] indicates normalizing attention/ffn outputs before adding them to the residual. When using gating, we reduce FFN’s width so that all methods have the same number of parameters. ‘-’ means the model diverges during training.

Method	Max LR	Avg PPL	HumanEval	MMLU	GSM8k	Hellaswag	C-eval	CMMLU
28 Layer, 1.7B Parameters, 400B Tokens , Batch Size=1024								
(1) Baseline	4.0×10^{-3}	7.499	28.66	50.21	27.82	64.94	49.15	49.52
(2) SDPA Elementwise	4.0×10^{-3}	7.404	29.27	51.15	28.28	65.48	50.72	50.72
28 Layer, 1.7B Parameters, 3.5T Tokens , Batch Size=2048								
(3) Baseline	4.5×10^{-3}	6.180	34.15	59.10	69.07	68.02	68.19	64.95
(4) SDPA Elementwise	4.5×10^{-3}	6.130	37.80	59.61	70.20	68.84	68.52	65.76
48 Layer, 1.7B Parameters, 400B Tokens , Batch Size=1024								
(5) Baseline	4.0×10^{-3}	7.421	28.05	52.04	32.98	65.96	51.11	51.86
(6) Baseline	8.0×10^{-3}	9.195	21.34	44.28	15.24	57.00	43.11	42.63
(7) Baseline+Sandwich Norm	8.0×10^{-3}	7.407	30.49	52.07	32.90	66.00	52.04	51.72
(8) SDPA Elementwise	4.0×10^{-3}	7.288	31.71	52.44	32.37	66.28	52.06	52.29
(9) SDPA Headwise	4.0×10^{-3}	7.370	31.10	53.83	34.12	65.59	55.07	52.38
(10) SDPA Elementwise	8.0×10^{-3}	7.325	31.10	54.47	36.62	66.40	53.91	53.80
48 Layer, 1.7B Parameters, 1T Tokens , Batch Size=4096								
(11) Baseline	5.3×10^{-3}	7.363	29.88	54.44	32.22	65.43	53.72	53.37
(12) Baseline	8.0×10^{-3}	-	-	-	-	-	-	-
(13) SDPA Elementwise	5.3×10^{-3}	7.101	34.15	55.70	36.69	67.17	54.51	54.68
(14) SDPA Elementwise	8.0×10^{-3}	7.078	31.71	56.47	39.73	67.38	55.52	55.77

In summary, we identify SDPA element-wise gating as the most effective method to augment the attention mechanism. Incorporating the SDPA output gate enables stable training under larger learning rates and batch sizes—regimes where the baseline often becomes unstable. This suggests that the optimal hyperparameter configuration shifts when using gating. *In practice, one effective way to leverage the gate is to start from the baseline’s optimal batch size and moderately increase the learning rate.* Further jointly tuning batch size and learning rate may yield additional gains.

4 Analysis: Non-Linearity, Sparsity, and Attention-Sink-Free

In this section, we conduct a series of experiments to explore why such a simple gating mechanism can yield significant improvements in performance and training stability. Here are the takeaways according to our analysis: (1) Gatings enhancing non-linearity consistently lead to performance gains (Sec 4.1); (2) The most effective SDPA elementwise gate introduces strong input-dependent sparsity (Sec 4.2), which then helps to eliminate the ‘massive activation’ and ‘attention sink’ phenomenon.

4.1 Non-linearity Improves the Expressiveness of Low-Rank Mapping in Attention

Inspired by prior works that utilize group norm for the SDPA output [13, 44], with the same setting in Sec. 3.2.1, we apply RMSNorm [25] independently to the

Table 3: Performance of different (non)-linearity augmentations.

Method	Activation Function	Avg PPL	Hellaswag	MMLU	GSM8k	C-eval
(1) Baseline	-	6.026	73.07	58.79	52.92	60.26
(2) SDPA Elementwise Gate	Sigmoid	5.761	74.64	60.82	55.27	62.20
(3) v Elementwise Gate	Sigmoid	5.820	74.38	59.17	53.97	61.00
(4) SDPA Additive Gate	SiLU	5.821	74.81	60.06	53.30	60.98
(5) SDPA GroupNorm	RMSNorm	5.847	74.10	60.15	53.75	61.14
(6) SDPA SiLU	SiLU	5.975	73.34	59.55	53.19	60.90
(7) SDPA Additive Gate	Identity	5.882	74.17	59.20	52.77	59.86

output of each attention head before concatenation. As shown in Tab. 3 row 5, applying RMSNorm, which introduces almost no additional parameters, also leads to a significant reduction in PPL.

In multi-head attention, the output of the i -th token, corresponding to the k -th head, can be expressed:

$$o_i^k = (\sum_{j=0}^i S_{ij}^k \cdot X_j W_V^k) W_O^k = \sum_{j=0}^i S_{ij}^k \cdot X_j (W_V^k W_O^k), \quad (6)$$

where W_O^k is the parameters of the output layer W_O corresponding to the k -th head². Here, S_{ij}^k is the attention score of the i -th token attending to the j -th token in the k -th head, X_j is the input to the attention for token j , and $X_j W_V^k$ is the value output of token j in the k -th head. From Equ. 6, we can merge $W_V^k W_O^k$ into one low-rank linear mapping applied over all X_j as $d_k < d_{model}$. With GQA, W_V is shared among heads within the same group, further diminishing the expressiveness.

