Softpick: No Attention Sink, No Massive Activations with Rectified Softmax

Anonymous ACL submission

Abstract

We introduce softpick, a rectified, not sum-to-one, drop-in replacement for softmax in transformer attention mechanisms that eliminates attention sink and massive activations. Our experiments with 340M and 1.8B parameter models demonstrate that softpick achieves 0% sink rate consistently. The softpick transformers produce hidden states with significantly lower kurtosis and creates sparse attention maps. Quantized models using softpick outperform softmax on standard benchmarks, with a particularly pronounced advantage at lower bit precisions. Our analysis and discussion shows how softpick has the potential to open new possibilities for quantization, low-precision training, sparsity optimization, pruning, and interpretability.

1 Introduction

The softmax function is widely used in statistics and particularly in machine learning as a way to normalize a vector of real numbers into a probability distribution. It has since been adopted as the de facto function to normalize the scores in the attention mechanism (Bahdanau et al., 2015; Sukhbaatar et al., 2015) used in the transformer architecture (Vaswani et al., 2017). The use of softmax in attention was a natural choice, since it represents the probability of a query "matching" to a key among many keys, which can be used to return values each weighted by that probability. However, our modern use of attention makes us question this intuition (Miller, 2023; Smith, 2025): why must they be non-zero probabilities that sum to one?

We continue to use softmax in practice because it is still the most capable normalization for attention. Softmax has great training stability with a neat Jacobian matrix resulting in a dense gradient, bounded Frobenius norm, and built-in regularization (Saratchandran et al., 2024), and has been proven to be sufficiently expressive in its nonlinearity (Chiang, 2024). Despite the benefits, softmax brings several oddities detrimental to language modeling. Its sum-to-one nature forces out a now well-observed behavior named attention sink (Xiao et al., 2024) where attention heads allocate a significant score towards a specific token, often the initial BOS token, which itself is semantically irrelevant (Gu et al., 2024). These sinks are, as far as we know, harmless to downstream performance. However, another symptom of this same effect of softmax is the appearance of extreme hidden state activations, dubbed massive activations (Sun et al., 2024), which grow larger as the model scales (Bondarenko et al., 2023). These activations are especially problematic for quantization, with entire algorithms built around their existence. Additionally, most of the current low-precision training approaches try to work around these massive activations as they are a

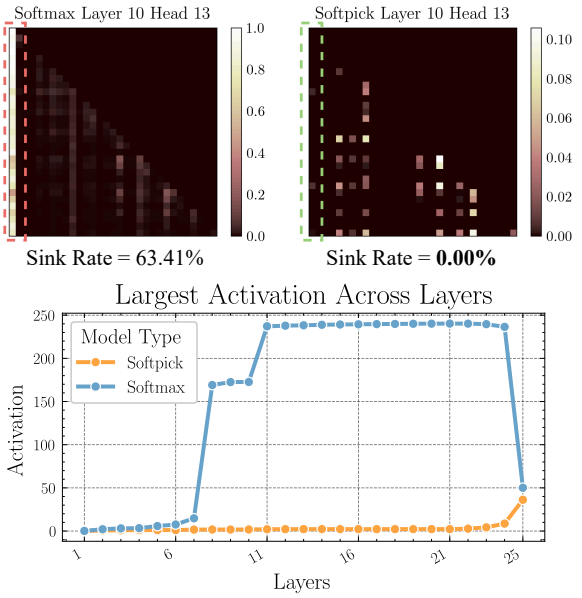


Figure 1: (Top) Comparison between the attention maps when using softmax vs softpick and overall sink rate of the 340M models. (Bottom) Largest hidden state activation per layer of the 340M models.

hassle to represent in low-bit precision. Removing these massive activations would open up a lot more possibilities.

Even with these quirks, softmax remains widely used in attention due to its stability and effectiveness. In this paper, we propose the softpick function in an attempt to find a normalization function more fitting for attention and its use in transformers. The softpick function mitigates both attention sink and massive activations by reshaping the softmax function to avoid strict sum-to-one behavior and allowing rectified outputs. Our contributions are as follows:

1. We propose the softpick function as a drop-in replacement to softmax in attention.

2. We experiment by training 340M and 1.8B parameter transformer models from scratch and show how softpick compares to softmax in benchmarks and training behavior. We also show that quantized softpick models down to 2-bit precision outperform softmax.

3. We analyze the resulting attention maps, attention sinks, and massive activations, or lack thereof. We show that softpick returns more legible attention maps, a sink rate of 0%, and no massive activations in the hidden states.

2 Background

Attention sink was introduced by Xiao et al. (2024) to describe heads that allocate disproportionate attention to semantically weak but frequent tokens, most commonly the BOS token. Barbero et al. (2025) argue that such sink-like heads can mitigate over-mixing by acting as approximate no-ops, while Gu et al. (2024) systematically study correlates of the phenomenon and report that sinks appear broadly in sufficiently trained transformer language models. Together, these results suggest that sink behavior is not an artifact of a specific setup, but is closely tied to attention normalization, in particular the sum-to-one constraint imposed by softmax.

Massive activations, introduced by Sun et al. (2024) as rare but extremely large hidden-state values, pose a challenge for quantization and low-precision training (Dettmers et al., 2022). Sun et al. (2024) further link these outliers to self-attention and softmax-driven sink behavior, motivating mitigation strategies that relax strict normalization. For example, Gu et al. (2024) show that removing normalization (e.g., sigmoid attention) reduces sinks, but sinks reappear once normalization is reintroduced. Appended-vector / KV-bias variants similarly require additional learned parameters and can degrade with depth (Gu et al., 2024; Sun et al., 2024). Softmax-1 relaxations add a constant to the denominator (Miller, 2023) and are evaluated in Kaul et al. (2024), while OpenAI et al. (2025) adopt a learnable-bias variant at scale to relax strict sum-to-one behavior. However, Owen et al. (2025) find KV-bias techniques are not a reliable mitigation for massive activations, and architectural alternatives such as gated attention trade additional parameters and computation for reduced sinks and outliers (Qiu et al., 2025). We propose softpick as a drop-in alternative that targets both sinks and massive activations without adding parameters or requiring custom optimizers.

3 Method

3.1 Softpick Function

Given a vector $\mathbf{x} \in \mathbb{R}^N$, we define the softpick function as

$$\text{Softpick}(\mathbf{x})_i = \frac{\text{ReLU}(e^{x_i} - 1)}{\sum_{j=1}^{N} |e^{x_j} - 1|} \tag{1}$$

with the rectified linear unit defined as $\text{ReLU}(x) = max(x, 0)$ and the absolute value function written as $|x|$. Similar to softmax, we need a numerically safe version to use in practice. We define numerically safe softpick as

$$\text{Softpick}(\mathbf{x})_i = \frac{\text{ReLU}(e^{x_i - m} - e^{-m})}{\sum_{j=1}^{N} |e^{x_j - m} - e^{-m}| + \epsilon} \tag{2}$$

where m is the maximum value inside $\mathbf{x}$ and $0 < \epsilon \ll 1$ is to avoid division by zero in the case that all inputs are exactly zero. This function is a drop-in replacement to softmax in the attention mechanism:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softpick}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{3}$$

3.2 Design Rationale

To understand the reasoning behind each component of the softpick function, it is best we go through a step-by-step recreation of the formula. We start from the vanilla softmax function.

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{4}$$

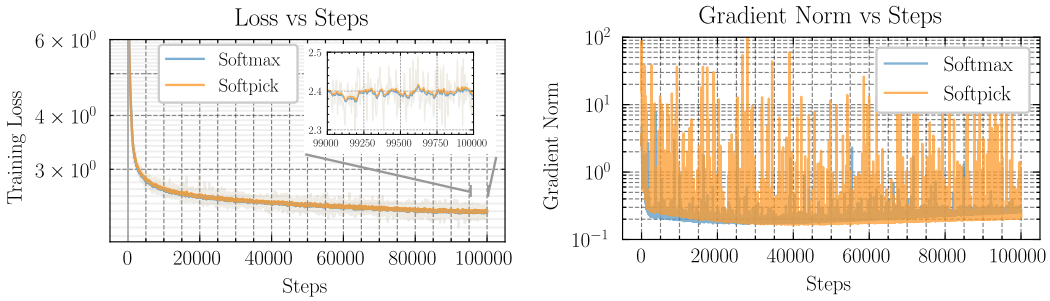


Figure 2: Training loss and gradient norm during training of 340M models.

The reason why we still do not modify this function too much is that the characteristics of softmax in training are ideal for attention. We want to maintain as much of the Jacobian matrix as possible and keep the gradient norm bounded that way. The next step would be to induce "null attention", or allowing attention to output zeros. We can achieve this by moving down the exponential by one, making it output negative values $-1 < y < 0$ for negative inputs. We then rectify this output with ReLU.

$$\text{Softmax}(\mathbf{x})_i = \frac{\text{ReLU}(e^{x_i} - 1)}{\sum_{j=1}^{N} \text{ReLU}(e^{x_j} - 1)} \quad (5)$$

This function has several desirable properties. First, it allows zero-valued outputs, which has several benefits in attention such as sparsity optimizations and minimizing noise from the accumulation of unneeded values. Second, unlike softmax, the denominator does not grow unbounded as N grows to infinity. This allows for a sharp output distribution regardless of sequence length, where only the positive inputs receive a score that sum to one. However, this is also its weakness. We experimented with this rectified-only softmax and found that it diverges from baseline softmax after a longer training period, and hypothesize that this is caused by heads "dying" and not being able to recover from outputting only zeros. This is obvious when we look at the Jacobian of the function, where negative inputs do not receive any gradients because they do not contribute to the output nor the denominator. Moreover, using this modification does not result in getting rid of attention sink. We go into more detail on this and show the experimental results of using a similar function in Appendix C. To fix the issue of negative inputs receiving minimal gradients, we allow them to contribute to the denominator.

$$\text{Softmax}(\mathbf{x})_i = \frac{\text{ReLU}(e^{x_i} - 1)}{\sum_{j=1}^{N} \text{Abs}(e^{x_j} - 1)} \quad (6)$$

The absolute function $\text{Abs}(x)$ or $|x|$ allows us to rectify each element for summation, resulting in a positive sum, but does not alter the derivative of e^x other than its sign, i.e. $\frac{d}{dx}|e^x| = -e^x$ for $x < 0$. This is crucial to one of our goals, which is to maintain the desired properties from the Jacobian of softmax. This way, gradients can flow even when the input to the function is negative. More importantly, this asymmetry between the numerator and denominator removes the strict requirement to sum to one, which is the main cause of attention sink.

3.3 Derivative

Given $\mathbf{s} = \text{Softpick}(\mathbf{x}) \in \mathbb{R}^N$ as the output, the partial derivative or elements of the Jacobian matrix of the (numerically safe) softpick function can be written as such:

$$\frac{\partial s_i}{\partial x_j} = \frac{e^{x_j - m}}{\Sigma} \left(\delta_{ij} \, \text{step}(x_i) - \text{sign}(x_j) s_i\right) \quad (7)$$

where δ_{ij} is the Kronecker delta, equal to 1 if $i = j$ and 0 otherwise, with $\text{step}(\text{x})$ being the step function that returns 0 if $x \leq 0$ and 1 if $x > 0$ and $sign(x)$ being the sign function that returns -1 if $x < 0$ and 1 if $x \geq 0$. Additionally, $\Sigma = \sum_{k=1}^{N} |e^{x_k - m} - e^{-m}| + \epsilon$ is the denominator of the softpick function. Although unlike softmax where only the outputs need to be kept for a naive backward pass implementation, the softpick derivative is nonetheless trivial for implementations that recompute the inputs, as is done in FlashAttention.

3.4 FlashAttention

The FlashAttention algorithm (Dao et al., 2022; Dao, 2024) allows for online single-pass computation of attention, bypassing the quadratic memory requirement. Because the $\text{ReLU}(x)$ and absolute $|x|$ functions uphold the multiplicative property

Task	Metric		340M			1.8B		
			Softmax	Softpick	Δ	Softmax	Softpick	Δ
Arc Easy	Acc Norm	↑	56.61	56.73	+0.12	67.21	62.04	-5.17
	Acc	↑	60.35	61.11	+0.76	72.73	68.60	-4.13
Lambada	Acc	↑	36.25	36.21	-0.04	49.43	43.51	-5.92
	Perplexity	↓	30.33	28.63	-1.70	11.38	15.81	+4.43
Piqa	Acc Norm	↑	66.59	66.49	-0.10	73.61	70.89	-2.72
	Acc	↑	66.97	66.59	-0.38	73.78	71.27	-2.51
Sciq	Acc Norm	↑	74.90	77.30	+2.40	86.40	80.40	-6.00
	Acc	↑	83.20	83.60	+0.40	90.10	87.00	-3.10
Wikitext	Word Perplexity	↓	23.85	24.32	+0.47	15.10	17.87	+2.77

Table 1: Comparison of softpick vs softmax on downstream tasks for 340M and 1.8B models. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

for positive-valued multipliers (such as e^x for any x), deriving an online version of softpick is possible, thus making it compatible with FlashAttention. We provide the algorithms for the more recent FlashAttention-2 version of the forward pass and backward pass that use the softpick function instead of softmax in Appendix A.

