Self-Adjust Softmax

Chuanyang Zheng[1], Yihang Gao[2], Guoxuan Chen[3], Han Shi[4], Jing Xiong[3], Xiaozhe Ren[4], Chao Huang[3], Zhenguo Li[4], Yu Li[1],

[1]The Chinese University of Hong Kong, [2]National University of Singapore
[3]The University of Hong Kong, [4]Noah's Ark Lab

Contact: cyzheng21@link.cuhk.edu.hk

Abstract

The softmax function is crucial in Transformer attention, which normalizes each row of the attention scores with summation to one. **Usually, tokens with larger attention scores are important for the final prediction. However, the softmax function can face a gradient vanishing issue for such important tokens (e.g., probabilities close to one), leading to optimization difficulties for the important tokens so that the performance may not be better.** In this paper, we propose Self-Adjusting Softmax (SA-Softmax) to address this issue by modifying $softmax(z)$ to $z \cdot softmax(z)$ and its normalized variant $\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$. We theoretically show that SA-Softmax provides enhanced gradient properties compared to the vanilla softmax function. Moreover, SA-Softmax Attention can be seamlessly integrated into existing Transformer models to their attention mechanisms with minor adjustments. We conducted experiments to evaluate the empirical performance of Transformer models using SA-Softmax compared to the vanilla softmax function. These experiments, involving models with up to 2.7 billion parameters, are conducted across diverse datasets, language tasks, and positional encoding methods.

1 Introduction

Transformer-based models (Vaswani et al., 2017) have delivered exceptional performances across widespread applications, including language processing (Zhang et al., 2020; Guo et al., 2022; Ainslie et al., 2023), computer vision (Alexey, 2020; Touvron et al., 2021; Liu et al., 2021b; Chen et al., 2024b; Peebles and Xie, 2023), quantitative research (Zhou et al., 2024; Liu et al., 2021c; Wu et al., 2023), and scientific machine learning (Taylor et al., 2022; Geneva and Zabaras, 2022). A critical component of the Transformer is its attention mechanism, which computes the importance and contribution of each token in a sequence for next-token generation. Central to this mechanism is the softmax function, a mathematical operation that normalizes attention scores token-wise, ensuring a summation of one. This property facilitates probabilistic interpretability and enables a more expressive attention mechanism. For example, Chen et al. (2024a); Xiao et al. (2024a); Xiong et al. (2025) observed that most attention scores are usually concentrated on specific tokens, allowing for more efficient Transformer architectures by discarding tokens with lower accumulative attention scores (Xiong et al., 2024). As a result, the normalized attention scores produced by softmax provide insights into the mechanism of next-token generation in LLMs. Moreover, compared to other attention functions, softmax exhibits some unique and advantageous properties, which contribute to the superior performance of softmax-based Transformer models (Han et al., 2024; Deng et al., 2023).

One of the primary limitations of softmax lies in its susceptibility to the gradient vanishing problem. When input values to the softmax function become highly polarized, i.e., extreme values that are very large or small, the resulting probabilities can exhibit extreme sparsity. This, in turn, leads to gradients that approach zero, impeding effective learning and optimization during backpropagation. Such issues are particularly pronounced in deep architectures, where the accumulation of small gradients can hinder convergence and degrade model performance (Vaswani et al., 2017; Duvvuri and Dhillon, 2024). Several variations have been proposed, including ReLU attention (Nair and Hinton, 2010; Chen et al., 2020; Wortsman et al., 2023; Shen et al., 2023) or sigmoid attention (Ramapuram et al., 2024). These alternatives aim to address specific shortcomings of softmax, such as its sensitivity to extreme input values or its restricted output range, which may limit the behavior of the attention mechanism. However, these approaches

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 7816–7836
November 4-9, 2025

often fall short of achieving comparable stability, interpretability, or general performance, especially in large-scale models where softmax continues to dominate due to its robustness and simplicity.

To address this limitation, we propose a novel modification to the softmax function, introducing Self-Adjusting Softmax (SA-Softmax), which enhances gradient propagation while preserving the probabilistic properties and ranking order of traditional softmax. Our approach builds on theoretical insights and empirical observations. First, we show theoretically that modifying the softmax function to $z \cdot softmax(z)$ amplifies gradient magnitudes, addressing gradient saturation under a range of typical conditions. Building on this formulation, we further refine the formulation to $\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$, incorporating the normalization while enhancing gradient flow. It also maintains the relative ordering of input values, which serves as a critical property for the effectiveness of attention mechanisms. The proposed modification of the vanilla softmax function ensures compatibility with standard Transformer architectures and facilitates seamless integration into existing frameworks.

1. We propose $z \cdot softmax(z)$ as an alternative to the vanilla softmax in the attention mechanism to improve gradient magnitudes, thereby enhancing backpropagation during training. Additionally, we refine $z \cdot softmax(z)$ to $\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$ with normalization, preserving a critical property of softmax while achieving superior performance.

2. We conduct extensive experiments across various datasets, tasks, and models, comparing the proposed SA-Softmax and its variants with the standard $softmax(z)$. Results demonstrate that our approach effectively mitigates gradient vanishing and consistently improves performances across models with different scales.

3. We validate the proposed methods on large-scale pre-training datasets with a training length of 2048. Moreover, we also show the effectiveness of the proposed method in downstream tasks.

2 Related Works

Transformer Attention. The Transformer model, introduced by Vaswani et al. (Vaswani et al., 2017), revolutionized the field of Natural Language Processing (NLP) with its self-attention mechanism. Unlike previous sequence models such as RNNs and LSTMs (Graves and Graves, 2012), Transformer does not rely on recurrent structures and instead uses self-attention to depict relationships between input tokens in parallel. Self-attention, also known as scaled dot-product attention, computes attention scores between input tokens using the query (Q), key (K), and value (V) vectors.

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

where d_k is the dimension of the key vectors (Vaswani et al., 2017). There are also *linearized attention* methods, such as the Linformer (Wang et al., 2020) and Performer (Choromanski et al., 2020), approximating the softmax attention function using low-rank approximations, reducing the computational complexity from $O(n^2)$ to $O(n)$. Another approach to reduce computational complexity is through *sparse attention*, where only a subset of attention scores are computed. For example, the Longformer (Beltagy et al., 2020) uses a combination of local windowed attention and global attention, reducing the attention complexity to $O(n)$ for sequences of length n.

Gradient Vanishing. The gradient vanishing problem refers to the phenomenon where gradients become exceedingly small during backpropagation (Lillicrap et al., 2020). Several works have explored the causes and potential solutions to the gradient vanishing problem. Gradient clipping (Zhang et al., 2019) is one practical solution to mitigate both vanishing and exploding gradients. This technique caps gradients at a maximum value to prevent them from becoming too small or too large. Pascanu (2013) explored gradient clipping in the context of RNNs and found that it can help stabilize training by preventing gradient explosions, which often arise due to large gradients propagating backward through deep networks. The skip connection (He et al., 2016) is the potential way to mitigate the gradient vanishing problem. For softmax attention, the gradient will become zero if one attention probability is too large (Vaswani et al., 2017).