Given that adding non-linearity between two linear mappings can improve their expressiveness [45], we have two modifications to mitigate the low-rank problem:

$$o_i^k = \left(\sum_{j=0}^i S_{ij}^k \cdot \text{Non-Linearity-Map}(X_j W_V^k) \right) W_O^k, \quad (7)$$

$$o_i^k = \text{Non-Linearity-Map} \left(\sum_{j=0}^i S_{ij}^k \cdot X_j W_V^k \right) W_O^k. \quad (8)$$

Notably, adding gating at the G_2 (Tab. 3 row 3) position corresponds to the first modification (Equ. 7), while adding gating (row 4) or group normalization (row 5) at the G_1 position corresponds to the second (Equ. 8). This also explains why adding gating or normalization at the G_5 position after W_O has no effect (Tab. 1 row 9) — it does not address the lack of non-linearity between W_V and W_O .

For additive gating at G_1 , the output of gating passes through SiLU (Tab. 3 row 4), also introducing some non-linearity, which explains the observed performance gains, albeit smaller than those achieved by multiplicative gating. Based on these insights, we conduct two additional experiments: (i) Adding SiLU only at the G_1 position without introducing additional parameters (Tab. 3 row 6). Notice this simple modification also leads to a modest reduction in PPL, but most benchmark scores remain unchanged. (ii) Removing SiLU from additive gating, such that the output of X_j after gating is directly added at the G_1 position (Tab. 3 row 7). This further diminishes the gains of additive gating.

In summary, the enhanced performance associated with effective gating variants is likely attributable to the introduction of non-linearity between W_V and W_O . Although applying gating at positions G_1 and G_2 can both introduce this non-linearity, these applications yield differing performance gains. This observed difference motivates us to further analyze the impacts of gating at these two positions.

4.2 Gating Introduces Input-Dependent Sparsity

We analyze the gating scores (Tab. 1, ‘Gate Score’ column) of models with gating applied at the value (G_2) and SDPA output (G_1) positions, evaluated on the test language modeling data. The mean gating scores for all layers are presented in Table 4, with the score distributions visualized in Fig. 3 (layer-wise scores in Appendix A.3). Key observations include:

(i) **Effective Gating Scores are Sparse.** SDPA output gatings (Elementwise/headwise) exhibit the lowest mean gating scores. Furthermore, the SDPA output gating score distribution shows a high concentration near 0, indicating substantial sparsity, consistent with its superior performance. (ii) **Head-Specific Sparsity Matters.** Enforcing shared gating scores across attention heads increases the overall gating scores and diminishes performance gains. Observations (i) and (ii) underscore the importance of *head-specific gating*, aligning with previous research demonstrating that individual attention heads capture distinct aspects of the input [46, 47, 48, 49].

(iii) **Query-Dependency Matters.** The scores for value gating (G_2) are higher than those for SDPA output gating (G_1), and the performance is inferior. This suggests that gating score sparsity is more effective when query-dependent rather than determined by the key and value. Specifically, SDPA output gating scores are derived from the hidden states corresponding to the current query (e.g. the Non-Linearity-Map in Eq 8 depends on X_i), whereas value gating scores are derived from hidden states associated with past keys and values (e.g. the Non-Linearity-Map in Eq 7 depends on each X_j). This implies that *gating score sparsity may filter out irrelevant contextual information for the query*. To further validate the importance of query-dependency, we introduce input-independent gating by zero-initializing learnable parameters ($q \times d_k$), applying a sigmoid function, and multiplying it with the SDPA output. As shown in row (6), input-independent gating improves upon the baseline, likely due to the introduction of non-linearity. Moreover, the high gating scores reinforce that effective sparsity should be input-dependent.

²Note that concatenating outputs from different heads and then multiplying with W_O is equivalent to multiplying each head’s output with its corresponding W_O^k before concatenation

Table 4: Performance of different gating methods with varying activation functions and average gate scores. ‘Act-Func’ refers to the activation function used for computing the gating scores, while ‘M-Act’ denotes the rounded maximum activation values of the hidden states output by each layer of the model. Additionally, ‘F-Attn’ represents the attention score of the first token, with higher values indicating more pronounced ‘attention sink’.

Method	Act-Func	Gate Score	M-Act	F-Attn	PPL	Hellaswag	MMLU	GSM8k
(1) Baseline	-	-	1053	0.467	6.026	73.07	58.79	52.92
(2) SDPA Elementwise Gate	Sigmoid	0.116	94	0.048	5.761	74.64	60.82	55.27
(3) SDPA Headwise Gate	Sigmoid	0.172	98	0.073	5.792	74.50	60.05	54.44
(4) SDPA Elementwise Head-shared Gate	Sigmoid	0.271	286	0.301	5.801	74.34	60.06	53.15
(5) v Elementwise Gate	Sigmoid	0.221	125	0.297	5.820	74.38	59.17	51.33
(6) SDPA Input Independent Gate	Sigmoid	0.335	471	0.364	5.917	73.64	59.02	52.40
(7) SDPA Elementwise Gate	NS-sigmoid	0.653	892	0.451	5.900	74.05	60.05	52.75

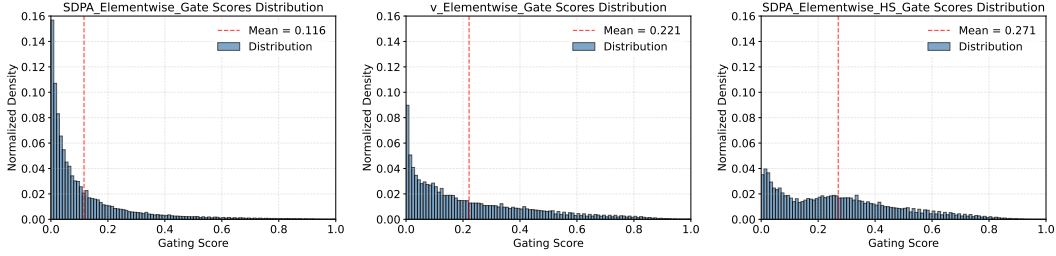


Figure 3: Gating score means and distributions for SDPA elementwise (Left), value Elementwise (Middle), and SDPA elementwise with head-shared gating (Right). Most gating scores are less than 0.5, indicating that the gating scores are sparse. Among them, the SDPA output gating score exhibits the strongest sparsity.