4 Experiments

To validate the viability of the softpick function over softmax, we conduct experiments on a-style (pre-norm, RoPE, SwiGLU MLP) transformers trained from scratch. We train 4 models, two for each size: 340M and 1.8B parameters, each with a softmax variant and a softpick variant. The detailed training configuration for all models is explained in Appendix B shown in Table 4. We use the flash-linear-attention repository (Yang and Zhang, 2024) and their Flame training framework (built on top of torchtitan (Liang et al., 2025)) because of their easily modifiable and fast triton (Tillet et al., 2019) kernels. We train each 340M model on a total of 52 billion tokens and each 1.8B model on 104 billion tokens sampled from the 100B subset of the fineweb-edu (Lozhkov et al., 2024) dataset. Each training run took approximately 18 hours for the 340M models and 116 hours for the 1.8B models on 8xH100 GPUs.

We also execute short training runs of models for analysis and comparison against other sink-mitigation / sink-free attention variants. Unless specified otherwise, we train these models on 2×A100 GPUs or 4×MI210 AMD GPUs. Each model uses the 340M configuration in Appendix B on Table 4. We intentionally limit these runs to 10k steps because attention sinks in the vanilla softmax baseline already emerge within this window, making it sufficient for diagnosing sink behavior and activation outliers while keeping the ablation compute manageable. As these models are minimally trained, we do not run benchmarks with them.

5 Results

Training Loss and Gradients We present the resulting training loss and gradient norm of the 340M model over 100,000 training steps (52B tokens) in Figure 2. We observed that the training loss of the softpick transformer tracks very closely to the vanilla softmax transformer, with only a 0.004 gap in the final training loss. This supports our design decision to maintain as much of the important components of the softmax function as possible. However, this does not hold at scale, as we observe a larger gap of 0.12 in the training loss of the larger 1.8B model. This will be reflected in the benchmark results. Tangentially, we observe that the total gradient norm of the models' parameters during training exhibits noteworthy behavior. At the start up to a third of the way into training, the gradient norm of the softpick model is higher than softmax, but goes down and stays below softmax until the end, and does not rise at as fast of a rate as softmax. Also important to note is that the magnitude of some gradient norms of the softpick model are much higher than the gradient norm peaks of softmax. We use gradient clipping with a maximum norm of 1.0 and do not see any instability caused by these large gradients.

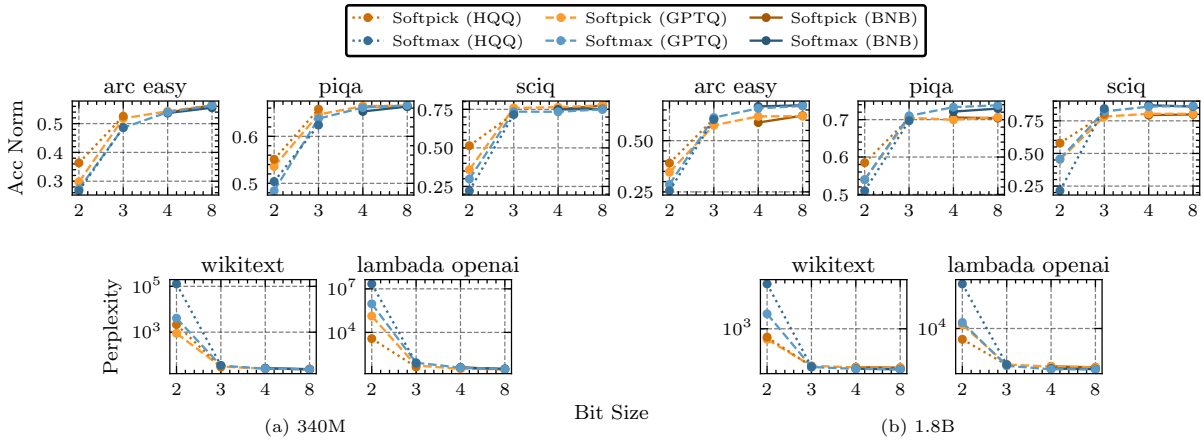


Figure 3: Quantization results of softmax vs. softpick across model scales. We deliver results on 2, 3, 4, and 8-bit quantization. Full tabular results in Appendix F and G.

Benchmarks We benchmark all models with general downstream tasks that are appropriate for the model sizes. We use the LM Evaluation Harness (Gao et al., 2023) to evaluate on these benchmarks: AI2 Reasoning Challenge (ARC-e) (Clark et al., 2018), Lambada (Paperno et al., 2016), specifically the OpenAI variant, PIQA (Bisk et al., 2020), and SciQ (Welbl et al., 2017). All 4 benchmarks report accuracy, with an addition of perplexity for Lambada. We also measure validation perplexity on Wikitext (Merity et al., 2017). See Table 1 for full results. We observe equal if not slightly better performance from softpick compared to softmax on the 340M model. However, the 1.8B models show that softpick does not seem to scale well, at least when using the same hyperparameters as softmax. Overall accuracy on benchmarks and perplexity are worse at this scale.

Quantization We also benchmark the effect of quantization on the trained models. We verified this using HQQ (Badri and Shaji, 2023), BitsandBytes (BNB) (Dettmers et al., 2022, 2023), and GPTQ (Frantar et al., 2022). These methods encompass different approaches, including those that utilize calibration data and those that do not. We use the same benchmarks as before. The results can be seen in Figure 3 with more complete tables in Appendix F. At the 340M scale, when using softpick, the quantized models are consistently better than when softmax is used, and quantization is less damaging as the precision goes down. This becomes less clear at the 1.8B as the gap in performance of the base models is larger. However, the gap does close at lower precisions.

Size	Model	ϵ_s=0.2	ϵ_s=0.3
340M	Softmax	68.28	63.41
	Softpick	0.00	0.00
1.8B	Softmax	41.73	14.96
	Softpick	0.00	0.00

Size	Model	Kurt.	Min.	Max.	Spars.%
340M	Softmax	33510.81	-207.55	240.27	4.53*
	Softpick	340.96	-45.03	36.21	99.34
1.8B	Softmax	74456.81	-173.48	371.43	4.28*
	Softpick	2193.27	-51.05	101.51	95.63

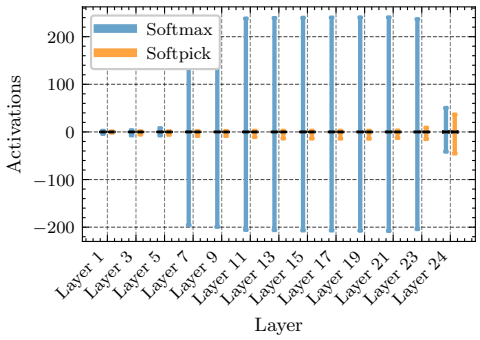
Table 2: Sink rate, activation statistics, and attention sparsity for softmax vs. softpick. Sparsity is the % of exact zeros in the causal (lower-triangular) attention matrix; for softmax, zeros arise only from numerical underflow.

6 Analysis

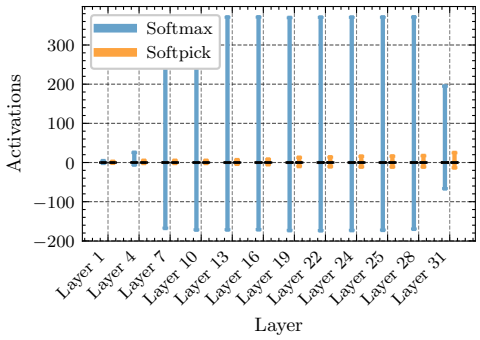
6.1 Attention Maps

To understand the difference that the softpick function makes compared to softmax, it is best to observe the attention maps directly. Attention maps are the matrices resulting from the computation of $\mathbf{A} = \text{Soft}\{\max, \text{pick}\}(\mathbf{QK}^T/\sqrt{d_k})$. We prepare three text inputs as samples to run through the models and extract the attention maps. We provide some clear example attention maps from the 340M models in Figure 5 and show more attention maps of all models in Appendix H.

First, it is clear that softpick behaves very differently in regards to how it normalizes attention scores compared to softmax. Due to the rectified numerator in softpick, scores are functionally sparse and concentrated into specific spots and re-



(a) 340M models



(b) 1.8B models

Figure 4: Box plots of the hidden state activations at some layers of the softmax (left, blue) and softpick (right, orange) models.

Model	ϵ_s=0.2	ϵ_s=0.3
Softmax	47.97	34.89
Softpick	0.02	0.00
Gated Attention	5.00	2.00
GPT-OSS Sink	19.34	8.78
Rectified	33.08	25.60
ReluSoftpick1	8.16	4.56
Scaled Softpick	0.01	0.00
Softmax+1	24.38	16.36

Model	Kurt.	Min.	Max.	Spars.%
Softmax	6452.72	-270.50	178.88	0.16*
Softpick	281.57	-59.06	25.36	92.74*
Gated Attention	135.62	-49.09	23.31	0.73*
GPT-OSS Sink	1800.34	-36.25	106.62	0.05*
Rectified	11054.50	-312.25	271.00	60.70*
ReluSoftpick1	452.59	-61.28	60.62	29.08*
Scaled Softpick	170.70	-53.69	30.81	91.95*
Softmax+1	574.16	-47.56	83.88	0.12*

Table 3: Sink rate, activation statistics, and attention sparsity for sink-free attention variants. Sparsity is the % of exact zeros in the causal (lower-triangular) attention matrix; for softmax, zeros arise only from numerical underflow.

gions, surrounded by actual zero scores. Although more general heads that attend to a wider range of tokens do still exist in the softpick model. Second, and more important to our thesis, is that attention sinks are nowhere to be seen with softpick.

To see the sparsity effect of softpick, we calculate the sparsity ratio of the attention map for softpick and softmax by running 10 text samples through the model, collecting the attention maps, and averaging the the sparsity, which is calculated by counting the number of exact zeros. Note that since this is a causal language model, we only consider the lower triangular scores of the attention map. As shown in Table 2, our method results in attention scores with 99.34% sparsity on the 340M model and 95.63% on the 1.8B model. While softmax in theory cannot return exact zeros, we observe a non-zero sparsity because in practice floating point scores can underflow to zero.

6.2 Attention Sink

Attention sink can be identified by the large scores on the first column in the attention map, which manifests as a bright vertical line leftmost on the attention heatmaps. Notably, the heads that have heavy attention sinks i.e. large scores on the BOS token and close to none on other tokens with softmax, become either specifically tuned to only score specific trigger tokens, or are completely shut off when not needed. An example of this can be seen in Figure 5, on head 4 of the 10th layer. The softmax model has a clear attention sink and barely attends to other tokens, while the softpick model completely shuts off the head (all zero scores) on the first input, and attends to some specific tokens on the second and third input sample text. We believe this behavior is a more concrete version of the active-dormant heads in vanilla attention, a previously observed phenomenon (Guo et al., 2024). We discuss this and its implications in §8.

In addition to visually observing attention sinks, we can use the sink rate metric proposed in (Gu et al., 2024), defined as $\frac{1}{L}\sum_{l=1}^{L}\frac{1}{H}\sum_{h=1}^{H}\mathbb{I}(\alpha_1^{l,h} > \epsilon_s)$ which is the percentage of heads in the transformer that on average has an attention score above some threshold ϵ_s in the first token, where $\alpha_1^{l,h}$ is the average value of the first column in $\mathbf{A}$ at layer l and head h. We calculate the sink rate of both models on 1000 samples from the validation set of SlimPajama (Soboleva et al., 2023) with $\epsilon_s = 0.3$ (as suggested in (Gu et al., 2024)) and also $\epsilon_s = 0.2$ and present the results in Table 2. The model using softpick has a sink rate of 0% for both thresholds and both model sizes, which means that no atten-

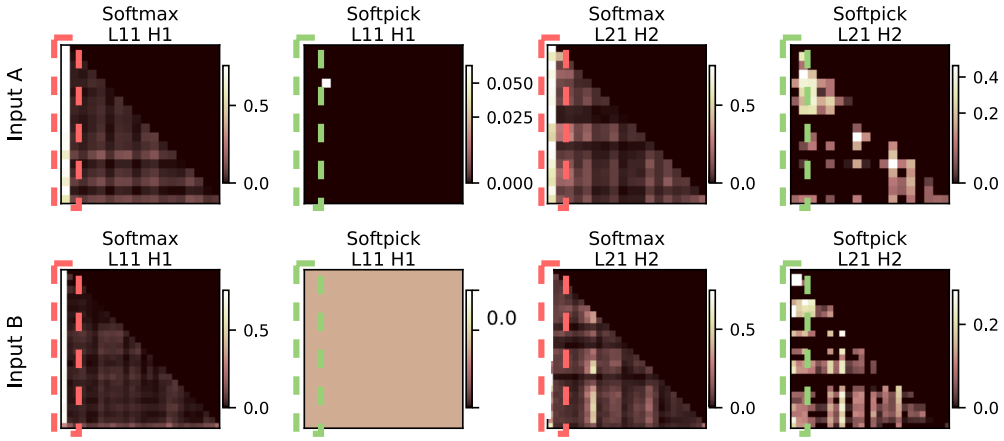


Figure 5: Attention maps of softmax and softpick 340M models on 2 different input texts. Two heads are visualized: Head 1 of Layer 11 and Head 2 of Layer 21. See more attention maps in Appendix H.

tion heads are overscoring the BOS token in any scenario, effectively eliminating attention sink completely. The effects of this can be seen when we analyze the hidden activations between transformer layers and the large values inside them.