Normalization. Batch Normalization (BN) (Ioffe, 2015) normalizes activations along the batch dimension, while Layer Normalization (LN)

(Ba, 2016) operates along the channel dimension, and Instance Normalization (IN) (Huang and Belongie, 2017) applies BN-like computations independently for each sample. Weight Normalization (WN) (Salimans and Kingma, 2016) instead normalizes filter weights directly. Group Normalization (GN) divides channels into groups, normalizing each group independently, and its computations are unaffected by batch size. Bjorck et al. (2018) show that in networks without BN, large gradient updates can cause diverging loss and uncontrolled activation growth with network depth, limiting learning rates. Similarly, Xu et al. (2019) demonstrates that layer normalization smooths gradients and highlights the importance of mean and variance derivatives, which re-center and re-scale backward gradients beyond forward normalization.

3 Method

3.1 Softmax Attention Mechanism

In the attention mechanism, the weight α_{ij} represents the attention score between token i (the query) and token j (the key). This score quantifies the relative importance of token j to token i, among all tokens in the input sequence. It is formulated as

$$\alpha_{ij} = softmax\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right) = \frac{\exp\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right)}{\sum_{j'} \exp\left(\frac{q_i^T k_{j'}}{\sqrt{d_k}}\right)}, \tag{1}$$

where q_i and k_j are the query and key vectors for tokens i and j, respectively, and d_k is a scaling factor based on the dimensionality of the keys (Vaswani et al., 2017). The softmax function ensures that the resulting attention scores α_{ij} are normalized and can be interpreted as probabilities, summing to one over all tokens j for a given query token i.

The final output of the attention mechanism for each query token i is then calculated as a weighted sum of the values v_j corresponding to each token j in the sequence, with the weight determined by the attention scores α_{ij}. The output of the attention mechanism for token i is defined as

$$\text{Attention}_i(Q, K, V) = \sum_j \alpha_{ij} v_j, \tag{2}$$

where Q, K, and V are matrices representing all queries, keys, and values for a given sequence. This approach allows the model to focus selectively on parts of the sequence that contribute meaningfully to the current query position (Bahdanau et al., 2015).

3.2 Gradient of Softmax Attention

Training Transformer models involves updating all trainable parameters using their gradients. The backpropagation process, which relies on the chain rule, requires the computation of the derivative of the softmax function with respect to its inputs. However, when the input values to the softmax function become extremely large or small, the function can enter flat regions. This results in vanishing gradients, which can hinder the efficient training of model parameters.

We denote the pre-softmax attention scores (i.e., the input to the softmax function before normalization) as

$$z_{i,j} = \frac{q_i^T k_j}{\sqrt{d_k}}, \tag{3}$$

then the derivative of the output attention scores (after passing through the softmax function) with respect to the input $z_{i,j}$ admits

$$\begin{aligned} \frac{\partial \alpha_{ij}}{\partial z_{i,j}} &= \alpha_{ij}(1 - \alpha_{ij}), \\ \frac{\partial \alpha_{ij}}{\partial z_{i,j'}} &= -\alpha_{ij}\alpha_{ij'}, \quad \text{for } j' \neq j. \end{aligned} \tag{4}$$

This Jacobian matrix structure implies that each attention weight depends not only on its own value but also on the values of all other weights. This property, while beneficial for capturing complex relationships, can also make optimization challenging in some scenarios, as explored in the next section.

3.3 Gradient Vanishing in Softmax Attention

One notable issue with softmax attention is the vanishing gradient problem, especially when attention scores become highly peaked. When the softmax output approaches 1 for a specific score and 0 for others, the gradients can become excessively small, slowing down or even halting learning. This is particularly problematic in deeper models where multiple layers of attention are stacked.

The vanishing gradient issue arises from the form of the softmax derivatives. We examine the two cases: Consider the token i in the attention mechanism, and let $z_{i,j}$ and $\alpha_{i,j}$ represent the attention scores of all tokens relative to token i, for $j = 1, 2, \ldots, T$. In the extreme case where one of the attention weights dominates, i.e., $\alpha_{i,j^*} \approx 1$ and $\alpha_{i,j} \approx 0$, for $j \neq j^*$. Then Equation 4 implies that $\frac{\partial \alpha_{i,j}}{\partial z_{i,j'}} \approx 0$ for all $j, j' = 1, 2, \ldots, T$. This result indicates that, under such circumstances, the derivative of the output attention weights with respect to the input pre-softmax attention scores vanishes, leading to gradient vanishing across all tokens. Moreover, in a milder case where $\alpha_{i,j} \approx 0$ holds for some j, we have $\frac{\partial \alpha_{i,j}}{\partial z_{i,j'}} \approx 0$ and $\frac{\partial \alpha_{i,j'}}{\partial z_{i,j}} \approx 0$ for $j' = 1, 2, \ldots, T$. This means that the derivative of the softmax function partially vanishes, if input and output correspond to $\alpha_{i,j}$ and $z_{i,j}$ of token j, resulting in gradient vanishing for those specific tokens.

In summary, when the extreme case arises where one attention score dominates while others approach zero, the softmax mechanism suffers from complete gradient vanishing for all tokens, leading to slow training and failure in gradient backpropagation. In the milder case, where some attention scores are close to zero, the derivatives associated with these tokens and their attention scores still vanish, causing suboptimal training performance.

The extreme case, where some attention scores approach zero, frequently occurs in attention mechanisms due to the exponential function's sensitivity to large values. In the following section, we first introduce a modification to the vanilla softmax, called Self-Adjusting Softmax (SA-Softmax), which is theoretically guaranteed to enhance and amplify gradient propagation. Additionally, we propose several variants of SA-Softmax, designed to further improve the effectiveness and stability of Transformer models by incorporating normalization techniques.

3.4 Self-Adjusting Softmax

To address the issue of potential gradient vanishing of the softmax function, we propose modifying the attention mechanism by scaling the softmax output with its input, called Self-Adjusting Softmax (SA-Softmax). Specifically, we redefine the output of attention scores as follows:

$$\beta_{i,j} = z_{i,j} \cdot softmax(z_{i,j}), \tag{5}$$

where $z_{i,j}$ is the pre-softmax attention score of the token i corresponding to the token j. This modification introduces an additional scaling term $z_{i,j}$ to calculate the final attention scores besides the standard softmax function, amplifying the gradient propagation compared to the original formulation.

Gradient Analysis of SA-Softmax. Let us evaluate the gradient of the modified attention scores $\beta_{i,j}$ with respect to the input $z_{i,j'}$. Differentiating $\beta_{i,j} = z_{i,j} \cdot softmax(z_{i,j})$ with respect to $z_{i,j'}$, we have

$$\begin{aligned} \frac{\partial \beta_{i,j}}{\partial z_{i,j}} &= softmax(z_{i,j}) + z_{i,j} \cdot \frac{\partial softmax(z_{i,j})}{\partial z_{i,j}} \\ &= \alpha_{i,j} + z_{i,j} \cdot \alpha_{i,j}(1 - \alpha_{i,j}), \end{aligned} \tag{6}$$

and

$$\begin{aligned} \frac{\partial \beta_{i,j}}{\partial z_{i,j'}} &= z_{i,j'} \cdot \frac{\partial softmax(z_{i,j})}{\partial z_{i,j'}} \\ &= -z_{i,j} \cdot \alpha_{i,j}\alpha_{i,j'}, \end{aligned} \tag{7}$$

with $j' \neq j$.