(iv) Less Sparse Gating is Worse. To further validate the importance of gating score sparsity, we reduce sparsity from the gating formulation. Specifically, we replace the sigmoid function with a modified Non-Sparse (NS) version:

$$\text{NS-sigmoid}(x) = 0.5 + 0.5 \cdot \text{sigmoid}(x), \quad (9)$$

which constrains the gating scores between [0.5, 1.0]. This ensures introducing non-linearity while removing gating score sparsity. As shown in Tab. 4 row (7), the gains of NS-sigmoid gating are inferior to those of SDPA output sigmoid gating. In Appendix A.3, we provide a more detailed discussion on how sparse gating scores affect the sparsity (the proportion of values below the threshold) in SDPA hidden states. We will discuss the impact of different sparsity levels on model behavior, including reducing the ‘attention sink’, in the next section.

4.3 SDPA Output Gating Reduces Massive Activation and Attention-Sink

Based on the observation that gating introduces sparsity to the SDPA output in an input-dependent manner, we hypothesized that this mechanism can filter out context irrelevant to the current query token, thereby mitigating the attention sink [8, 7]. To verify this, we analyze the distribution of attention scores (averaged over all heads) and the proportion of attention scores allocated to the first token (Fig. 2, Tab. 4, ‘F-Attn’ column). Massive activations [7] (large values in hidden states) are belied to lead to the attention sink to their corresponding tokens. Inspired by this, we also compute the mean of the maximum hidden state activations across layers, as shown in the ‘M-Act’ column of Tab. 4. More detailed layer-wise results are provided in the Appendix A.4.

We can observe: **(i)** Head-wise and element-wise query-dependent sigmoid gating at the SDPA output (G_1) largely reduces the attention score allocated to the first token and decreases massive activations. **(ii)** Enforcing shared gating scores across heads or applying gating only after the value projection (G_2) decreases massive activations, but does not reduce attention scores to the first token. This reinforces the importance of head-specific gating and suggests that *massive activations are not a necessary condition for attention sinks*. **(iii)** Reducing the input-dependence of gating (row 6) or using NS-sigmoid to reduce sparsity (row 7) intensifies both massive activations and attention sink.

Collectively, these observations indicate that *input-dependent, head-specific gating of the SDPA output introduces significant sparsity, thereby mitigating the attention sink*. Furthermore, sparsity in the SDPA outputs reduces massive activations within the model, with increased sparsity leading to

smaller activations. *This may explain the improved training stability with gating: by reducing massive activations, the model is less susceptible to numerical errors during BF16 training [50].* We also observe that massive activations originate primarily from early layers (e.g., layer 5), where the FFN outputs large values, consistent with [51]. Once added to the residual stream, these activations are propagated through subsequent layers via the pre-norm mechanism. This aligns with the effectiveness of sandwich normalization [43] in enhancing training stability (Table 2, row 7): applying LayerNorm to the FFN output prevents these large activations from entering the residual stream.

4.4 SDPA Output Gating Facilitates Context Length Extension

Based on the attention-sink-free pattern, we evaluate the SDPA gating’s effect in the long-context setting. Specifically, we extend the context length for the models trained on 3.5T tokens. We increase the RoPE [52] base

Table 5: Performance of different methods across varying sequence lengths. ‘YaRN Extended’ indicates the expanded context length variant. ‘(values)’ indicate the performance declines after extending the context length.

Method	4k	8k	16k	32k	64k	128k
Baseline	88.89	85.88	83.15	79.50	-	-
SDPA-Gate	90.56	87.11	84.61	79.77	-	-
YaRN Extended						
Baseline	82.90(-6.0)	71.52(-14.4)	61.23(-21.9)	37.94(-41.56)	37.51	31.65
SDPA-Gate	88.13(-2.4)	80.01(-7.1)	76.74(-7.87)	72.88(-6.89)	66.60	58.82

from 10k to 1M and continue training on data with a sequence length of 32k for an additional 80B tokens. This gives us models with a context length of 32k. Subsequently, we use YaRN [53] to extend the context length to 128k. We evaluate models on the RULER benchmark [23] and summarize results in Tab. 5. We observe the following: **(i)** Under the 32k setting, models with gating slightly outperform the baseline. This suggests that within the training length, the attention sink phenomenon may not hurt the model’s long-context performance. **(ii)** When the context length is extended to 128k using YaRN, both the baseline and gated models experience a decline within the original 32k range. This observation is consistent with previous works on extending context length by modifying RoPE [54, 53, 55]. Even though the decline is less pronounced for models with gating. **(iii)** At context lengths of 64k and 128k, the gated attention models outperform the baseline significantly. From these observations, we hypothesize that adding gating helps the model adapt to the context-length extension. A possible explanation is that baseline models rely on attention sinks to adjust the distribution of attention scores. [55] derives the effects of changing the RoPE based on the attention and hidden state distributions. When techniques like YaRN are applied to modify the RoPE base, the attention sink pattern may struggle to adapt in a training-free manner, leading to a noticeable drop in performance. In contrast, models with gating primarily rely on input-dependent gating scores to control information flow, making them more robust to such changes.