6.3 Massive Activations

We analyze the effect of softpick on massive activations by examining the hidden state outputs after every transformer layer. We present some key metrics in Table 2 and box plots of the hidden state activation distribution of every some layers of the 340M and 1.8B models in Figure 4. These metrics are calculated by running 10 samples of text through the model and collecting all hidden state vectors to evaluate them all at once. The box plots visualize how large the massive activations of the softmax model are, making the boxes (Q1, median, and Q3) barely visible due to the large minimum and maximum values. This is especially prominent from the middle layers up to second to last layer. Meanwhile, the softpick models do not exhibit these massive activations. There is an exception in the last layer where it matches the softmax model, most likely due to the need to project back into the vocabulary space. This large difference can be seen clearly in the metrics as well. Kurtosis of all hidden state activations in the 340M model is significantly reduced from 33,510 to 340, a hundred-fold reduction. The minimum and maximum values are also reduced by an order of magnitude. Note that the minimum and maximum values of the softpick model are in fact from the last layer, all previous layers have even smaller activations.

6.4 Comparison with Other Sink-Free Methods

We compare softpick to several other sink-mitigation/sink-free attention variants using the same outlier diagnostics: sink rate (lower is better), activation statistics (kurtosis and extrema), and the fraction of exact zeros in the causal attention matrix ("Spars.%"). We omit downstream benchmarks because these ablation runs are too short to yield meaningful capabilities (see §4). Table 3 reports results for $\epsilon_s \in \{0.2, 0.3\}$.

Across both thresholds, softpick (and scaled softpick) most strongly suppresses sinks, achieving near-zero sink rates while substantially reducing activation extremes relative to softmax. Other methods help but less consistently: gated attention reaches single-digit sink rates but remains dense; softmax+1 roughly halves sink rate with near-zero sparsity; rectified is more sink-prone and shows the heaviest-tailed activations despite moderate sparsity. Softmax-like methods are effectively non-sparse (aside from underflow), whereas softpick yields genuinely sparse attention with many exact zeros, consistent with its mechanism.

7 Scalability

7.1 Long Context and Underscoring

Unlike softmax where every value must have a non-zero score, softpick can assign zero to unneeded values, theoretically leading to better retrieval for long context use. However, we have yet to see this benefit in our testing. On the passkey retrieval task (Lu et al., 2024), softpick performs

comparably to, but does not outperform, softmax across multiple sequence lengths (Table 5 in Appendix C). We hypothesize that this stems from an underscoring effect: as context length grows and attention becomes increasingly sparse, the normalization in softpick can significantly reduce the magnitude of scores assigned to the few relevant tokens, especially when many irrelevant tokens receive negative scores. This results in weaker value signals, as evidenced by the reduced scale in retrieval-specific heads that only attend to singular tokens (Appendix H, see the color bar value of the heatmaps). We also hypothesize that this is one of the reasons why the larger 1.8B model with softpick does not perform as well as softmax. We explored scaling the query-key scores prior to softpick, inspired by Scalable-Softmax (Nakanishi, 2025) and its use in Llama 4 (Meta, 2023), but found no improvement in retrieval and observed worse training and downstream performance (Appendix C).

7.2 Dead Attention Heads

We investigate why softpick may scale poorly to larger models. One hypothesis is that some attention heads remain dormant for long periods during training, which may hinder gradient flow or reduce effective capacity, effectively creating dead heads. To test this, we run in-distribution inference on 5M tokens using checkpoints from 0 to 100k steps (every 10k). We label a head as dead if its maximum absolute output is close to zero ($\epsilon = 10^{-6}$) for a token and this holds for at least 95% of tokens. The 340M model decreases from 17.19% to 9.11% dead heads over training, while the 1.8B model decreases from 42.87% to 20.90%. This suggests larger models exhibit more dormant heads, in this case more than double the proportion of dead attention heads in the 1.8B model compared to the 340M model, which may reduce capacity and/or gradient flow. Since our training is limited to 100B tokens, the continued decline in dead head percentage also suggests we may be undertraining, and scaling to trillion-token regimes may improve softpick's scalability. Full trends across checkpoints are shown in Appendix D (Figure 6).

8 Discussion and Implications

Across quantization, low-precision training, and efficiency considerations, a central challenge in modern transformers is the presence of massive activation outliers and attention sinks. Prior work has shown that these outliers distort scaling in activation quantization (Dettmers et al., 2022, 2023), degrade low-bit training stability (Fishman et al., 2024), and motivate increasingly complex mitigation strategies such as double quantization, activation smoothing, Hadamard transforms, or fine-grained quantization schemes (Xiao et al., 2023; Wang et al., 2025; DeepSeek-AI et al., 2024). By eliminating attention sinks and suppressing extreme activations, softpick directly addresses the root cause of these issues. Hence, models using softpick retain accuracy under aggressive quantization and benefit at lower bit widths, suggesting that simpler quantization and low-precision training pipelines may become viable without specialized outlier-handling mechanisms.

The sparsity induced in attention maps could enable potential inference acceleration via sparse kernels and reduce unnecessary computation in attention-value products. This sparsity also clarifies head behavior: heads that remain inactive until triggered are easier to identify and safely prune, potentially extending prior work on active-dormant heads (Guo et al., 2024) and head pruning (Wang et al., 2021; Shim et al., 2021). Finally, the resulting attention maps are more interpretable and visually legible, which may benefit analysis techniques such as attention rollout (Abnar and Zuidema, 2020). Since attention sinks and activation outliers are not unique to language models, softpick can also generalize to vision, video, and multimodal transformers, where similar artifacts have been observed (Darcet et al., 2024; Wen et al., 2025; Kumar et al., 2024).

9 Conclusion

We introduced Softpick, a rectified, non-sum-to-one drop-in replacement for softmax in transformer attention. Across 340M and 1.8B models trained from scratch, softpick eliminates attention sinks (0% sink rate), induces genuinely sparse attention maps, and strongly suppresses massive activation outliers, leading to improved robustness under low-bit post-training quantization, especially at very low precisions. While results at 340M are competitive with softmax, scaling and long-context behavior remain open challenges, motivating future work on softmax alternatives with desirable properties for transformer language models.

Limitations

First, softpick does not yet scale reliably under our current training recipe: while it is competitive with softmax at 340M, it underperforms at 1.8B when we reuse the same setup. We hypothesize that either a) underscoring of long contexts becomes severe at larger scales, resulting in poor context retrieval or b) during training, some attention heads may become persistently inactive (dead), reducing the model's effective capacity and contributing to the performance gap at larger scale. Additionally, as we explain in §7.1, softpick models do not yet yield improved long-context retrieval in our experiments (e.g., passkey retrieval). Despite producing sharper and sparser attention maps, the results suggest that additional work is needed to maintain strong value signals as context length increases. Lastly, given a fixed compute budget, we prioritized experimental breadth with training the full 340M/1.8B models, running additional short-run ablations across multiple sink-mitigation variants and softpick-related modifications, and acquiring metrics for analysis, over exhaustive large-scale hyperparameter sweeps and substantially longer training runs; we therefore leave a more comprehensive scaling study (larger parameter counts, longer contexts, and longer training horizons) to future work.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Hicham Badri and Appu Shaji. 2023. Hqq: Half-quadratic quantization for large language models (blog post). https://mobiusml.github.io/hqq_blog/. Accessed: 2025-12-30.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Federico Barbero, Álvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. 2025. Why do llms attend to the first token? *Preprint*, arXiv:2504.02732.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2023. Quantizable transformers: Removing outliers by helping attention heads do nothing. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

David Chiang. 2024. Transformers in uniform tc^0. *Preprint*, arXiv:2409.13629.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. 2024. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2024. Deepseek-v3 technical report.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Maxim Fishman, Brian Chmiel, Ron Banner, and Daniel Soudry. 2024. Scaling fp8 training to trillion-token llms. *ArXiv preprint*, abs/2409.12517.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. 2024. When attention sink emerges in language models: An empirical view. *Preprint*, arXiv:2410.10781.

Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael I. Jordan, and Song Mei. 2024. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. *Preprint*, arXiv:2410.13835.

Prannay Kaul, Chengcheng Ma, Ismail Elezi, and Jiankang Deng. 2024. From attention to activation: Unravelling the enigmas of large language models. *Preprint*, arXiv:2410.17174.

Shashi Kumar, Srikanth R. Madikeri, Juan Pablo Zuluaga, Esaú Villatoro-Tello, Iuliia Nigmatulina, Petr Motlícek, E ManjunathK, and Aravind Ganapathiraju. 2024. Xlsr-transducer: Streaming asr for self-supervised pretrained models. *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Wanchao Liang, Tianyu Liu, Less Wright, Will Constable, Andrew Gu, Chien-Chin Huang, Iris Zhang, Wei Feng, Howard Huang, Junjie Wang, Sanket Purandare, Gokul Nadathur, and Stratos Idreos. 2025. Torchtitan: One-stop pytorch native solution for production ready LLM pretraining. In *The Thirteenth International Conference on Learning Representations*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Yi Lu, Xin Zhou, Wei He, Jun Zhao, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longheads: Multi-head attention is secretly a long context processor. *Preprint*, arXiv:2402.10685.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Meta. 2023. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation.

Evan Miller. 2023. Attention is off by one.

Ken M. Nakanishi. 2025. Scalable-softmax is superior for attention. *Preprint*, arXiv:2501.19399.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Louis Owen, Nilabhra Roy Chowdhury, Abhay Kumar, and Fabian Güra. 2025. A refined analysis of massive activations in llms. *Preprint*, arXiv:2503.22329.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hemanth Saratchandran, Jianqiao Zheng, Yiping Ji, Wenbo Zhang, and Simon Lucey. 2024. Rethinking softmax: Self-attention with polynomial activations. *Preprint*, arXiv:2410.18613.

Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. 2021. Layer-wise pruning of transformer attention heads for efficient language modeling. In *2021 18th International SoC Design Conference (ISOCC)*, page 357–358. IEEE.

Ethan Smith. 2025. Softmax attention is a fluke.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. 2024. Massive activations in large language models. *Preprint*, arXiv:2402.17762.

Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE.

Hongyu Wang, Shuming Ma, and Furu Wei. 2025. Bitnet v2: Native 4-bit activations with hadamard transformation for 1-bit llms.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Yuxin Wen, Jim Wu, Ajay Jain, Tom Goldstein, and Ashwinee Panda. 2025. Analysis of attention in video diffusion transformers. *Preprint*, arXiv:2504.10317.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Songlin Yang and Yu Zhang. 2024. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism.

A FlashAttention Algorithms

Algorithm 1 FlashAttention-2 Forward Pass with Softpick

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, block sizes B_c, B_r, denominator epsilon ϵ

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.
Divide the output $\mathbf{O} \in \mathbb{R}^{N \times d}$ into T_r blocks $\mathbf{O}_i, \ldots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, and divide the logsumexp L into T_r blocks $L_i, \ldots, L_{T_r}$ of size B_r each.
for $1 \leq i \leq T_r$ **do**
  Load $\mathbf{Q}_i$ from HBM to on-chip SRAM.
  On chip, initialize $\mathbf{O}_i^{(0)} = (0)_{B_r \times d} \in \mathbb{R}^{B_r \times d}$, $\ell_i^{(0)} = (0)_{B_r} \in \mathbb{R}^{B_r}$, $m_i^{(0)} = (-\infty)_{B_r} \in \mathbb{R}^{B_r}$.
  for $1 \leq j \leq T_c$ **do**
    Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.
    On chip, compute $\mathbf{S}_i^{(j)} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.
    On chip, compute $m_i^{(j)} = \max(m_i^{(j-1)}, \mathrm{rowmax}(\mathbf{S}_i^{(j)})) \in \mathbb{R}^{B_r}$,
    $\tilde{\mathbf{P}}_i^{(j)} = \exp(\mathbf{S}_i^{(j)} - m_i^{(j)}) - \exp(-m_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise),
    $\tilde{\mathbf{R}}_i^{(j)} = ReLU(\tilde{\mathbf{P}}_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise),
    $\tilde{\mathbf{A}}_i^{(j)} = |\tilde{\mathbf{P}}_i^{(j)}| \in \mathbb{R}^{B_r \times B_c}$ (pointwise),
    $\ell_i^{(j)} = e^{m_i^{j-1} - m_i^{(j)}} \ell_i^{(j-1)} + \mathrm{rowsum}(\tilde{\mathbf{A}}_i^{(j)}) \in \mathbb{R}^{B_r}$.
    On chip, compute $\mathbf{O}_i^{(j)} = \mathrm{diag}(e^{m_i^{(j-1)} - m_i^{(j)}})^{-1} \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{R}}_i^{(j)} \mathbf{V}_j$.
  end for
  On chip, compute $\ell_i^{(T_c)} = \ell_i^{(T_c)} + \epsilon$.
  On chip, compute $\mathbf{O}_i = \mathrm{diag}(\ell_i^{(T_c)})^{-1} \mathbf{O}_i^{(T_c)}$.
  On chip, compute $L_i = m_i^{(T_c)} + \log(\ell_i^{(T_c)})$.
  Write $\mathbf{O}_i$ to HBM as the i-th block of $\mathbf{O}$.
  Write L_i to HBM as the i-th block of L.
end for
Return the output $\mathbf{O}$ and the logsumexp L.

Algorithm 2 FlashAttention-2 Backward Pass with Softpick

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{dO} \in \mathbb{R}^{N \times d}$ in HBM, vector $L \in \mathbb{R}^N$ in HBM, block sizes B_c, B_r.

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \ldots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \ldots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

Divide $\mathbf{O}$ into T_r blocks $\mathbf{O}_i, \ldots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide $\mathbf{dO}$ into T_r blocks $\mathbf{dO}_i, \ldots, \mathbf{dO}_{T_r}$ of size $B_r \times d$ each, and divide L into T_r blocks $L_i, \ldots, L_{T_r}$ of size B_r each.