Implications for Gradient Vanishing. According to Equation 6, considering the extreme case where $\alpha_{i,j^*} \approx 1$, the gradient is amplified as the first term α_{i,j^*} is dominant and governs the gradient. Moreover, for tokens where $\alpha_{i,j} \approx 0$, the gradient is enhanced by the dynamic and self-adjusting scaler $z_{i,j}$, as demonstrated in Equations 6 and 7. Therefore, our method significantly enhances the gradient propagation for tokens with $\alpha_{i,j'} \approx 1$ and improves the gradient for tokens satisfying $\alpha_{i,j} \approx 0$ through the dynamic and self-adjusting scalers.

Comparison with Standard Softmax. In the standard softmax attention, the gradient softmax output $\alpha_{i,j}$ with respect to the input $z_{i,j}$ tends to vanish when $\alpha_{i,j}$ approaches 0 or 1. By introducing an additional self-adjusting term in the attention score computation (i.e., modifying $softmax(z)$ to $z \cdot softmax(z)$), we allow for a more resilient gradient. As shown in Equations 6 and 7, this approach may not completely eliminate the gradient vanishing problem, it significantly mitigates its effects, especially in cases with long sequences or deep networks, where gradients from softmax attention typically diminish (Vaswani et al., 2017).

3.5 Variants of SA-Softmax

In this section, we further develop some variants of SA-Softmax, by utilizing normalization techniques on the self-adjusting term to further stabilize the training process.

Variant 1: $(z - z_{\min}) \cdot softmax(z)$ A notable potential inconsistency in SA-Softmax arises from the negative attention scores, which can lead to unpredictable and difficult-to-interpret behavior in the attention mechanism.

To address this issue, we propose a modified approach that shifts the self-adjusting term by its minimum value along the sequence. Specifically, we reformulate the attention computation as $(z - z_{\min}) \cdot softmax(z)$, where $z_{\min}$ represents the minimum value of z across the sequence. This modification ensures that all attention scores are non-negative, thereby stabilizing the scaling effect across different z_i:

$$\gamma_{i,j} = (z_{i,j} - z_{i,\min}) \cdot softmax(z_{i,j}) \tag{8}$$

where $z_{i,\min} := \min\{z_{i,j} : j = 1, 2, \ldots, T\}$ denotes the minimum values of $z_{i,j}$ along the sequence. This adjustment enhances the robustness of the attention mechanism by ensuring consistency and stability in the scaling of attention scores.

Variant 2: $\frac{z - z_{\min}}{z_{\max} - z_{\min}} \cdot softmax(z)$ The first variant introduced a shift in the self-adjusting term to ensure the non-negativity of attention scores. Building on this idea, a more widely used technique is normalization. To further stabilize training, we normalize the self-adjusting term to $\frac{z - z_{\min}}{z_{\max} - z_{\min}} \in [0, 1]$. Therefore, the attention scores are calculated as follows:

$$\delta_{i,j} = \frac{z_{i,j} - z_{i,\min}}{z_{i,\max} - z_{i,\min}} \cdot softmax(z_{i,j}), \tag{9}$$

where $z_{i,\max} = \max\{z_{i,j} : j = 1, 2, \ldots, T\}$ denotes the maximum value of $z_{i,j}$ along the sequence, and $z_{i,\min} := \min\{z_{i,j} : j = 1, 2, \ldots, T\}$ represents the minimum value. This normalization ensures that the self-adjusting term $\frac{(z - z_{\min})}{z_{\max} - z_{\min}}$ lies within the bounded region $[0, 1]$, resulting in a stable scaling effect across different input distributions. This formulation provides a stable gradient computation, as the adjusting term, normalized by $x_{\max} - x_{\min}$, prevents excessively large values and ensures all values fall within a bounded range.

Variant 3: $\frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)} \cdot softmax(z)$ To make it easier for the model optimization, we add a threshold to further normalize the $\frac{(z - z_{\min})}{z_{max} - z_{min}}$ $z - z_{\min}$ to $\frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)} \in$ [0,1].

$$\delta_i = \frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)} \cdot softmax(z_i) \tag{10}$$

where $z_{\max} = \max(z)$ and $z_{\min} = \min(z)$. When the z becomes positive and $z_{\max} - x_{\min} \gg 0$, $\frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)}$ will close to 1 so that the $\frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)} \cdot softmax(z_i)$ degrades to $softma(z)$.

4 Experiment

Datasets. We use the Arxiv and Books3 dataset for our experiments. The training is conducted using a batch size of 512 or 1024 sequences, each with a sequence length from length 128 to length 2048. The models are trained for 50000 iterations. Throughout the training process, we monitor both the training and the gradient. We also evaluate our methods in downstream datasets, such as sequence classification and machine translation.

Experiment Setting. We begin by conducting experiments on the Arxiv and Books datasets, evaluating model performance across training sequence lengths ranging from 128 to 1024 with various positional encodings. Next, we validate our method on models of varying scales, from 125M to 2.7B parameters. Following this, we analyze the performance of different model variants and assess their ability to extrapolate to longer sequence lengths. Subsequently, we further validate the method on downstream tasks, including text classification and machine translation. Lastly, we visualize the gradient behavior across different methods to provide deeper insights into their effectiveness. The experiment setting details are presented in Appendix A. By default, we use the $\frac{z - min(z_{\min}, 0)}{max(0, z_{\max}) - min(z_{\min}, 0)} \cdot softmax(z)$.

4.1 Compare with Baseline Performance

The SA-Softmax could improve performance across different position encoding. The results in Table 1 highlight the effectiveness of SA-Softmax in improving perplexity for Kerple (Chi et al., 2022), FIRE (Li et al., 2023), RoPE (Su et al., 2024) and DAPEV2-Kerple (Zheng et al., 2024b). Without SA-Softmax (✗), RoPE achieves perplexities of 89.60, 38.29, and 28.78 for sequence lengths 128, 512, and 1024, respectively. With SA-Softmax (✓), these values drop to 89.36, 37.57, and 27.57, showcasing its contribution. DAPEV2-Kerple exhibits more significant improvements, with perplexities dropping from 84.33, 36.25, and 26.86 to 83.63, 35.93, and 26.56 across the respective sequence lengths when SA-Softmax is applied.

Table 1: The perplexity on Arxiv and Books dataset with different position encodings.