5 Related Works

5.1 Gating in Neural Networks

Gating mechanisms have been widely adopted in neural networks. Early works such as LSTMs [1] and GRUs [9] introduce gates to regulate information flow across time steps, addressing gradient vanishing/exploding issues by selectively retaining or discarding information. Highway Networks [2] extend this concept to feedforward networks, enabling the successful training of very deep architectures. SwiGLU [24] introduce gating mechanisms into transformer FFN layers, enhancing their expressive power and becoming a standard component in many open-source LLMs [56, 36].

Several works on state-space models [3, 10, 57] and Linear Attention, such as FLASH [4], RetNet [13], Lightning Attention [16, 58, 59], and Gated Delta Networks [17], also incorporate gating modules to control information of token-mixer modules. AlphaFold2 [5] and Forgetting Transformer [6] introduce gating mechanisms to the output of softmax attention. Some works [60, 12, 14, 15, 61] also apply operations similar to gating to augment softmax attention. Attention on Attention (AoA) [11] also modulates the attention output with a sigmoid gating, depending on the query. Although these works demonstrate the effectiveness of gating, a comprehensive understanding of its precise mechanisms and the reasons behind its effectiveness still needs exploration. This could contribute to a broader appreciation of gating’s importance beyond RNNs and facilitate designs that better leverage gating’s unique advantages. For example, while Switch Heads [19, 18], NSA [20], and MoSA [62]

employ sigmoid-based gating [63] for selection, further investigation into isolating gating’s specific contribution could offer valuable insights. Comparisons with baselines incorporating similar gating mechanisms in standard transformers could offer a more refined perspective on the effectiveness of their proposed selection mechanisms. The work most closely related to ours is Quantizable Transformers [64], which also finds that applying gating in softmax attention alleviates extreme attention concentration and outliers in hidden states in encoder models like BERT and ViT. While this work primarily leverages gating to eliminate outliers for model quantization, we provide a detailed analysis of various gating variants, uncovering their benefits through enhanced non-linearity and sparsity, as well as improved training stability. Building on these insights, we scale up gated attention models, demonstrating gating’s broad applicability and impact.

5.2 Attention Sink

StreamingLLM [8] formally identifies ‘attention sink’, in which specific tokens receive large attention scores. Similarly, in ViT, some redundant tokens act as ‘registers’ to store attention scores [65]. Later, Massive Activation [7] shows that excessive attention scores are assigned to tokens associated with massive activation values. However, our work reveals that value output (G_2) gating eliminates massive activations, yet attention sinks persist, indicating that massive activations are necessary for attention sinks. Similarly, attention sinks are characterized as non-informative ‘key biases’ that store redundant attention scores, arguing that softmax’s inherent normalization dependency drives this behavior [22]. Experimental attempts to modify softmax attention, such as replacing softmax with unnormalized sigmoid attention [66, 22], adding softmax attention gate or clip [64], calibrating attention scores [67], and modifying softmax computation [68] and denominator [69], show promise in mitigating attention sinks. Another stream of works try to move the sink tokens from input tokens to manually added components, like ‘registers’ [65], ‘meta tokens’ [70] and learnable ‘sink’ [71]. Our work shows that sparse gating after SDPA eliminates attention sinks in both dense (1B-parameter) and MoE (15B-parameter) models, even when trained on 3.5T tokens. Furthermore, we uncover the potential of eliminating attention sinks to benefit context-length extension.

6 Conclusion and Limitations

This work systematically investigates gating mechanisms in softmax-attention, revealing their significant impact on performance, training stability, and attention dynamics. This simple mechanism enhances non-linearity, introduces input-dependent sparsity, and eliminates ‘attention sink’. Additionally, gating facilitates context length extension, allowing models to generalize effectively to longer sequences without retraining. We will release the ‘attention-sink-free’ models, providing a foundation for future research into attention mechanisms.

The broader implications of non-linearity on the dynamics of attention and the overall training process remain under-explored. We don’t provide a theoretical explanation for how attention sinks influence the model’s ability to generalize to longer sequences.

References

- [1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [2] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
- [3] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- [4] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9099–9117. PMLR, 2022.

- [5] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
- [6] Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. *arXiv preprint arXiv:2503.02130*, 2025.
- [7] Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 2024.
- [8] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.
- [9] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.
- [10] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
- [11] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *International Conference on Computer Vision*, 2019.
- [12] Lanqing Xue, Xiaopeng Li, and Nevin L Zhang. Not all attention is needed: Gated attention network for sequence data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6550–6557, 2020.
- [13] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023.
- [14] Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. Improving transformers with dynamically composable multi-head attention. *arXiv preprint arXiv:2405.08553*, 2024.
- [15] Xuechen Zhang, Xiangyu Chang, Mingchen Li, Amit Roy-Chowdhury, Jiasi Chen, and Samet Oymak. Selective attention: Enhancing transformer through principled context control. *Advances in Neural Information Processing Systems*, 37:11061–11086, 2024.
- [16] Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024.
- [17] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.
- [18] Robert Csordas, Kazuki Irie, Jurgen Schmidhuber, Christopher Potts, and Christopher D Manning. Moeut: Mixture-of-experts universal transformers. *arXiv preprint arXiv:2405.16039*, 2024.
- [19] Robert Csordas, Piotr Piekos, Kazuki Irie, and Jurgen Schmidhuber. Switchhead: Accelerating transformers with mixture-of-experts attention. *Advances in Neural Information Processing Systems*, 37:74411–74438, 2024.
- [20] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.
- [21] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [22] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. *arXiv preprint arXiv:2410.10781*, 2024.