Initialize $\mathbf{dQ} = (0)_{N \times d}$ in HBM and divide it into T_r blocks $\mathbf{dQ}_1, \ldots, \mathbf{dQ}_{T_r}$ of size $B_r \times d$ each. Divide $\mathbf{dK}, \mathbf{dV} \in \mathbb{R}^{N \times d}$ in to T_c blocks $\mathbf{dK}_1, \ldots, \mathbf{dK}_{T_c}$ and $\mathbf{dV}_1, \ldots, \mathbf{dV}_{T_c}$, of size $B_c \times d$ each.

Compute $D = \mathrm{rowsum}(\mathbf{dO} \circ \mathbf{O}) \in \mathbb{R}^d$ (pointwise multiply), write D to HBM and divide it into T_r blocks $D_1, \ldots, D_{T_r}$ of size B_r each.

for $1 \le j \le T_c$ **do**

  Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

  Initialize $\mathbf{dK}_j = (0)_{B_c \times d}, \mathbf{dV}_j = (0)_{B_c \times d}$ on SRAM.

  for $1 \le i \le T_r$ **do**

    Load $\mathbf{Q}_i, \mathbf{O}_i, \mathbf{dO}_i, \mathbf{dQ}_i, L_i, D_i$ from HBM to on-chip SRAM.

    On chip, compute $\mathbf{S}_i^{(j)} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.

    On chip, compute $\mathbf{E}_i^{(j)} = \exp(\mathbf{S}_{ij} - L_i) \in \mathbb{R}^{B_r \times B_c}$,
    $\mathbf{P}_i^{(j)} = \mathbf{E}_i^{(j)} - \exp(-L_i) \in \mathbb{R}^{B_r \times B_c}$,
    $\mathbf{R}_i^{(j)} = ReLU(\mathbf{P}_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$.

    On chip, compute $\mathbf{dV}_j \leftarrow \mathbf{dV}_j + (\mathbf{R}_i^{(j)})^\top \mathbf{dO}_i \in \mathbb{R}^{B_c \times d}$.

    On chip, compute $\mathbf{dP}_i^{(j)} = \mathbf{dO}_i \mathbf{V}_j^\top \in \mathbb{R}^{B_r \times B_c}$,
    $\mathbf{dR}_i^{(j)} = step(\mathbf{S}_i^{(j)}) \circ \mathbf{dP}_i^{(j)} \in \mathbb{R}^{B_r \times B_c}$ (pointwise multiply or use where() function),
    $\mathbf{dA}_i^{(j)} = sign(\mathbf{S}_i^{(j)}) \circ D_i \in \mathbb{R}^{B_r \times B_c}$ (pointwise multiply or use where() function).

    On chip, compute $\mathbf{dS}_i^{(j)} = \mathbf{E}_i^{(j)} \circ (\mathbf{dR}_i^{(j)} - \mathbf{dA}_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$.

    Load $\mathbf{dQ}_i$ from HBM to SRAM, then on chip, update $\mathbf{dQ}_i \leftarrow \mathbf{dQ}_i + \mathbf{dS}_i^{(j)} \mathbf{K}_j \in \mathbb{R}^{B_r \times d}$, and write back to HBM.

    On chip, compute $\mathbf{dK}_j \leftarrow \mathbf{dK}_j + \mathbf{dS}_i^{(j)\top} \mathbf{Q}_i \in \mathbb{R}^{B_c \times d}$.

  end for

  Write $\mathbf{dK}_j, \mathbf{dV}_j$ to HBM.

end for

Return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$.

B Training Configuration

Param.	Value.	Param.	Value.
Architecture			
Hidden size	{1024, 2048}	# Heads	{16, 32}
# Layers	{24, 32}	KV heads	{16, 32}
Seq. length	4096	RoPE θ	10k
Vocab size	32k	Tied emb.	False
Optimization & training			
Optimizer	AdamW	LR schedule	cosine
LR	{3e-4, 2e-4}	Min LR (%)	10%
Warmup	{1k, 2k}	Train steps	{100k, 200k}
Dev. batch	{16, 8}	Analysis steps	10k
Grad. accum.	{1, 2}	Global batch	128
Softpick ϵ	1e-6	Max grad norm	1.0

Table 4: Training configuration and hyperparameters for {340M, 1.8B} models.

C Other Experiments

We report negative results from two additional variants related to Softpick. First, we replace the standard softmax with a *rectified-only softmax*, which masks negative logits before normalization:

$$\nu(x) = \begin{cases} x & x \geq 0, \\ -\infty & x < 0, \end{cases}$$

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{\nu(x_i)}}{\sum_{j=1}^{N} e^{\nu(x_j)}}.$$

Second, we evaluate a *scalable-softpick* variant inspired by Nakanishi (2025), which introduces a sequence-length-dependent scaling factor:

$$\alpha = \frac{s \log n}{\sqrt{d_k}}, \tag{8}$$

where s is a per-head trainable parameter and n is the sequence length. In our implementation, n is a vector that assigns a (possibly different) effective sequence length to each token position. The resulting attention becomes

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softpick}(\alpha\, \mathbf{Q}\mathbf{K}^T)\, \mathbf{V}. \tag{9}$$

We train both variants using the same setup and configuration as our main 340M-parameter models. We report training loss and gradient-norm curves in Figure 7, benchmark results in Table 6, and passkey retrieval results in Table 5.

D Dead Head Analysis

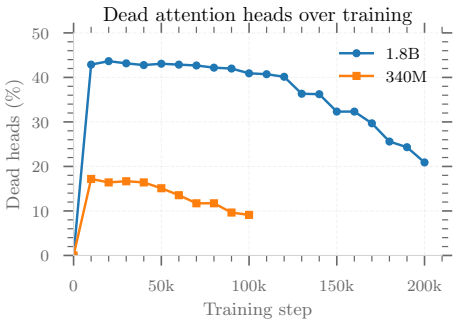


Figure 6: Percentage of Dead Head on Softpick on 340M and 1.8B Parameters Across Training Steps

From Figure 6, we can see that at the start of training, the weights are randomly initialized; therefore, dead heads have not yet emerged. However, after only the first 10k steps, we observe that the number of dead heads reaches its peak. Subsequently, this count consistently decreases until the end of training.

E Passkey Retrieval Results

Model	Sequence Length				
	890	1983	3075	4167	4986
Softmax	95.5	97.0	73.0	70.5	0.0
Softpick	94.0	91.0	75.0	66.5	0.0
Rectified-Only	95.5	68.0	57.5	24.5	0.0
Scalable-Softmax	98.0	96.9	87.0	55.0	0.0

Table 5: Passkey retrieval results including the other experiments.

Table 5 reports passkey retrieval accuracy across increasing sequence lengths. Performance generally degrades as context grows, reflecting the increased difficulty of retrieving the key from longer inputs. All methods collapse to 0.0 at length 4986 because our models were trained with a maximum context length of 4096, so this setting is out-of-distribution and exceeds the trained positional range.

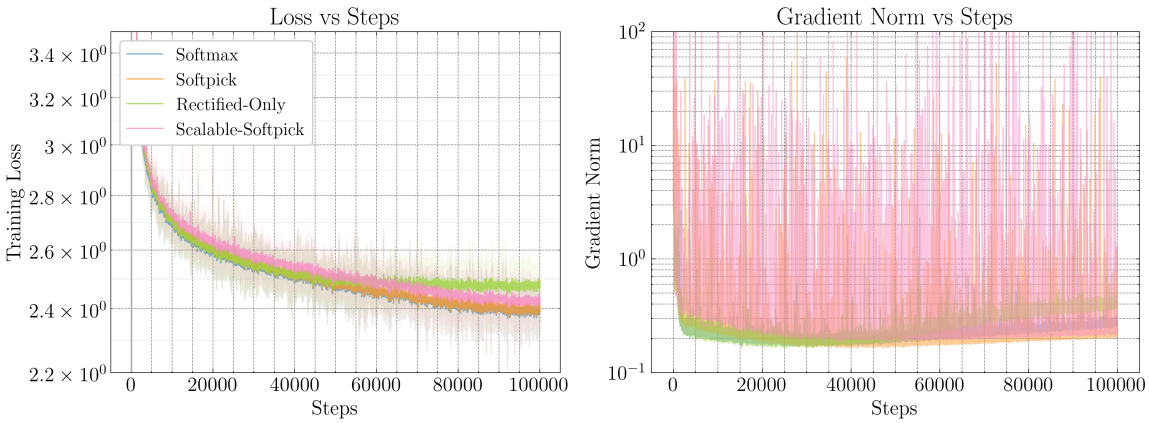


Figure 7: Training loss and gradient norm during training, including other experiments.

Task	Metric		Softmax	Softpick	Rectified-Only	Scalable-Softpick
Arc Easy	Acc Norm	↑	56.31	56.61	51.52	55.89
	Acc	↑	60.61	60.82	57.66	60.98
Lambada	Acc	↑	36.35	36.21	29.05	33.17
	Perplexity	↓	30.33	28.67	53.77	34.49
Piqa	Acc Norm	↑	66.43	66.27	66.27	66.32
	Acc	↑	66.87	66.43	66.43	66.92
Sciq	Acc Norm	↑	75.10	77.20	68.80	75.00
	Acc	↑	83.30	83.60	78.80	83.70
Wikitext	Word Perplexity	↓	24.01	24.41	27.93	25.13

Table 6: Comparison of softmax, softpick, rectified-only softmax, and scalable-softpick performance on downstream tasks. ↑ = Higher is better, ↓ = Lower is better.

F Full Results on Quantization of the 340M Models

Table 7: Comparison of softpick vs softmax performance for BNB (Dettmers et al., 2022, 2023) quantization methods. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

Task	Metric		Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm	↑	4-bit	53.66	53.87	+0.21
	Acc	↑	4-bit	58.88	59.60	+0.72
	Acc Norm	↑	8-bit	55.51	56.36	+0.84
	Acc	↑	8-bit	59.81	60.82	+1.01
Lambada	Acc	↑	4-bit	33.63	31.73	-1.90
	Perplexity	↓	4-bit	37.59	39.25	+1.67
	Acc	↑	8-bit	35.53	35.86	+0.33
	Perplexity	↓	8-bit	31.22	28.74	-2.48
Piqa	Acc Norm	↑	4-bit	65.29	66.38	+1.09
	Acc	↑	4-bit	65.40	66.32	+0.92
	Acc Norm	↑	8-bit	66.32	66.32	+0.00
	Acc	↑	8-bit	67.36	66.38	-0.98
Sciq	Acc Norm	↑	4-bit	74.90	75.10	+0.20
	Acc	↑	4-bit	82.90	82.80	-0.10
	Acc Norm	↑	8-bit	75.20	77.40	+2.20
	Acc	↑	8-bit	83.40	83.50	+0.10
Wikitext	Word Perplexity	↓	4-bit	26.38	26.47	+0.09
			8-bit	23.95	24.40	+0.46

Table 8: Comparison of softpick vs softmax performance for GPTQ (Frantar et al., 2022) quantization. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

Task	Metric	Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm ↑	2-bit	27.27	29.76	+2.48
	Acc ↑	2-bit	27.02	29.88	+2.86
	Acc Norm ↑	3-bit	48.57	51.81	+3.24
	Acc ↑	3-bit	53.83	56.65	+2.82
	Acc Norm ↑	4-bit	53.96	54.29	+0.34
	Acc ↑	4-bit	59.93	59.26	-0.67
	Acc Norm ↑	8-bit	56.31	56.36	+0.04
	Acc ↑	8-bit	60.31	60.94	+0.63
Lambada	Acc ↑	2-bit	0.04	0.43	+0.39
	Perplexity ↓	2-bit	930980.10	133928.18	-797051.92
	Acc ↑	3-bit	23.71	29.07	+5.36
	Perplexity ↓	3-bit	82.12	52.32	-29.80
	Acc ↑	4-bit	33.55	34.97	+1.42
	Perplexity ↓	4-bit	37.30	31.08	-6.21
	Acc ↑	8-bit	36.44	36.15	-0.29
	Perplexity ↓	8-bit	30.06	28.66	-1.41
Piqa	Acc Norm ↑	2-bit	48.42	53.59	+5.17
	Acc ↑	2-bit	52.12	54.52	+2.39
	Acc Norm ↑	3-bit	63.76	64.64	+0.87
	Acc ↑	3-bit	64.53	65.56	+1.03
	Acc Norm ↑	4-bit	66.10	66.43	+0.33
	Acc ↑	4-bit	65.89	66.59	+0.71
	Acc Norm ↑	8-bit	66.59	66.54	-0.05
	Acc ↑	8-bit	66.81	66.21	-0.60
Sciq	Acc Norm ↑	2-bit	29.80	35.80	+6.00
	Acc ↑	2-bit	27.80	34.80	+7.00
	Acc Norm ↑	3-bit	73.50	75.90	+2.40
	Acc ↑	3-bit	79.30	82.30	+3.00
	Acc Norm ↑	4-bit	73.40	76.40	+3.00
	Acc ↑	4-bit	81.00	82.70	+1.70
	Acc Norm ↑	8-bit	75.00	77.60	+2.60
	Acc ↑	8-bit	83.20	83.70	+0.50
Wikitext	Word Perplexity ↓	2-bit	4081.76	893.35	-3188.41
		3-bit	34.57	31.53	-3.04
		4-bit	25.69	25.53	-0.16
		8-bit	23.84	24.31	+0.47

Table 9: Comparison of softpick vs softmax performance for HQQ (Badri and Shaji, 2023) quantization methods. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