Data	PE	SA-Softmax	128	512	1024
Arxiv	Kerple	✗	14.61	6.70	5.47
Arxiv	Kerple	✓	14.51	6.66	5.44
Arxiv	FIRE	✗	14.76	6.67	5.43
Arxiv	FIRE	✓	14.46	6.59	5.38
Arxiv	RoPE	✗	14.86	6.70	5.52
Arxiv	RoPE	✓	14.62	6.63	5.49
Arxiv	DAPEV2-Kerple	✗	14.27	6.63	5.26
Arxiv	DAPEV2-Kerple	✓	**14.10**	**6.36**	**5.20**
Books	Kerple	✗	88.88	38.46	28.65
Books	Kerple	✓	87.56	37.95	28.37
Books	FIRE	✗	88.48	38.12	28.57
Books	FIRE	✓	87.98	37.34	28.00
Books	RoPE	✗	89.60	38.29	28.78
Books	RoPE	✓	89.36	37.57	28.57
Books	DAPEV2-Kerple	✗	84.33	36.25	26.86
Books	DAPEV2-Kerple	✓	**83.63**	**35.93**	**26.56**

This demonstrates the universal applicability of SA-Softmax to enhance position encoding methods.

DAPEV2-Kerple achieves the best performance, especially with SA-Softmax. Among all tested configurations, DAPEV2-Kerple combined with SA-Softmax yields the lowest perplexity scores, outperforming both the baseline RoPE and RoPE with SA-Softmax. For instance, at a sequence length of 1024, DAPEV2-Kerple with SA-Softmax achieves a perplexity of 26.56, compared to 30.29 for RoPE with SA-Softmax. This superiority is consistent across shorter sequence lengths as well, with DAPEV2-Kerple maintaining its advantage even without SA-Softmax. These results confirm that DAPEV2-Kerple is the most effective position encoding method for reducing perplexity in language modeling tasks.

The proposed SA-Softmax improves both short and long-sequence modeling. The analysis indicates that SA-Softmax enhances performance at all sequence lengths, demonstrating its ability to handle both short-range and long-range dependencies effectively. The reductions in perplexity are kept at longer sequence lengths, particularly for DAPEV2-Kerple (e.g., a drop from 26.86 to 26.56 for length 1024), suggesting that SA-Softmax still provides better optimization for long contexts. This capability is critical for modern language models that often deal with extensive input sequences.

The SA-Softmax still works well on longer training length. The results in Table 2 demonstrate the impact of using SA-Softmax (SA-Softmax, indicated by ✓) versus not using it (✗) on the Arxiv and Books datasets under different positional encodings (RoPE and DAPEV2-Kerple) across evaluation lengths from 128 to 2048, with a training length of 2048. For both datasets and positional encodings, SA-Softmax consistently improves performance, as evidenced by lower perplexity values. On the Arxiv dataset, DAPEV2-Kerple with SA-Softmax achieves the best results, with perplexity decreasing from 8.70 at length 128 to 4.61 at length 2048, outperforming baseline DAPEV2-Kerple in all cases. For the Books dataset, DAPEV2-Kerple combined with SA-Softmax achieves the lowest perplexity. Similarly, RoPE with SA-Softmax also achieves better performance than baseline RoPE on the Arxiv and Books dataset from evaluation length 128 to length 2048. These results indicate that SA-Softmax effectively enhances model performance, and works well on longer training lengths.

Table 2: The perplexity on the Arxiv and Books dataset with training length 2048, evaluated from length 128 to length 2048.

Dataset	PE	SA-Softmax	128	256	512	1024	2048
Arxiv	RoPE	✗	9.14	7.53	5.42	5.00	4.95
Arxiv	RoPE	✓	9.05	7.46	5.38	4.96	4.92
Arxiv	DAPEV2-Kerple	✗	8.80	7.22	5.16	4.74	4.64
Arxiv	DAPEV2-Kerple	✓	8.70	7.15	5.13	4.72	4.61
Books	RoPE	✗	35.99	31.32	25.97	24.32	22.65
Books	RoPE	✓	35.71	31.15	25.906	24.23	22.63
Books	DAPEV2-Kerple	✗	34.13	29.54	24.28	22.60	20.85
Books	DAPEV2-Kerple	✓	33.60	29.16	24.06	22.40	20.71

4.2 The performance on Larger Model Size

Table 3: The perplexity on the Arxiv and Books dataset with different model sizes, with training length 512.

PE	Dataset	SA-Softmax	125M	350M	1.3B	2.7B
RoPE	Arxiv	✗	6.70	6.26	6.01	5.93
RoPE	Arxiv	✓	6.63	6.20	5.92	5.83
DAPEV2-Kerple	Arxiv	✗	6.63	6.02	5.79	5.70
DAPEV2-Kerple	Arxiv	✓	6.36	5.97	5.74	5.65
RoPE	Book	✗	38.29	33.81	30.94	29.98
RoPE	Book	✓	37.57	33.17	30.24	29.15
DAPEV2-Kerple	Book	✗	36.25	32.20	29.32	28.15
DAPEV2-Kerple	Book	✓	35.93	31.82	28.91	27.75

SA-Softmax still enhances performance for larger model sizes. Table 3 demonstrates the effectiveness of SA-Softmax across different model sizes, ranging from 125M to 2.7B parameters, on the Books and Arxiv datasets. The results show that, as model size increases, the integration of SA-Softmax consistently improves performance compared to the baseline (✗). For example, with RoPE on the Books dataset, the perplexity at 2.7B parameters decreases from 29.98 to 29.15 when SA-Softmax is applied. Similarly, for DAPEV2-Kerple on the same dataset, perplexity improves from 28.15 to 27.75 at the largest model size, highlighting the compatibility of SA-Softmax with large-scale models.

SA-Softmax delivers consistent improvements across datasets. The results are consistent across both the Books and Arxiv datasets, confirming the generalizability of SA-Softmax. On the Arxiv dataset, for instance, RoPE with SA-Softmax reduces perplexity across all model sizes, from 6.70 to 6.63 at 125M parameters and from 5.93 to 5.83 at 2.7B parameters. Similar trends are observed for DAPEV2-Kerple, where the improvements are slightly less pronounced but still consistent. These findings indicate that SA-Softmax is robust and effective across diverse text corpora and model configurations.

4.3 The Performance of Different Variants

Table 4: The perplexity on the Books dataset with training length 512, compared to baselines.

Length	**Variant**	RoPE	$DAPEV2-Kerple$
128	$softmax(z)$	89.60	84.33
128	$z * softmax(z)$	85.98	82.63
128	$(z - z_{max}) * softmax(z)$	86.10	**82.08**
128	$\frac{(z-z_{\min})}{z_{max}-z_{min}} \cdot softmax(z)$	89.36	84.21
128	$\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$	**89.36**	83.63
512	$softmax(z)$	38.29	36.25
512	$z * softmax(z)$	38.07	35.82
512	$(z - z_{max}) * softmax(z)$	39.47	**35.70**
512	$\frac{(z-z_{\min})}{z_{max}-z_{min}} \cdot softmax(z)$	37.93	36.21
512	$\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$	**37.57**	35.93
1024	$softmax(z)$	28.78	26.86
1024	$z * softmax(z)$	28.96	26.62
1024	$(z - z_{max}) * softmax(z)$	30.26	26.78
1024	$\frac{(z-z_{\min})}{z_{max}-z_{min}} \cdot softmax(z)$	28.73	26.74
1024	$\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$	**28.57**	**26.56**

The $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ variant is a robust default choice. Table 4 shows that this variant consistently improves perplexity across all sequence lengths (128, 512, and 1024) and for both RoPE and DAPEV2-Kerple position encodings. For RoPE, this variant achieves the best performance at 128, 512, and 1024 lengths, and for DAPEV2-Kerple, it also achieves better performance than baseline from length 128 to length 1024. This suggests that the variant balances performance improvements across different configurations, making it a reliable choice when specific experimental conditions are not predefined.