- [23] Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekish, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What’s the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.
- [24] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- [25] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.
- [26] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.
- [27] Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models, 2025.
- [28] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.
- [29] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- [30] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- [31] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [32] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [33] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.
- [34] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [35] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmu: Measuring massive multitask language understanding in chinese, 2023.
- [36] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- [37] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

- [38] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers, 2022.
- [39] Francesco D’Angelo, Maksym Andriushchenko, Aditya Vardhan Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *Advances in Neural Information Processing Systems*, 37:23191–23223, 2024.
- [40] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.
- [41] Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*, 2023.
- [42] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [43] Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. Cogview: Mastering text-to-image generation via transformers, 2021.
- [44] Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. *arXiv preprint arXiv:2410.05258*, 2024.
- [45] Guido Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks, 2014.
- [46] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.
- [47] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE, 2021.
- [48] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
- [49] Zekun Wang, Jingchang Chen, Wangchunshu Zhou, Haichao Zhu, Jiafeng Liang, Liping Shan, Ming Liu, Dongliang Xu, Qing Yang, and Bing Qin. Smarttrim: Adaptive tokens and attention pruning for efficient vision-language models. *arXiv preprint arXiv:2305.15033*, 2023.
- [50] Stanislav Budzinskiy, Wenyi Fang, Longbin Zeng, and Philipp Petersen. Numerical error analysis of large language models. *arXiv preprint arXiv:2503.10251*, 2025.
- [51] Itay Yona, Ilia Shumailov, Jamie Hayes, Federico Barbero, and Yossi Gandelsman. Interpreting the repeated token phenomenon in large language models. *arXiv preprint arXiv:2503.08908*, 2025.
- [52] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- [53] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.
- [54] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023.
- [55] Zican Dong, Junyi Li, Jinhao Jiang, Mingyu Xu, Wayne Xin Zhao, Bingning Wang, and Weipeng Chen. Longred: Mitigating short-text degradation of long-context large language models via restoration distillation. *ArXiv*, abs/2502.07365, 2025.

- [56] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [57] Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Stuffed mamba: Oversized states lead to the inability to forget. *arXiv preprint arXiv:2410.07145*, 2024.
- [58] Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Various lengths, constant speed: Efficient language modeling with lightning attention. *arXiv preprint arXiv:2405.17381*, 2024.
- [59] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.
- [60] Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou. Talking-heads attention. *arXiv preprint arXiv:2003.02436*, 2020.
- [61] Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Multi-token attention. *arXiv preprint arXiv:2504.00927*, 2025.
- [62] Piotr Piękos, Róbert Csordás, and Jürgen Schmidhuber. Mixture of sparse attention: Content-based learnable sparse attention via expert-choice routing, 2025.
- [63] Robert Csordas, Kazuki Irie, and Jurgen Schmidhuber. Approximating two-layer feedforward networks for efficient transformers. *arXiv preprint arXiv:2310.10837*, 2023.
- [64] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36:75067–75096, 2023.
- [65] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [66] Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, et al. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint arXiv:2409.04431*, 2024.
- [67] Zhongzhi Yu, Zheng Wang, Yonggan Fu, Huihong Shi, Khalid Shaikh, and Yingyan Celine Lin. Unveiling and harnessing hidden attention sinks: Enhancing large language models without training through attention calibration. *arXiv preprint arXiv:2406.15765*, 2024.
- [68] Zayd M. K. Zuhri, Erland Hilman Fuadi, and Alham Fikri Aji. Softpick: No attention sink, no massive activations with rectified softmax, 2025.
- [69] Evan Miller. Attention is off by one, 2023.
- [70] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, et al. Hymba: A hybrid-head architecture for small language models. *arXiv preprint arXiv:2411.13676*, 2024.
- [71] OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025.

A Supplement Material

A.1 Switch Head Baselines

In this section, we present detailed experiments related to Switch Heads. The Switch Head paper demonstrates that introducing sparse activation in attention—where each token selects the top- k experts from a pool of key/value/output experts via learnable sigmoid routing—enables the model to achieve comparable results to the baseline. This suggests that, within the Switch Head framework, both expert parameters and activated parameters are beneficial, with more being better under the same total parameter budget.

Table 6: Performance of different switch head methods with varying parameter additions and configurations. ‘switch kv’ and ‘switch v’ refer to introducing selective computing in key-value and value components, respectively. ‘Switch kv, 8top8’ means there are 8 key and value map experts, and each token select top8 experts. Notice ‘Switch v, 1top1’ is equivalent to v Headwise Gate in Tab. 1 row (11).