Task	Metric		Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm	↑	2-bit	26.68	36.36	+9.68
	Acc	↑	2-bit	26.81	37.54	+10.73
	Acc Norm	↑	3-bit	48.65	52.57	+3.91
	Acc	↑	3-bit	52.78	57.95	+5.18
Lambada	Acc	↑	2-bit	0.00	5.76	+5.76
	Perplexity	↓	2-bit	21857842.67	3741.81	-21854100.86
	Acc	↑	3-bit	26.18	32.14	+5.96
	Perplexity	↓	3-bit	76.81	41.85	-34.96
Piqa	Acc Norm	↑	2-bit	50.38	55.11	+4.73
	Acc	↑	2-bit	52.29	56.04	+3.75
	Acc Norm	↑	3-bit	62.40	65.78	+3.37
	Acc	↑	3-bit	63.55	66.00	+2.45
Sciq	Acc Norm	↑	2-bit	22.20	51.40	+29.20
	Acc	↑	2-bit	21.60	53.60	+32.00
	Acc Norm	↑	3-bit	71.60	72.10	+0.50
	Acc	↑	3-bit	80.50	80.70	+0.20
Wikitext	Word Perplexity	↓	2-bit	127870.20	2139.86	-125730.34
			3-bit	35.52	30.24	-5.29

G Full Results on Quantization of the 1.8B Models

Table 10: Comparison of softpick vs softmax performance for BNB (Dettmers et al., 2022, 2023) quantization methods. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

Task	Metric		Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm	↑	4-bit	66.75	58.84	-7.91
	Acc	↑	4-bit	71.93	67.05	-4.88
	Acc Norm	↑	8-bit	67.26	62.21	-5.05
	Acc	↑	8-bit	72.22	69.07	-3.16
Lambada	Acc	↑	4-bit	47.84	41.26	-6.58
	Perplexity	↓	4-bit	12.90	18.56	+5.66
	Acc	↑	8-bit	48.65	43.14	-5.51
	Perplexity	↓	8-bit	11.74	16.11	+4.36
Piqa	Acc Norm	↑	4-bit	72.14	70.57	-1.58
	Acc	↑	4-bit	72.63	71.00	-1.63
	Acc Norm	↑	8-bit	72.96	70.40	-2.56
	Acc	↑	8-bit	73.56	71.22	-2.34
Sciq	Acc Norm	↑	4-bit	86.90	79.40	-7.50
	Acc	↑	4-bit	90.90	86.90	-4.00
	Acc Norm	↑	8-bit	86.00	79.90	-6.10
	Acc	↑	8-bit	90.30	86.90	-3.40
Wikitext	Word Perplexity	↓	4-bit	16.17	18.97	+2.80
			8-bit	15.20	17.94	+2.74

Table 11: Comparison of softpick vs softmax performance for GPTQ (Frantar et al., 2022) quantization. ↑=Higher is Better, ↓=Lower is Better. Δ = Softpick - Softmax.

Task	Metric	Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm ↑	2-bit	28.62	34.60	+5.98
	Acc ↑	2-bit	29.67	33.88	+4.21
	Acc Norm ↑	3-bit	61.28	57.49	-3.79
	Acc ↑	3-bit	64.90	64.27	-0.63
	Acc Norm ↑	4-bit	65.74	61.87	-3.87
	Acc ↑	4-bit	70.20	68.73	-1.47
	Acc Norm ↑	8-bit	66.96	61.95	-5.01
	Acc ↑	8-bit	72.77	68.60	-4.17
Lambada	Acc ↑	2-bit	1.14	1.59	+0.45
	Perplexity ↓	2-bit	27256.52	18730.23	-8526.29
	Acc ↑	3-bit	40.71	36.77	-3.94
	Perplexity ↓	3-bit	20.58	25.33	+4.75
	Acc ↑	4-bit	49.64	40.95	-8.69
	Perplexity ↓	4-bit	11.43	17.94	+6.50
	Acc ↑	8-bit	49.60	43.41	-6.19
	Perplexity ↓	8-bit	11.35	15.82	+4.47
Piqa	Acc Norm ↑	2-bit	53.97	53.92	-0.05
	Acc ↑	2-bit	54.46	55.22	+0.76
	Acc Norm ↑	3-bit	71.00	70.51	-0.49
	Acc ↑	3-bit	69.91	70.18	+0.27
	Acc Norm ↑	4-bit	73.34	69.97	-3.37
	Acc ↑	4-bit	71.98	71.11	-0.87
	Acc Norm ↑	8-bit	73.83	70.67	-3.16
	Acc ↑	8-bit	73.72	71.27	-2.45
Sciq	Acc Norm ↑	2-bit	45.80	45.30	-0.50
	Acc ↑	2-bit	43.90	43.80	-0.10
	Acc Norm ↑	3-bit	82.20	78.30	-3.90
	Acc ↑	3-bit	88.00	85.50	-2.50
	Acc Norm ↑	4-bit	85.90	80.40	-5.50
	Acc ↑	4-bit	90.50	86.60	-3.90
	Acc Norm ↑	8-bit	86.40	80.40	-6.00
	Acc ↑	8-bit	90.00	87.20	-2.80
Wikitext	Word Perplexity ↓	2-bit	4660.08	336.22	-4323.86
		3-bit	19.24	21.77	+2.52
		4-bit	15.84	18.46	+2.62
		8-bit	15.10	17.86	+2.77

Table 12: Comparison of softpick vs softmax performance for HQQ (Badri and Shaji, 2023) quantization methods. ↑=Higher is better, ↓=Lower is better. Δ = Softpick - Softmax.

Task	Metric		Quantization	Softmax	Softpick	Δ
Arc Easy	Acc Norm	↑	2-bit	67.21	61.99	-5.22
	Acc	↑	2-bit	72.64	68.60	-4.04
	Acc Norm	↑	3-bit	67.21	61.99	-5.22
	Acc	↑	3-bit	72.64	68.60	-4.04
Lambada	Acc	↑	2-bit	49.56	43.61	-5.96
	Perplexity	↓	2-bit	11.38	15.83	+4.45
	Acc	↑	3-bit	49.56	43.61	-5.96
	Perplexity	↓	3-bit	11.38	15.83	+4.45
Piqa	Acc Norm	↑	2-bit	73.45	70.89	-2.56
	Acc	↑	2-bit	73.78	71.38	-2.39
	Acc Norm	↑	3-bit	73.45	70.89	-2.56
	Acc	↑	3-bit	73.78	71.38	-2.39
Sciq	Acc Norm	↑	2-bit	86.30	80.40	-5.90
	Acc	↑	2-bit	90.20	87.10	-3.10
	Acc Norm	↑	3-bit	86.30	80.40	-5.90
	Acc	↑	3-bit	90.20	87.10	-3.10
Wikitext	Word Perplexity	↓	2-bit	15.10	17.86	+2.77
			3-bit	15.10	17.86	+2.77

H More Attention Maps

To show the difference between attention maps produced by softmax and softpick, we run 3 sample text inputs through the models and extract the resulting attention maps. The sample text inputs are the following:

Input 1: "The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism."

Input 2: "According to all known laws of aviation, there is no way that a bee should be able to fly. Its wings are too small to get its fat little body off the ground. The bee, of course, flies anyway. Because bees don't care what humans think is impossible."

Input 3: "An idol whose dream is to become the owner of a fast food chain. Kiara is a phoenix, not a chicken or turkey (Very important). She burns brightly, working herself to the bone since she'll just be reborn from her ashes anyway."

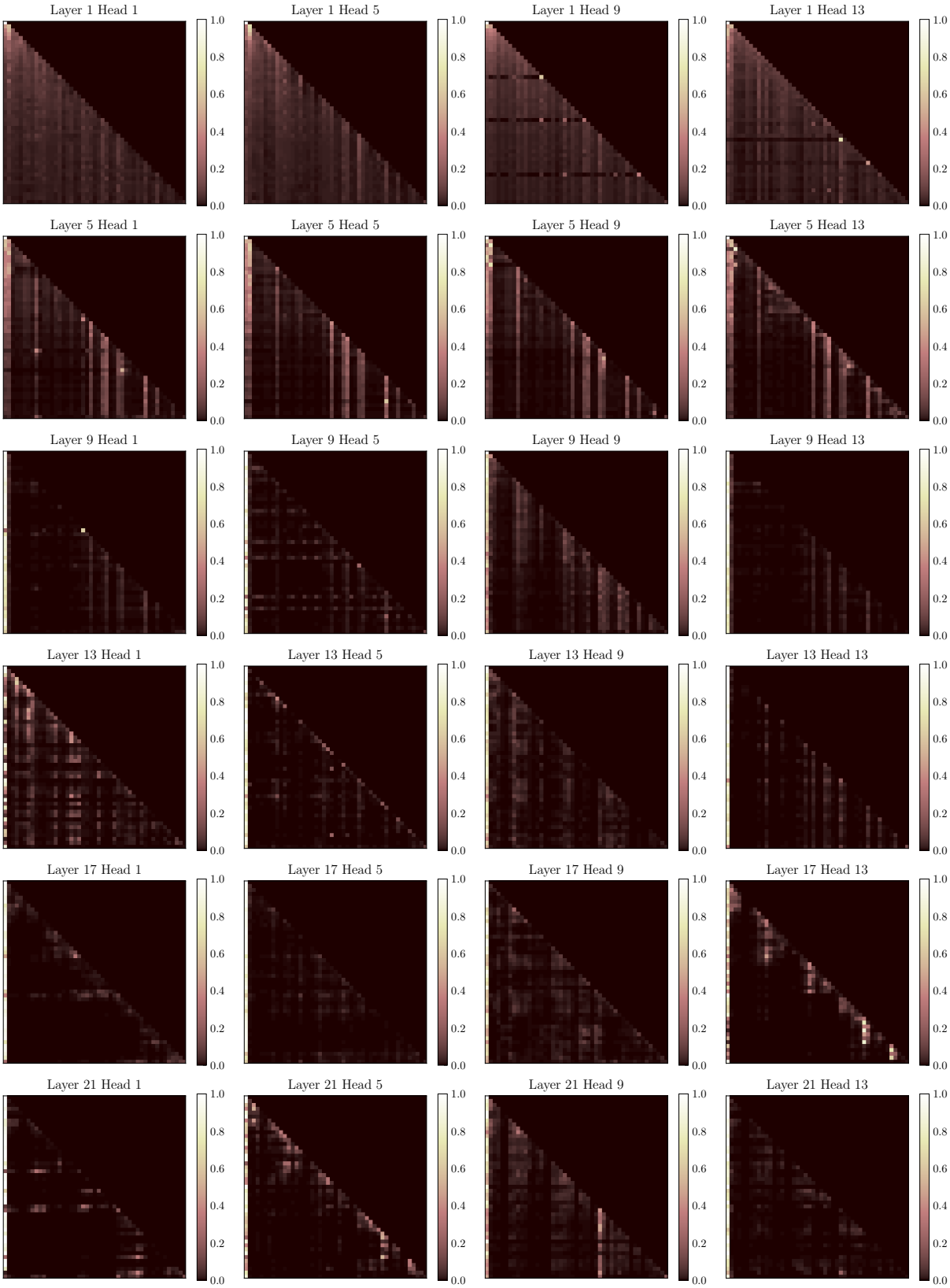


Figure 8: More attention maps of the softmax 340M model on sample text input 1.

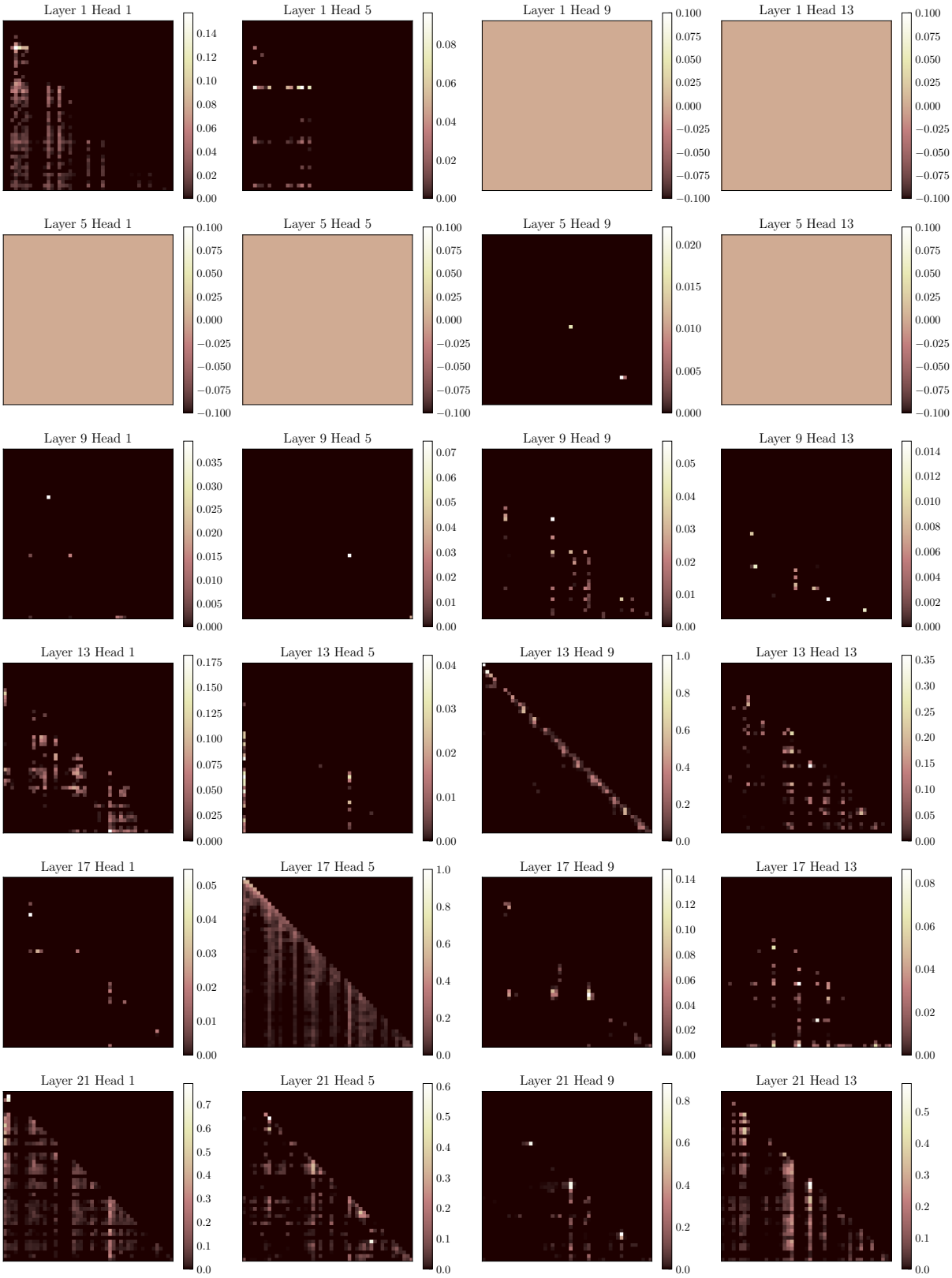


Figure 9: More attention maps of the softpick 340M model on sample text input 1.