Optimal SA-Softmax variants depend on the experimental setup and position encoding method. The results reveal that different variants perform best under different conditions. For example, the $(z - z_{\max}) \cdot softmax(z)$ variant achieves the lowest perplexity for DAPEV2-Kerple at lengths 128 (82.08) and 512 (35.70), outperforming all other configurations for these specific setups. Similarly, the standard $z \cdot softmax(z)$ variant shows competitive performance at 1024-length sequences, achieving 26.62 for DAPEV2-Kerple. These variations highlight that while certain formulations may work well across the board, optimal performance often depends on the interaction between the sequence length and the positional encoding technique.

4.4 Performance on Downstream Tasks

Table 5: The performance on downstream tasks, with 125M model size and 300B training tokens.

Dataset	Metrics	Softmax	SA-Softmax
Lambda	ppl↓	21.63	**20.43**
WikiText	ppl↓	27.57	**27.47**
ARCEasy	acc↑	45.92	**47.52**
HellaSwag	acc↑	30.34	**30.42**
PiQA	acc↑	64.64	**64.69**
OpenBookQA	acc↑	16.80	**18.00**
SciQ	acc↑	76.80	**77.60**
Winogrande	acc↑	51.54	**51.85**

Pretrain Setting. We pre-train a 125M model with 300B tokens from the Pile dataset and evaluate the performance on the downstream tasks (Black et al., 2022). Following the setting of previous works (Black et al., 2022), the training steps are 143000 with a training length of 2048 and a global batch size 1024.

The SA-Softmax achieves better performance than baseline softmax. As shown in Table 5, SA-Softmax demonstrates superior performance compared to the baseline Softmax model across a wide range of tasks. The improvements are particularly notable in tasks requiring language modeling and reasoning. For instance, on the Lambda dataset (Paperno et al., 2016), SA-Softmax achieves a significant reduction in perplexity (ppl) from 21.63 to 20.43. Similarly, on WikiText (Merity et al., 2017), SA-Softmax reduces perplexity from 27.57 to 27.47. Additionally, it attains higher accuracy (acc) on several datasets, including ARCEasy (Clark et al., 2018b), HellaSwag

(Zellers et al., 2019), PiQA(Clark et al., 2018b), OpenBookQA(Bisk et al., 2020), SciQ (Welbl et al., 2017), and Winogrande (Kocijan et al., 2020). These results underscore the effectiveness of SA-Softmax, even with a relatively small model size, when trained on a large-scale corpus. with potential for further improvements through increased model size and training data.

5 The Performance on Classification and Translation Taks

Table 6: Accuracy achieved on various downstream classification tasks. The **Improve** Δ column shows the improvement in percentage points when using SA-Softmax compared to Softmax.

Dataset	Softmax	SA-Softmax	Δ
AG-News	93.75	95.83	2.08
DBPedia	99.11	100	0.09
Yelp-Review	65.00	67.50	2.50
YahooNews	72.92	73.96	1.04
AmazonNews	62.50	68.75	6.25

Table 7: Performance comparison on IWSLT2017 machine translation tasks. Bold values indicate the best performance for each pair.

Input	SA-Softmax	en	nl	de	it	ro
en	✗	-	25.98	22.53	24.08	21.98
en	✓	-	**26.25**	**23.57**	**24.67**	**22.21**
nl	✗	31.43	-	18.57	15.89	14.67
nl	✓	**32.10**	-	**19.21**	**16.14**	**15.04**
de	✗	26.83	18.44	-	14.55	**13.72**
de	✓	**27.49**	**18.76**	-	**14.76**	13.57
it	✗	28.31	15.50	15.65	-	15.77
it	✓	**28.55**	**15.65**	**15.97**	-	**16.09**
ro	✗	28.75	15.42	15.72	18.27	-
ro	✓	**29.21**	**16.71**	**16.11**	**18.54**	-

The Performance on Classification and Translation Tasks, the experiment setting is presented in Appendix A. The results across both classification and machine translation tasks indicate the consistent effectiveness of SA-Softmax over traditional methods such as Softmax. On classification tasks (Table 6), SA-Softmax achieves notable improvements across all datasets, with the highest improvement observed on the AmazonNews dataset (+6.25 percentage points) and significant gains on AG-News (+2.08), Yelp-Review (+2.50), and YahooNews (+1.04). On machine translation tasks (Table 7), SA-Softmax consistently outperforms baseline methods across multiple language pairs. These results collectively indicate that SA-Softmax enhances both the accuracy and generalization capabilities of models, making it a promising alternative to traditional Softmax-based approaches.

5.1 Visualization of Attention Output

We also visualize the attention probability for different methods in Appendix F.

$\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ **and** $softmax(z)$ **present similar pattern, compared to z** $\cdot$ $softmax(z)$ **.** As shown Appendix F, the $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ range may be larger than baseline $softmax(z)$. Also, the $softmax(z)$ and $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ are more similar, compared to $z * softmax(z)$. The $z * softmax(z)$ may have some special attention patterns, as shown in layer 3 and layer 10.

Attention scores can be negative, contrary to previous beliefs that attention scores must be positive. In prior work, the research community has attempted to replace softmax attention with ReLU attention or Sigmoid attention, operating under the assumption that attention scores should always be positive (Nair and Hinton, 2010; Chen et al., 2020; Wortsman et al., 2023; Shen et al., 2023) (Ramapuram et al., 2024) However, in this work, we successfully demonstrate that attention scores can indeed take on negative values. As shown in Appendix F, we observe that transformers can still be effectively trained even when the attention scores contain negative elements and the sum of each row is not strictly equal to one.

6 Conclusion

We propose Self-Adjusting Softmax, a modification designed to improve gradient dynamics and enhance performance in transformers. To demonstrate the effectiveness of SA-Softmax, we conduct extensive experiments, including analyses with various positional encodings, training lengths, and model sizes and different variants. Additionally, we evaluate SA-Softmax on downstream tasks, where the variant $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ consistently proves to be the most effective across diverse settings. This powerful adjustment significantly enhances transformer scalability and generalization, offering promising potential for a wide range of applications.

Limitations

The proposed method needs to find the max and min values first for the normalization. Therefore, there may be additional costs.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. 2023. CoLT5: Faster long-range transformers with conditional computation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Dosovitskiy Alexey. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*.

Jimmy Lei Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. 2018. Understanding batch normalization. *Advances in neural information processing systems*, 31.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2019. Accurate computation of the log-sum-exp and softmax functions. *arXiv preprint arXiv:1909.03469*.

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2021. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, and Dougal Maclaurin. 2018. JAX: Autograd and XLA. https://github.com/google/jax. Accessed: 2024-09-25.

John S Bridle. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Dengsheng Chen, Jun Li, and Kai Xu. 2020. Arelu: Attention-based rectified linear unit. *arXiv preprint arXiv:2006.13858*.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. 2024a. SepLLM: Accelerate large language models by compressing one segment into one separator. *arXiv preprint arXiv:2412.12094*.