Method	Added Param (M)	PPL	MMLU	GSM8k	Hellaswag	C-eval
(1) Baseline (q32, kv4)	-	6.026	58.79	52.92	73.07	60.26
(2) Switch kv, 8top8	38	5.847	59.17	52.54	73.32	61.01
(3) Switch kv, 4top4	13	5.935	58.14	53.27	73.75	59.67
(4) Switch v, 4top4	13	5.820	59.02	52.77	73.34	61.74
(5) Switch v, 8top2	25	5.870	59.10	53.53	74.17	62.34
(6) Switch v, 1top1	3	5.808	59.32	53.53	74.38	62.61

Looking at the results in Tab. 6, we observe an interesting trend: while increasing the number of activated kv experts (with the same expert parameter settings) appears to offer some improvement in PPL (row 4 vs. 5), the gains in overall benchmark performance are less pronounced. Notably, the best results for both benchmark scores and PPL were achieved by ‘Switch v 1top1’ (row 6), which, as mentioned earlier, is analogous to applying sigmoid gating directly to the output of the value layer. These findings raise an intriguing question about the primary driver of the performance improvements observed in these experiments. It suggests that the introduction of gating itself plays a significant role in the effectiveness of this approach.

A.2 More Discussion on Sparse Gating Score

In our experiments, we evaluate three distinct LLM architectures: two dense variants with 1.7 billion parameters and one sparse MoE model with an effective size of approximately 15 billion parameters (denoted as 15A2B). The two dense models differ primarily in depth and hidden dimension: the *1.7B-28 layer* model uses 28 transformer layers with a hidden size of 2048, while the *1.7B-48 layer* variant employs 48 layers but a reduced hidden size of 1536 to maintain a comparable parameter count. Both dense models tie input and output embeddings and apply query-key normalization for training stability. Full architectural hyperparameters are summarized in Table 7. To support reproducibility and future research, we plan to open-source both 1.7B dense models.

A.3 More Discussion on Sparse Gating Score

In this section, we analyze the impact of gating score sparsity on attention output. First, we examine the mean values of SDPA output before and after applying gating to the hidden states. Specifically, we calculated the mean absolute values of Y and Y' before and after G_1 at each layer, as shown in Fig. 4. We also included results from a baseline without gating for comparison. The results indicate that: (1) after gating, the mean value of hidden states decreased from 0.71 to 0.05, corresponding to the generally small gating scores; (2) the gated hidden states closely resemble the baseline, suggesting that gating might serve a similar function as attention sink in filtering out irrelevant information.

We further analyze the proportion of hidden states below certain thresholds before and after gating, as shown in Fig 5. The results reveal that: (1) after gating, the sparsity in hidden states significantly increases across different thresholds. Since the mean gating scores are already small, multiplying hidden states by a small number naturally pushes some values below the threshold. Therefore, (2) we further multiply the pre-gating hidden states by the average gating score and observed that the increase in sparsity is smaller than with original gating. This suggests that sparse gating scores enhance sparsity in hidden states.

Table 7: Architectural specifications of the target LLMs used in our experiments. The 15A2B model is a Mixture-of-Experts (MoE) architecture with 128 experts and top-8 routing. All models use a head dimension of 128 and apply query-key normalization. Embedding weights are tied in the dense models but not in the MoE model.

Model	1.7B-28 layer	1.7B-48 layer	15A2B MoE
Layers	28	48	24
Query Heads	16	16	32
Key / Value Heads	8	8	4
Head Dimension	128	128	128
Tie Embedding	Yes	Yes	No
QK Normalization	Yes	Yes	Yes
Hidden Size	2048	1536	2048
FFN Size	6144	4608	768
Number of Experts	—	—	128
Top- k	—	—	8

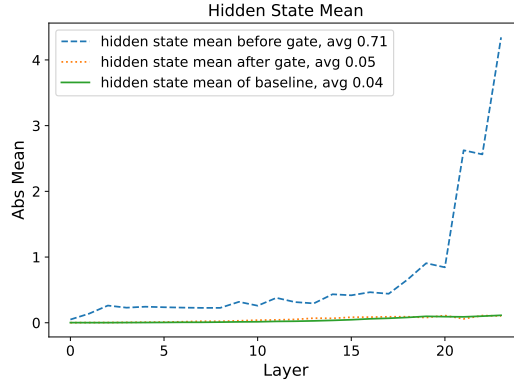


Figure 4: Mean absolute values before and after gating. The baseline and post-gating values are similar.

A.4 Layerwise Massive Activations and Attention Sinks

In this section, we compare and analyze the presence of massive activations and attention sinks (the attention score of the first token) within the model. From the results, we observe the following:

For the baseline (row 1), the output of the 6th layer’s FFN contains massive activations, which are subsequently added to the residual stream, causing large activations to persist in the residuals of subsequent layers. Correspondingly, significant attention sink phenomena emerge starting from the 6th layer. After applying gating to the SDPA output (row 2), the outputs of the earlier layers in the network remain relatively small overall, with massive activations growing gradually as the layer depth increases. Notably, no significant attention sink phenomenon is observed in any layer of the network.

When gating is applied only at the value layer (row 3), the model exhibits massive activations similar to row 2. However, a certain degree of attention sink phenomenon persists. This indicates that massive activations are not a necessary condition for the emergence of attention sinks. When enforcing shared gating scores across different heads (row 4) or modifying the activation function of gating to suppress sparsity (row 5), the sparsity introduced by gating is reduced. In these cases, both massive activations and attention sinks become comparable to those observed in the baseline.

These observations suggest that introducing sufficient sparsity within the attention mechanism may help mitigate the occurrence of massive activations. However, further investigation is needed to fully understand the interplay between sparsity, massive activations, and attention sinks, particularly in the context of scaling to deeper and larger models.