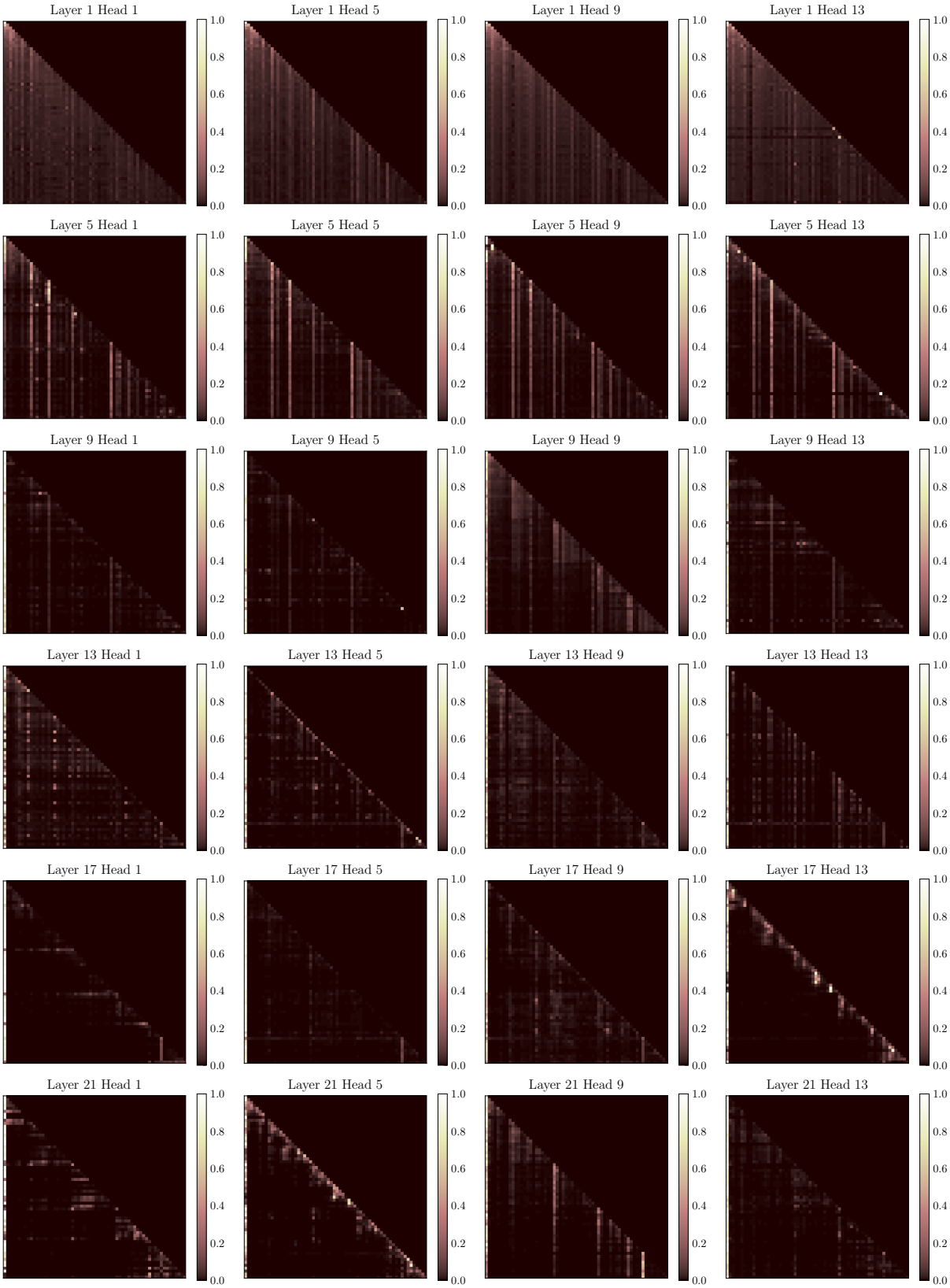


Figure 10: More attention maps of the softmax 340M model on sample text input 2.

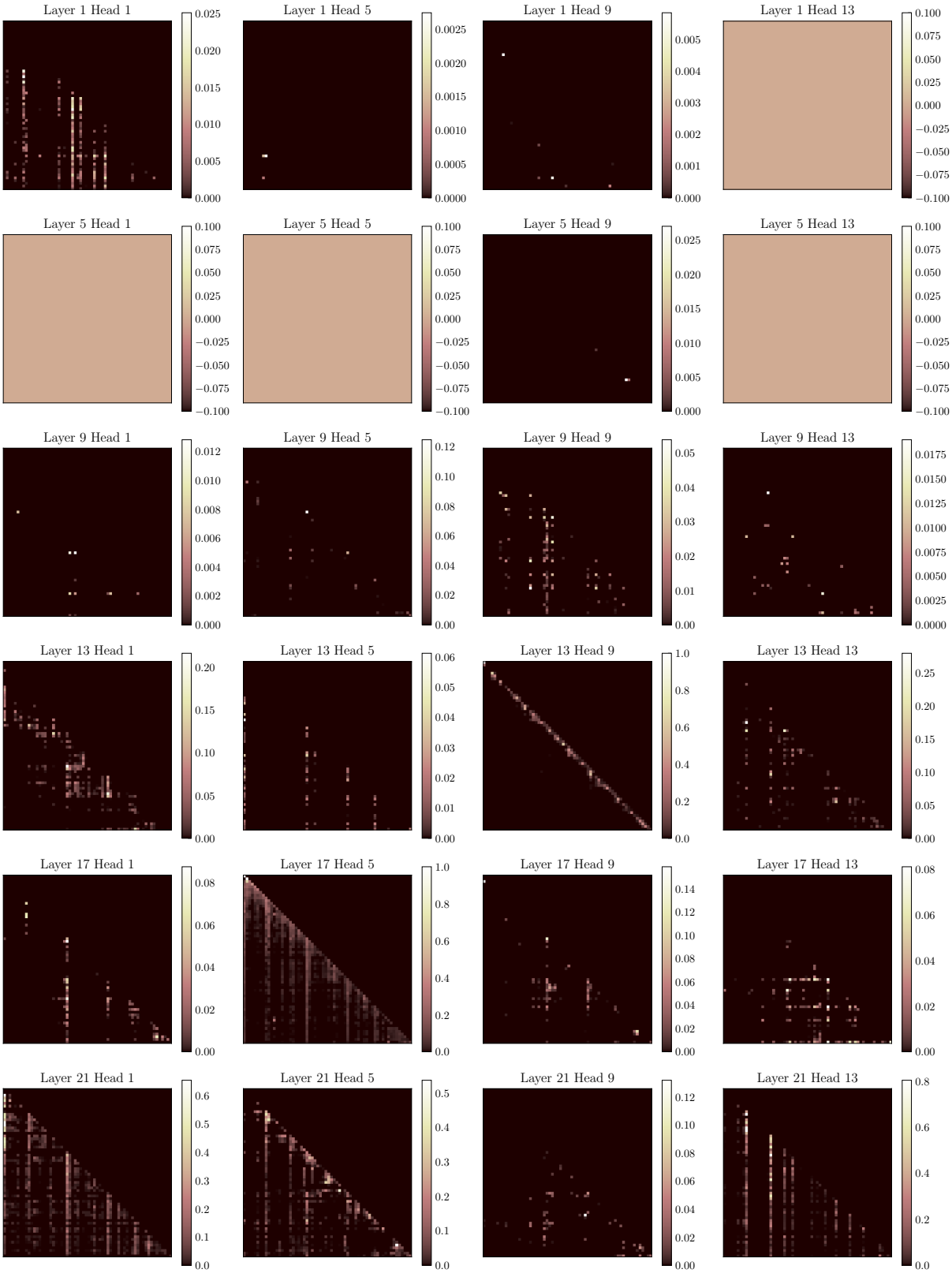


Figure 11: More attention maps of the softpick 340M model on sample text input 2.

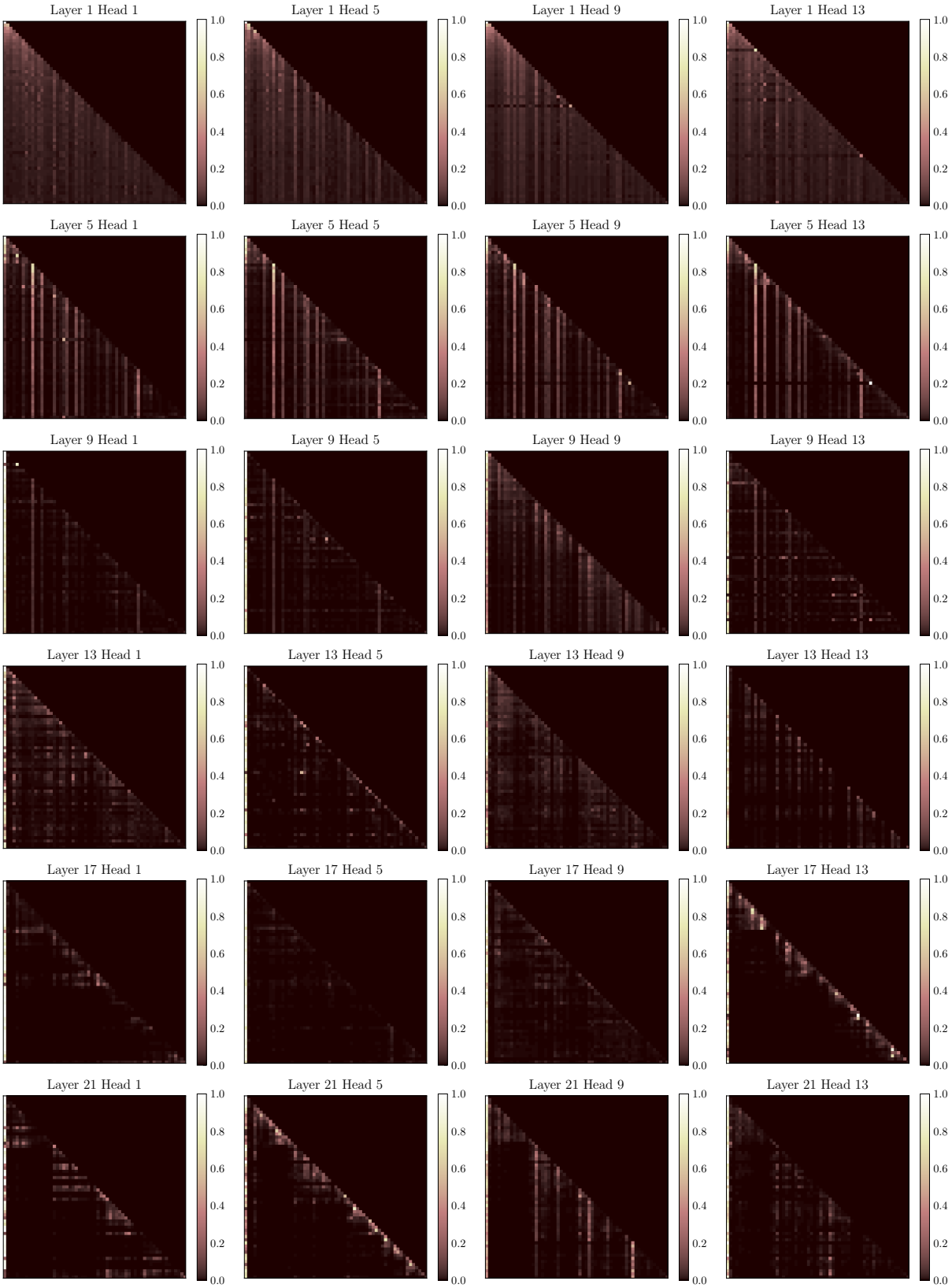


Figure 12: More attention maps of the softmax 340M model on sample text input 3.