Junsong Chen, Jincheng YU, Chongjian GE, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. 2024b. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *The Twelfth International Conference on Learning Representations*.

Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, and Alexander Rudnicky. 2022. Kerple: Kernelized relative positional embedding for length extrapolation. *Advances in Neural Information Processing Systems*, 35:8386–8399.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019a. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019b. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kyunghyun Cho. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Q Davis, Afroz Mohiuddin, Łukasz Kaiser, et al. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

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

Google Cloud. 2023. Google cloud tpu v5e: Next-generation ai hardware for large-scale model training. https://cloud.google.com/blog/products/ai-machine-learning/introducing-tpu-v5e. Accessed: 2024-09-25.

George E Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, et al. 2023. Benchmarking neural network training algorithms. *arXiv preprint arXiv:2306.07179*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Tri Dao, Daniel Fu, Xinyang G Wang, et al. 2024. Flashattention 2: Faster attention with better memory scheduling. *arXiv preprint arXiv:2401.14155*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Yichuan Deng, Zhao Song, and Tianyi Zhou. 2023. Superiority of softmax: Unveiling the performance edge over linear attention. *arXiv preprint arXiv:2310.11685*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Timothy Dozat. 2016. Incorporating nesterov momentum into adam.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sai Surya Duvvuri and Inderjit S Dhillon. 2024. Laser: Attention with exponential transformation. *arXiv preprint arXiv:2411.03493*.

Zafeirios Fountas, Martin A Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou-Ammar, and Jun Wang. 2024. Human-like episodic memory for infinite context llms. *arXiv preprint arXiv:2407.09450*.

Team Gemini. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Nicholas Geneva and Nicholas Zabaras. 2022. Transformers for modeling physical systems. *Neural Networks*, 146:272–289.

Justin M. Gilmer, George E. Dahl, Zachary Nado, Priya Kasimbeg, and Sourabh Medapati. 2023. init2winit: a jax codebase for initialization, optimization, and tuning research.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Research Google. 2023. Pax: A JAX-based neural network training framework. https://github.com/google/paxml. Accessed: 2024-09-25.

Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Proc. Interspeech*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. *Findings of the Association for Computational Linguistics: NAACL*.

Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. 2024. Bridging the divide: Reconsidering softmax and linear attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510.

Sergey Ioffe. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 252–262.

Vid Kocijan, Elias Chamorro-Perera, Damien Sileo, Jonathan Raiman, and Peter Clark. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2002. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2023. Functional interpolation for relative positions improves long context transformers. *arXiv preprint arXiv:2310.04418*.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. 2020. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021a. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021b. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2021c. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the Twenty-ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4513–4519.

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Team Meta-AI. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018a. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018b. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

MLCommons. Bert dataset documentation. `https://github.com/mlcommons/training/blob/master/language_model/tensorflow/bert/dataset.md`. Accessed: 2024-10-16.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

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.

R Pascanu. 2013. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. Wic: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, volume 21, pages 1–67.

Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, et al. 2024. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint arXiv:2409.04431*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021a. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021b. Efficient routing transformers: Dynamic token interaction models for natural language processing. *arXiv preprint arXiv:2003.05997*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Kai Shen, Junliang Guo, Xu Tan, Siliang Tang, Rui Wang, and Jiang Bian. 2023. A study on relu and softmax in transformer. *arXiv preprint arXiv:2302.06461*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, et al. 2023. A survey of reasoning with foundation models. *arXiv preprint arXiv:2312.11562*.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3261–3275. Curran Associates, Inc.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

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.

Mitchell Wortsman, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. 2023. Replacing softmax with relu in vision transformers. *arXiv preprint arXiv:2309.08586*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, Song Han, and Maosong Sun. 2024a. InfLLM: Unveiling the intrinsic capacity of LLMs for understanding extremely long sequences with training-free memory. *arXiv preprint arXiv:2402.04617*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Jing Xiong, Chengming Li, Min Yang, Xiping Hu, and Bin Hu. 2022a. Expression syntax information bottleneck for math word problems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2166–2171.

Jing Xiong, Zixuan Li, Chuanyang Zheng, Zhijiang Guo, Yichun Yin, Enze Xie, Zhicheng Yang, Qingxing Cao, Haiming Wang, Xiongwei Han, et al. 2023a. Dq-lore: Dual queries with low rank approximation re-ranking for in-context learning. *arXiv preprint arXiv:2310.02954*.

Jing Xiong, Jianghan Shen, Fanghua Ye, Chaofan Tao, Zhongwei Wan, Jianqiao Lu, Xun Wu, Chuanyang Zheng, Zhijiang Guo, Lingpeng Kong, et al. 2024. Uncomp: Uncertainty-aware long-context compressor for efficient large language model inference. *arXiv preprint arXiv:2410.03090*.

Jing Xiong, Jianghan Shen, Chuanyang Zheng, Zhongwei Wan, Chenyang Zhao, Chiwun Yang, Fanghua Ye, Hongxia Yang, Lingpeng Kong, and Ngai Wong. 2025. Parallelcomp: Parallel long-context compressor for length extrapolation. *arXiv preprint arXiv:2502.14317*.

Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, et al. 2023b. Trigo: Benchmarking formal mathematical proof reduction for generative language models. *arXiv preprint arXiv:2310.10180*.

Jing Xiong, Zhongwei Wan, Xiping Hu, Min Yang, and Chengming Li. 2022b. Self-consistent reasoning for solving math word problems. *arXiv preprint arXiv:2210.15373*.

Ruibin Xiong, Yingquan Yang, Di He, Kai Zheng, Shuxin Zheng, Yaliang Lan, Jingdong Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32.

Yang You, Jing Li, Sashank Reddi, Jason Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

Shuailong Zhang, Huanbo Liu, Shuyan Liu, Yuwei Wang, Jiawei Liu, Zhiyu Gao, Wei Xu, Yiming Xu, Xin Sun, Lei Cui, et al. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Chuanyang Zheng, Yihang Gao, Han Shi, Minbin Huang, Jingyao Li, Jing Xiong, Xiaozhe Ren, Michael Ng, Xin Jiang, Zhenguo Li, et al. 2024a. Dape: Data-adaptive positional encoding for length extrapolation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chuanyang Zheng, Yihang Gao, Han Shi, Jing Xiong, Jiankai Sun, Jingyao Li, Minbin Huang, Xiaozhe Ren, Michael Ng, Xin Jiang, et al. 2024b. Dape v2: Process attention score as feature map for length extrapolation. *arXiv preprint arXiv:2410.04798*.

Jin Peng Zhou, Charles E Staats, Wenda Li, Christian Szegedy, Kilian Q Weinberger, and Yuhuai Wu. 2024. Don't trust: Verify – grounding LLM quantitative reasoning with autoformalization. In *The Twelfth International Conference on Learning Representations*.

Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, et al. 2024. Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *arXiv preprint arXiv:2406.15486*.

A Model Configuration

Pretrain Setting. All experiments are conducted on 8 GPUs. The 125M and 350M model configurations are the following.