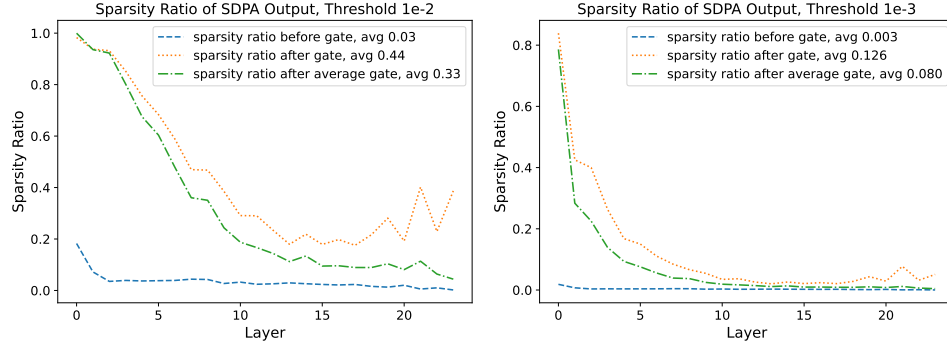


Figure 5: Proportion of SDPA output values below threshold after gating (Left: 1e-2, Right: 1e-3). We also include sparsity measurements obtained by multiplying the average gating score with pre-gating hidden states.

A.5 More Layerwise Gating Scores

In this section, we analyze the distribution of gating scores under two additional constraints while using SDPA output gating as the baseline (row 1, elementwise/headwise): (1) enforcing the same gating score across different heads (row 2, left), and (2) restricting the minimum value of the gating scores (row 2, right). When enforcing shared gating scores across different heads, the gating scores for most layers increase. This indicates that different heads require different sparsity, highlighting the importance of head-specific gating mechanisms.

A.6 Other Attempt to Stabilize Training

We observe that both the addition of sandwich normalization [43] and gating mechanisms eliminate massive activations while improving training stability. This prompts us to explore whether simpler methods could prevent large activations within residuals. Specifically, we introduce a clipping operation to constrain the outputs of attention and FFN layers before they enter the residual connection, limiting their values to the range $(-clip, clip)$. However, we find that regardless of whether the clip value was set to 300 or 100, the model still encounters convergence issues at a learning rate of $8e-3$. This suggests that the instability in pre-norm model training is not solely due to large activations within residuals. It is likely that any layer producing large outputs can lead to stability problems, indicating the need for further investigation into the root causes of training instability.

A.7 Adding Gating in Continue Training

We also experiment with incorporating an attention output gating mechanism during the continued training phase. However, we find that this approach neither mitigates the massive activations and attention sinks already present in the model nor significantly affects the final performance. We believe this is largely because the effectiveness of gating mechanisms stems primarily from their effects on the model’s training dynamics, and thus their impact is limited when applied to models that are not trained from scratch.

A.8 Broader Impacts

This work focuses on improving the efficiency, stability, and context-handling capabilities of LLMs. The potential positive societal impacts include: **Improved Accessibility:** More efficient and stable training methods can lower the computational cost of developing and deploying LLMs, potentially making them more accessible to researchers and organizations with limited resources. **Enhanced Performance in Long-Context Applications:** The ability to handle longer contexts more effectively can lead to improvements in applications such as document summarization, question answering, and code generation, where understanding relationships across extended text is crucial. **Advancements in Model Scalability:** Our findings offer insights and practical guidance for designing advanced models, contributing to ongoing efforts to scale LLMs while maintaining or improving their performance and stability.