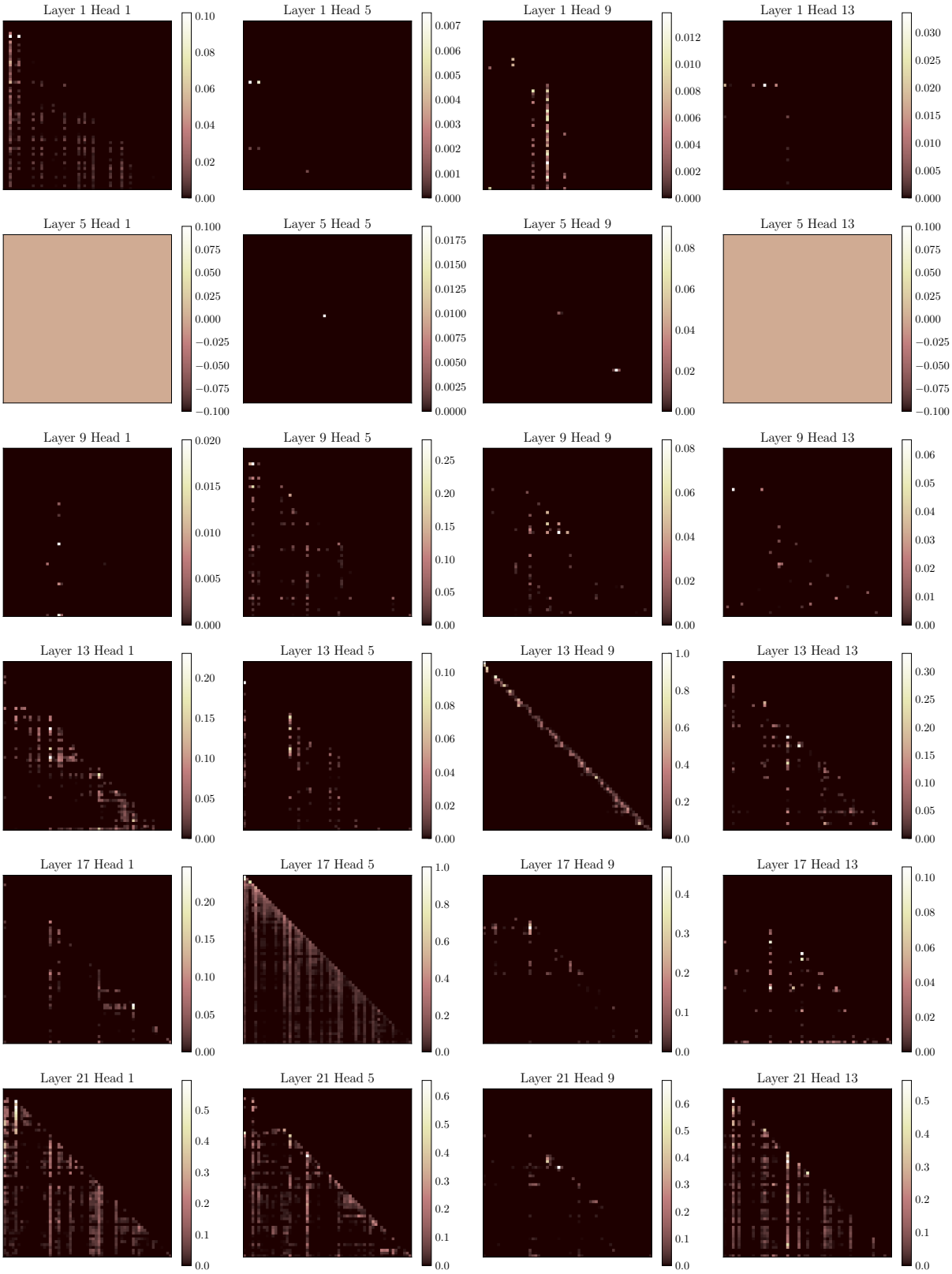


Figure 13: More attention maps of the softpick 340M model on sample text input 3.

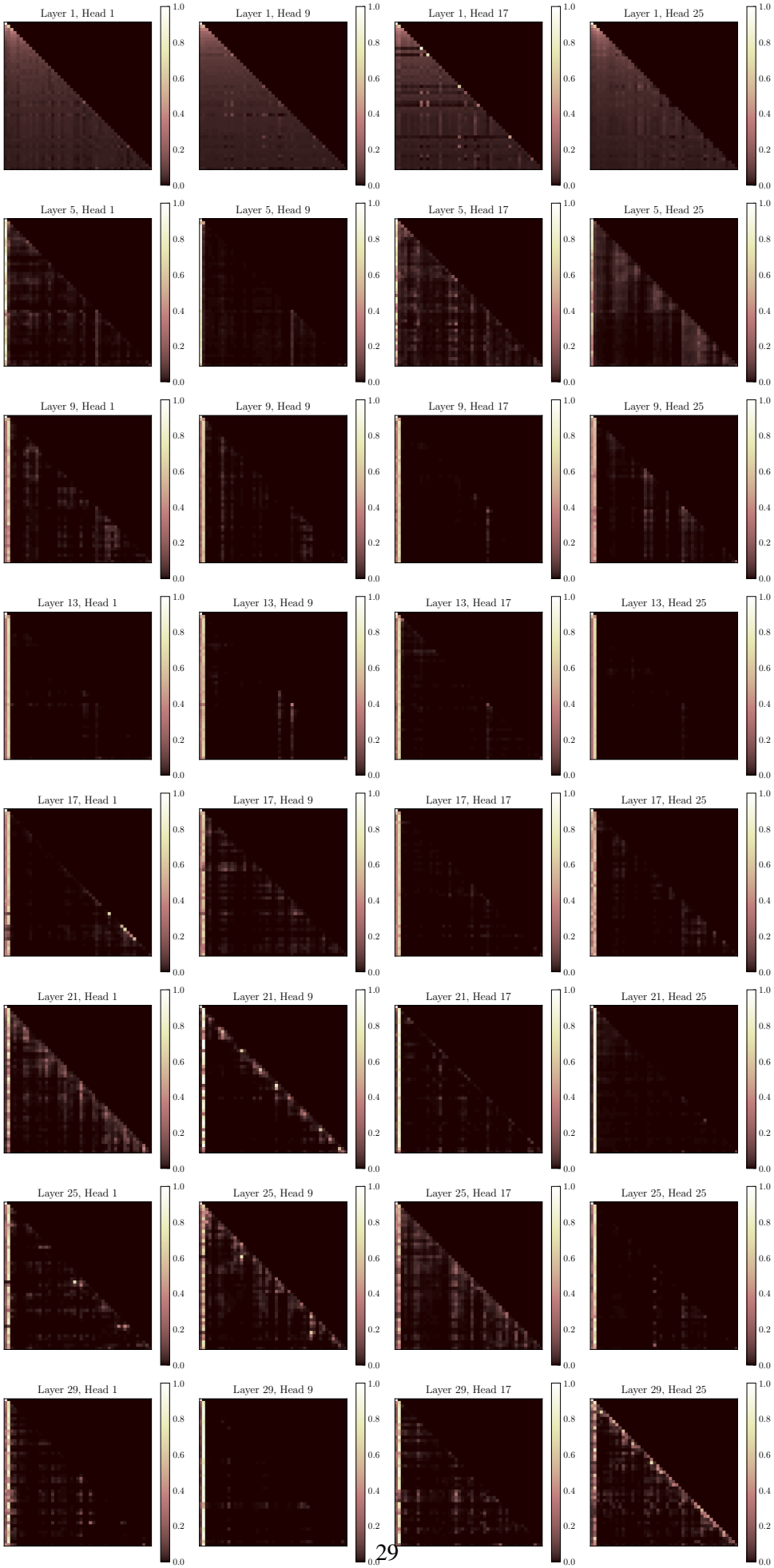


Figure 14: More attention maps of the softmax 1.8B model on sample text input 1.

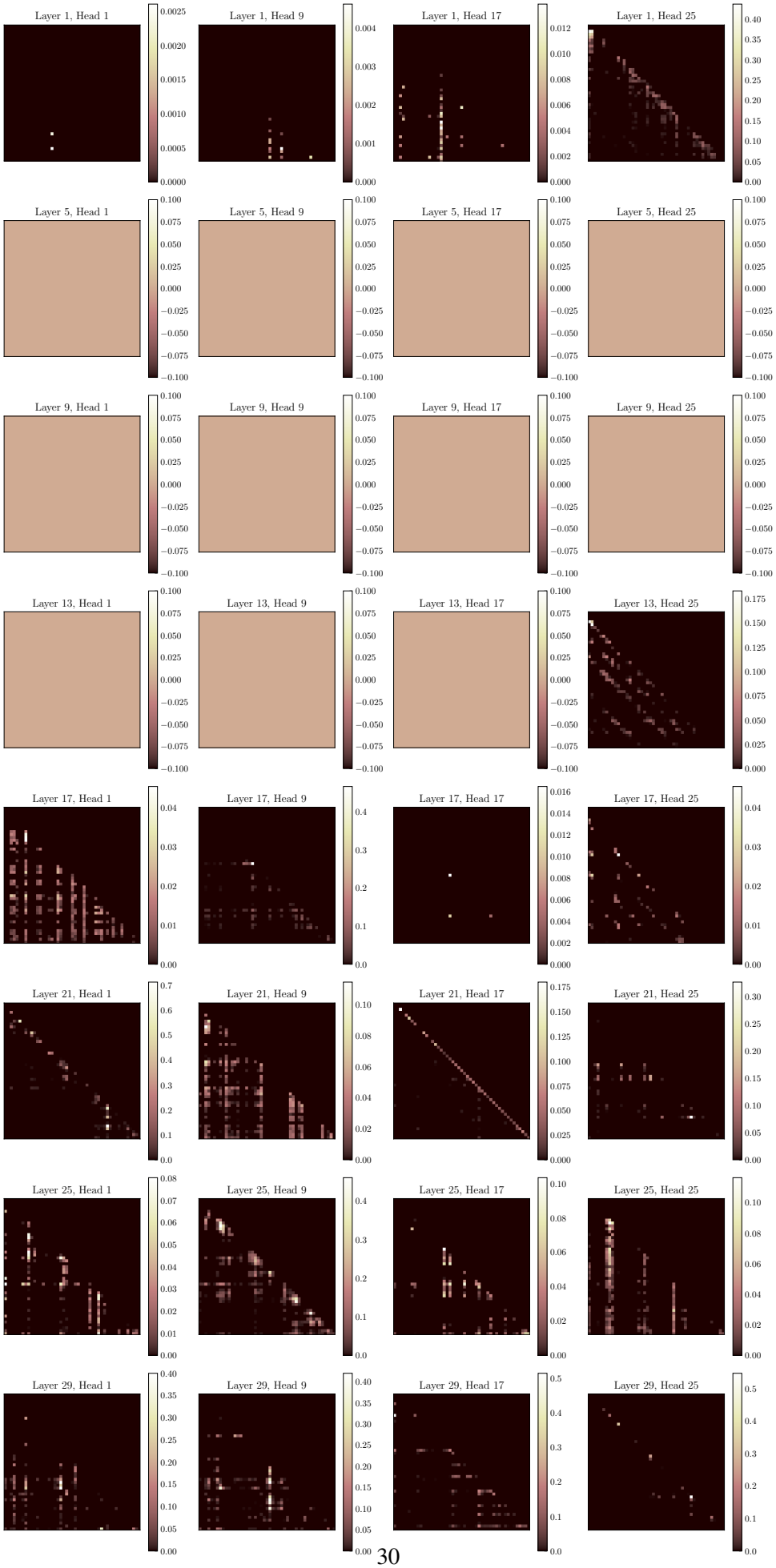


Figure 15: More attention maps of the softpick 1.8B model on sample text input 1.

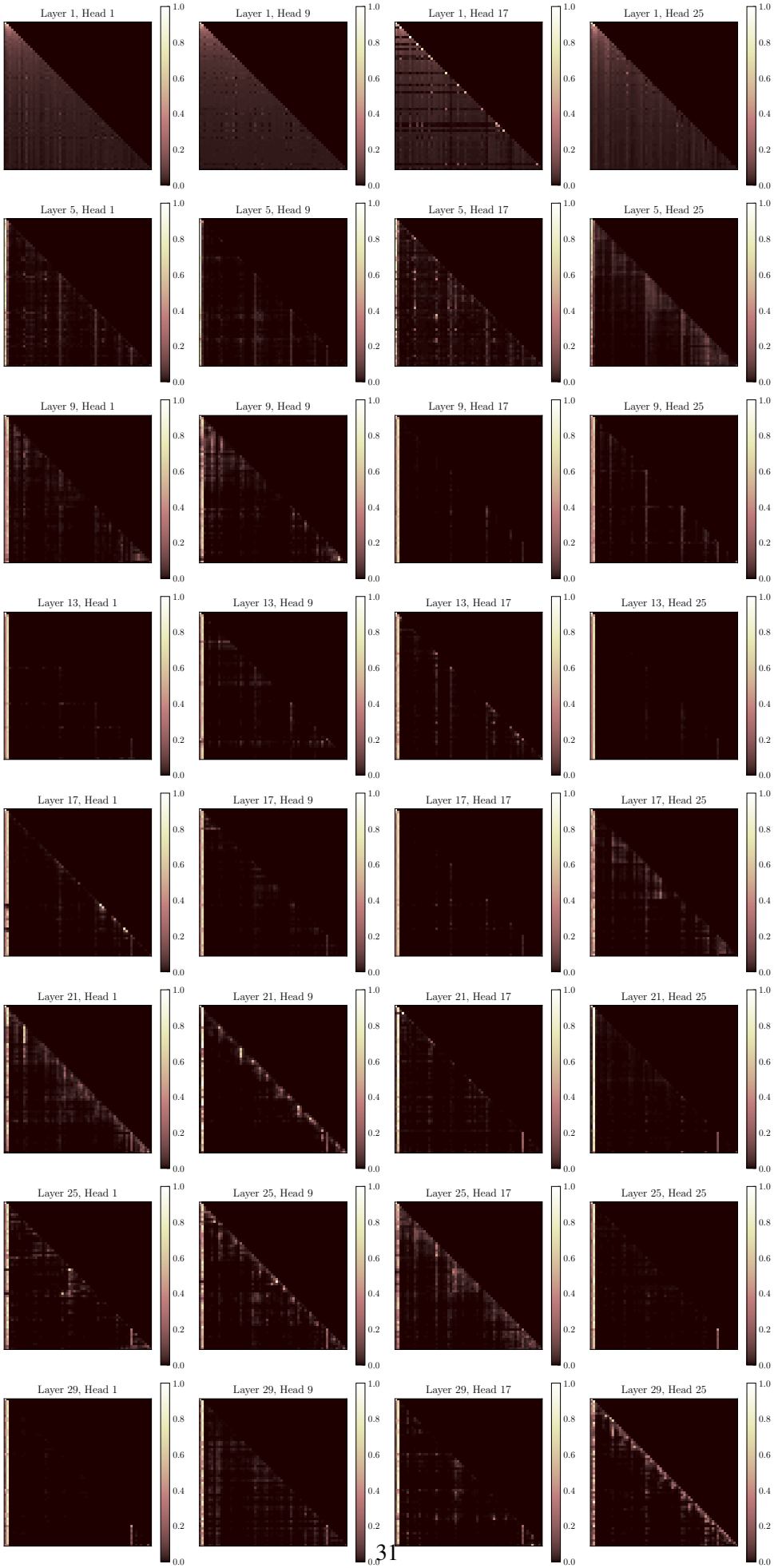


Figure 16: More attention maps of the softmax 1.8B model on sample text input 2.