Table 8: **Model Configurations.**

	125M	**350M**
Training sequence length	512	512
Batch size	2×8	2×8
Number of iterations	50k	50k
Dropout prob.	0.0	0.0
Attention dropout prob.	0.0	0.0
Attention head	12	16
Feature dimension	768	1024
Layer number	12	24
Optimizer	Adam	Adam
Optimizer parameter betas	[0.9, 0.95]	[0.9, 0.95]
Learning rate	$6e-4$	$3e-4$
Precision	float132	float32

Experiment Setting for Classification and Translation tasks. For the sequence classification tasks presented in Table 6, the feature dimension is set to 128, with 2 attention heads and 6 layers. The datasets are AGNews, DBPedia, Yelp-Review, YahooNews, AmazonNews (Zhang et al., 2015). In contrast, for the machine translation tasks shown in Table 7, the feature dimension is increased to 512, with 8 attention heads and 12 layers. The dataset comes from IWSLT2017 datasets (Cettolo et al., 2017).

B Error Bar

Table 9: The perplexity on Books3 dataset with three random seeds.

Method	**SA-Softmax**	Mean	Std
Kerple	✗	38.21	0.3873
Kerple	✓	37.71	0.3826
FIRE	✗	38.00	0.2211
FIRE	✓	37.24	0.2786
RoPE	✗	38.03	0.2165
RoPE	✓	37.48	0.3287
DAPEV2-Kerple	✗	35.92	0.3821
DAPEV2-Kerple	✓	35.58	0.4037

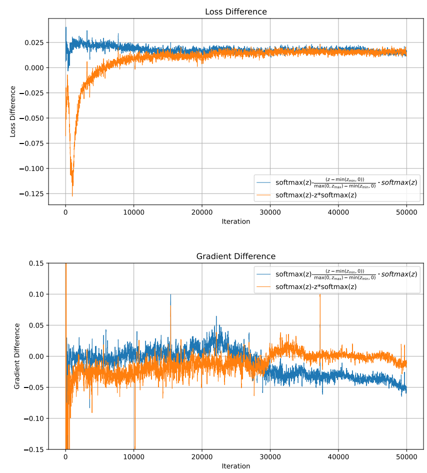

Figure 1: The loss difference and gradient difference between our methods and baseline.

C Analyze the Training Loss and Gradient

Optimizer Gradient. As shown in Figure 1, comparing the gradients of $softmax(z)$ and $z \cdot softmax(z)$, the latter shows larger gradients initially due to the multiplicative factor of z. A detailed analysis in the methodology section confirms this behavior. In contrast, the normalized variant, $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$, produces gradients similar to or smaller than $softmax(z)$ early on but can grow larger later in training. This is due to normalization, which stabilizes updates but reduces gradient magnitude in the early stages.

Training Loss Across Steps. For DAPEV2-Kerple, $z \cdot softmax(z)$ and $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ can both achieve lower loss than baseline $softmax(z)$. However, $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ is better than baseline $softmax(z)$ through the whole training steps, while the $z \cdot softmax(z)$ achieves better performance than baseline at late training step. This may caused by that the $\frac{(z-\min(z_{\min},0))}{\max(0,z_{\max})-\min(z_{\min},0)} \cdot softmax(z)$ is a normalized version so that easier for optimizer.

D Risk

This work focuses on utilizing self-adjust softmax to improve the transformer architecture. This is no specific risk. Also, we use AI assistants for writing.

E Time Cost

Table 10: The perplexity on Books3 dataset with three random seeds.

Model	**Cost**	Softmax	SA-Softmax
125M	Time Cost (ms)	143.99	160.46
125M	Memory Cost (GB)	2.67	2.67
350M	Time Cost (ms)	284.92	342.47
350M	Memory Cost (GB)	6.65	6.65
1.3B	Time Cost (ms)	473.92	509.28
1.3B	Memory Cost (GB)	16.19	16.19

F Visualization of Attention Score

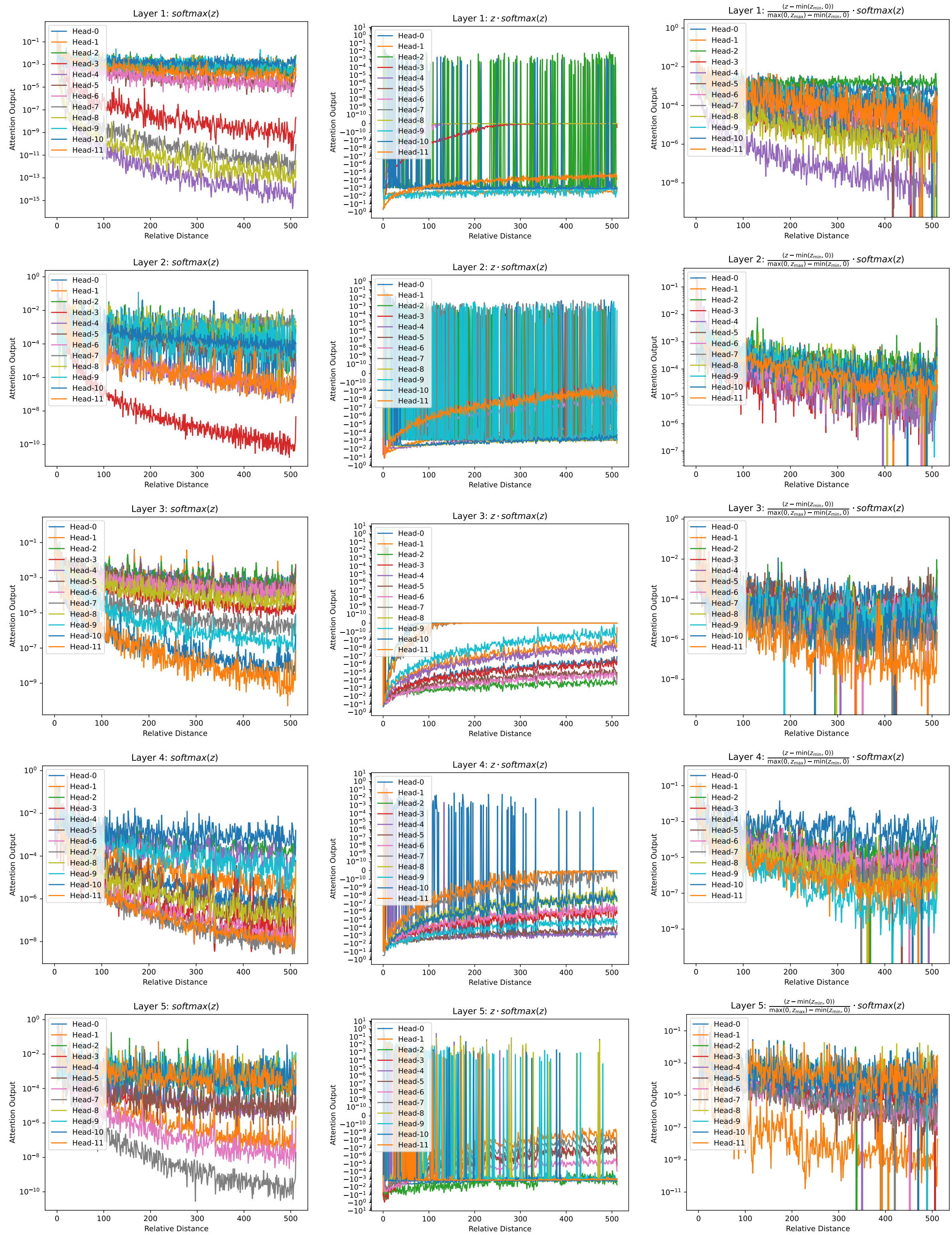

Figure 2: **The visualization of attention output, from left to right: 1)** $softmax(z)$**; 2)** $x * softmax(z)$**; 3)** $\frac{(z - min(z_{\min}, 0))}{max(0, z_{max}) - min(z_{min}, 0)} \cdot softmax(z)$**.**