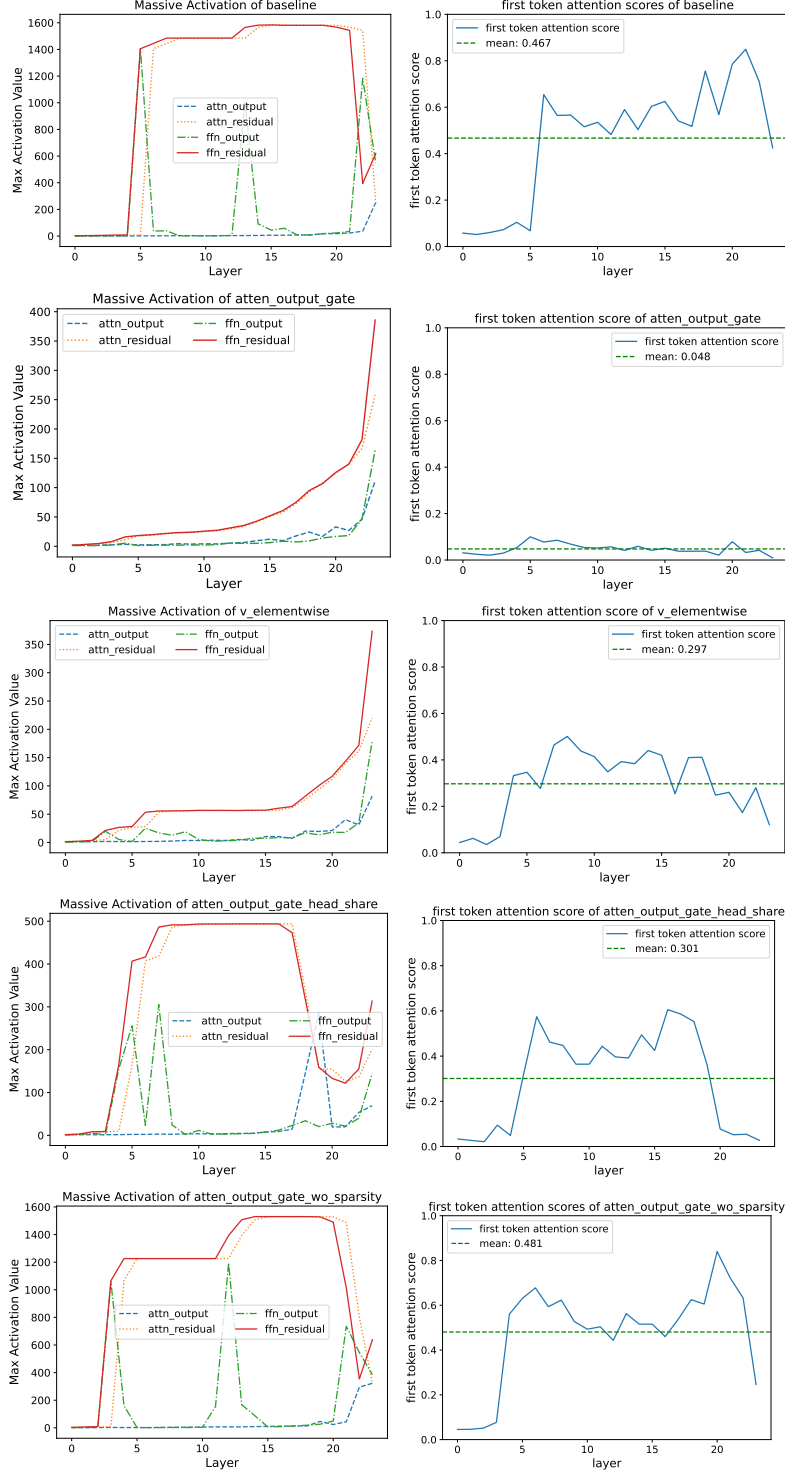


Figure 6: Comparison of massive activations and attention sink phenomena across different gating configurations. Row 1 (Baseline): Significant massive activations and attention sinks emerge after the 6th layer. Row 2 (SDPA Gating): Reduced activations and no attention sinks observed. Row 3 (Value Layer Gating): Similar activations to Row 2 but with residual attention sinks. Rows 4–5 (Reduced Sparsity via cross-head share and NS-sigmoid): Massive activations and attention sinks resemble the baseline.

A.9 Licenses for Existing Assets

We conduct experiments based on Megatron-LM. We acknowledge the original authors and contributors of Megatron-LM and respect their licensing terms. The specific license terms are available at

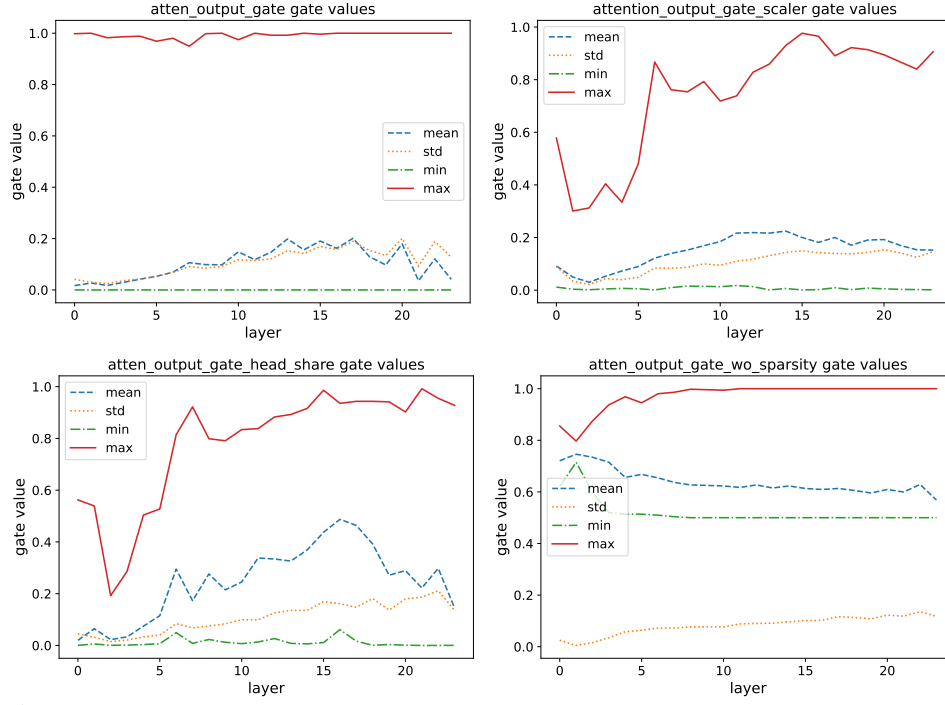


Figure 7: Distribution of gating scores under different constraints for SDPA output gating variants.

<https://github.com/NVIDIA/Megatron-LM/blob/main/LICENSE>. Any released code implementing our proposed gated attention mechanism will be released under the MIT License, allowing for open research and further development.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We elucidate our contributions in the Abstract and Intrudctions.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss the limitations in Sec 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: Our study is mainly empirical exploration.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We report our detailed training settings in Sec 3.2.1 and Sec 3.2.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide a Pytorch version implementation for the main method in the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We report our detailed training settings including hyperparameters in Sec 3.2.1 and Sec 3.2.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: LLMs pre-training consumes a significant amount of computational resources, making it impractical to conduct multiple experiments to obtain error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We discussion computation cost in Sec 3.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have read this code.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts in Appendix A.8.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: Our study is an empirical exploration and the codes and models we release are intended solely for further research and are not meant for direct industrial application.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We discuss this in Appendix A.9.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: Related codes will be released with suitable license.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: This work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: This work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: LLM is used only for writing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.