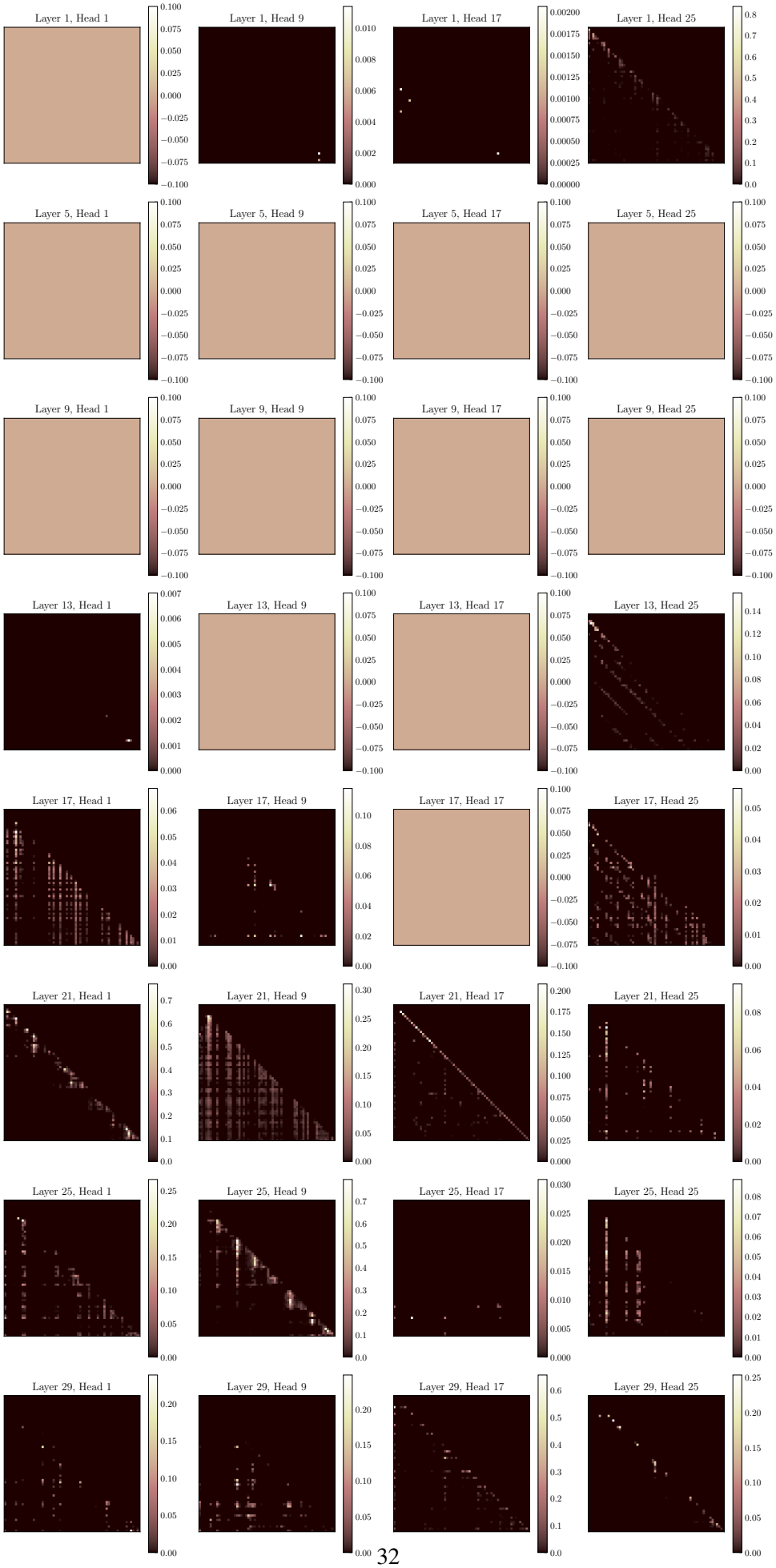


Figure 17: More attention maps of the softpick 1.8B model on sample text input 2.

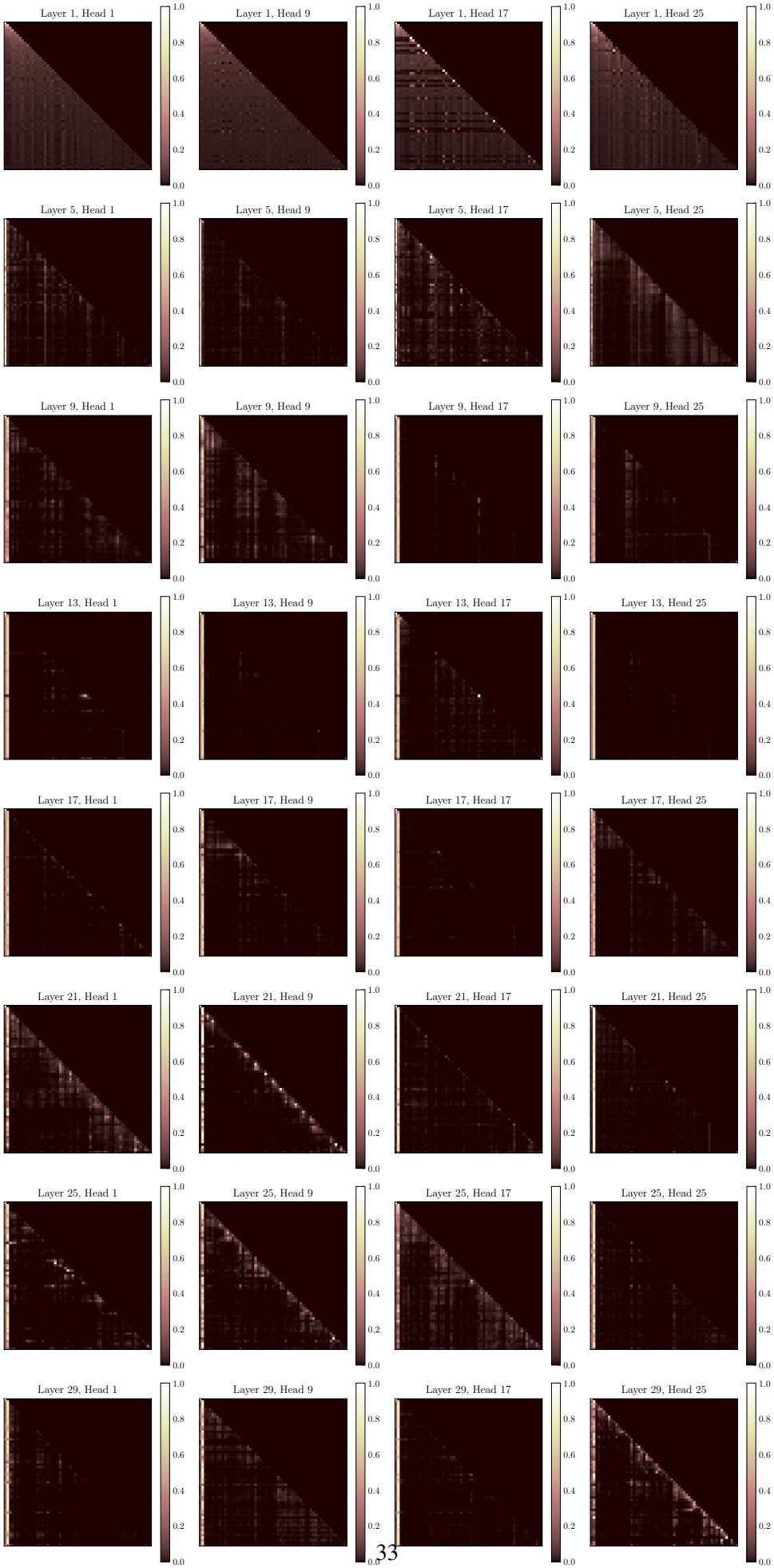


Figure 18: More attention maps of the softmax 1.8B model on sample text input 3.

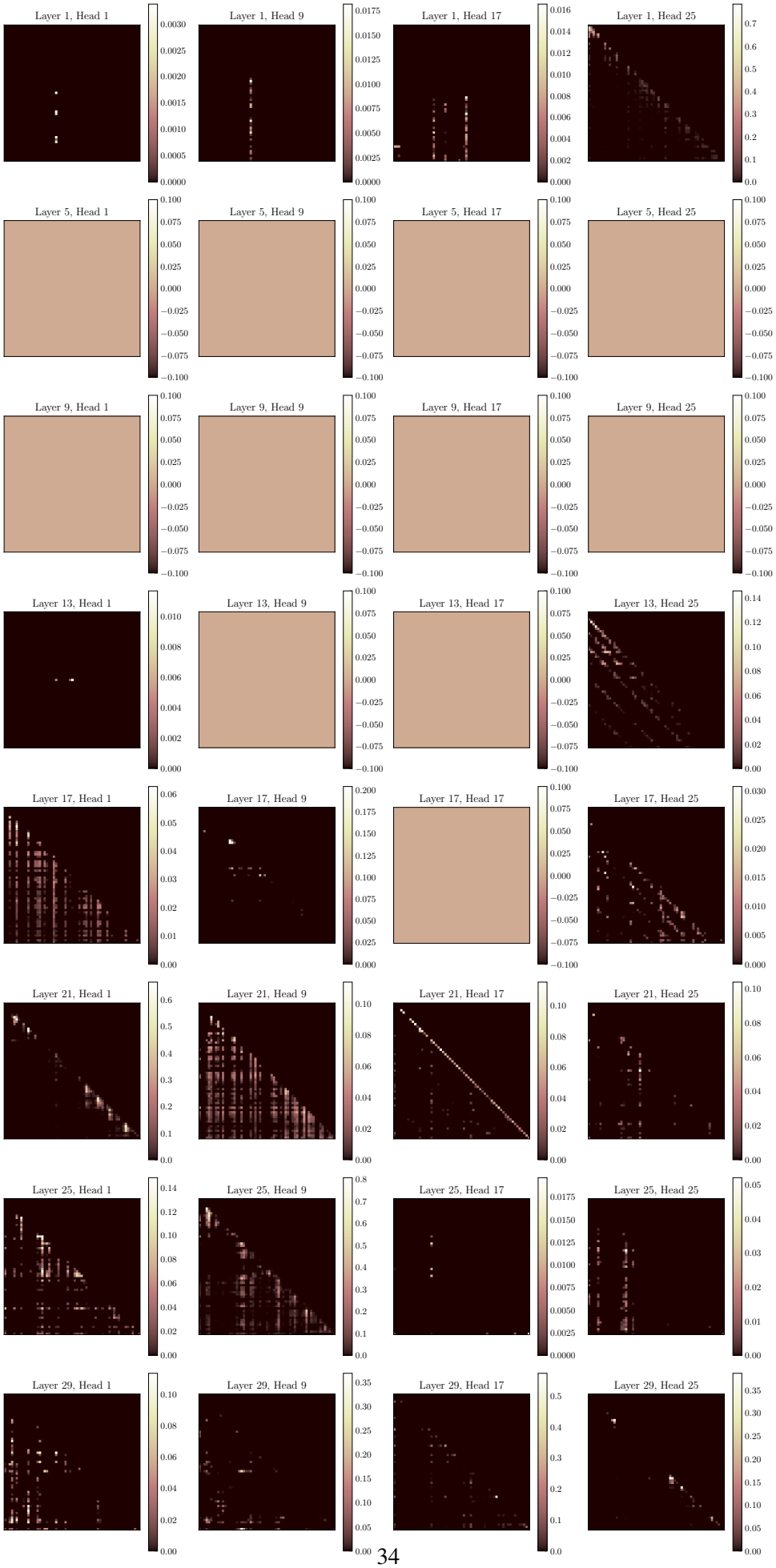


Figure 19: More attention maps of the softpick 1.8B model on sample text input 3.