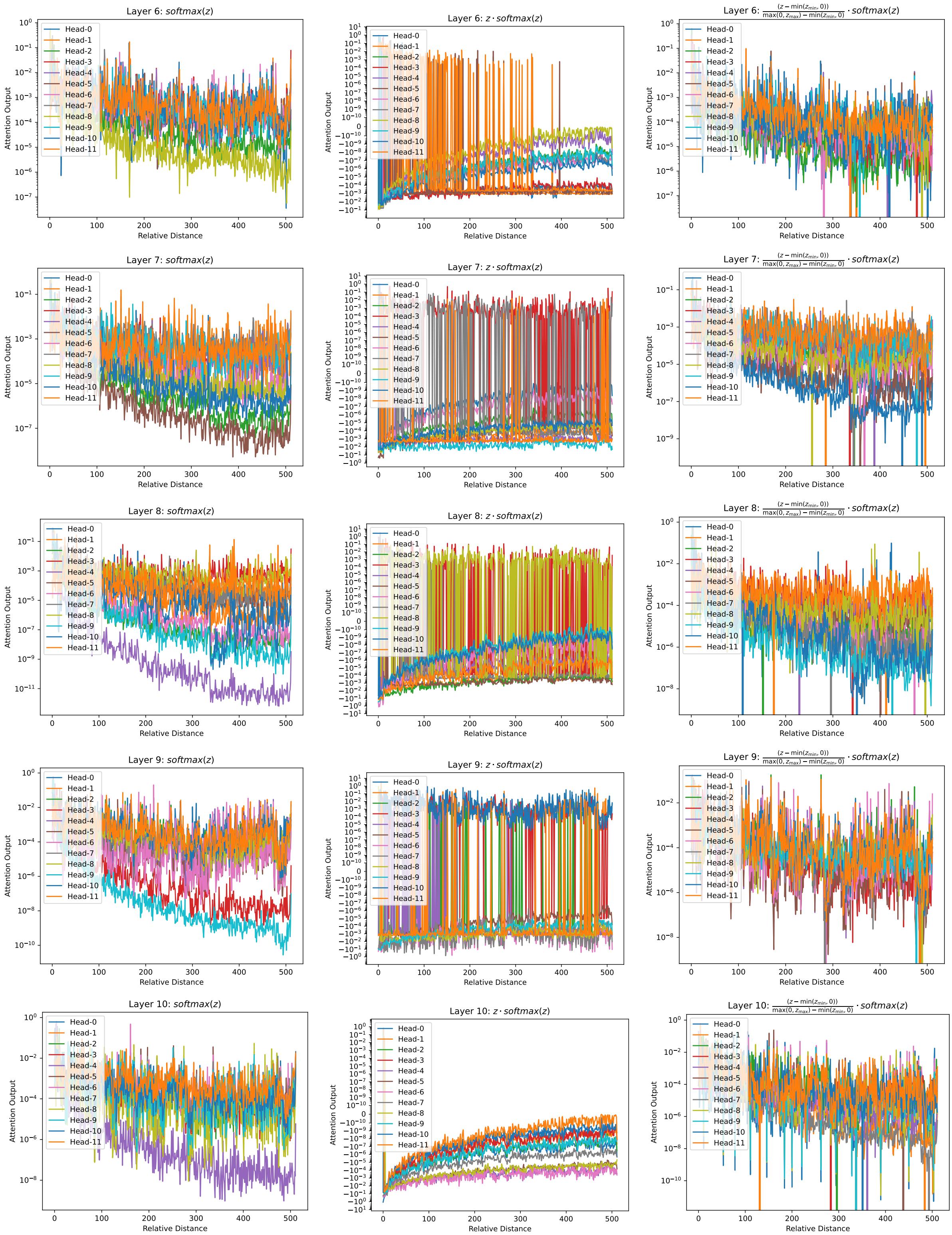

Figure 3: **The visualization of attention output, from left to right: 1) $softmax(z)$; 2) $x * softmax(z)$; 3) $\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$.**

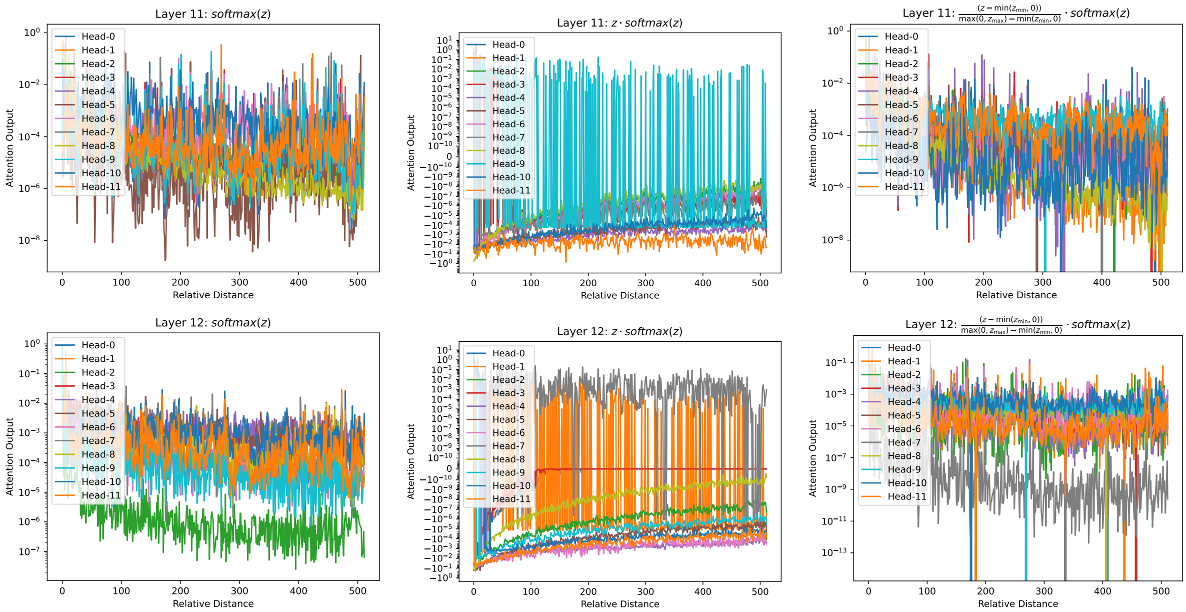

Figure 4: **The visualization of attention output, from left to right: 1)** $softmax(z)$**; 2)** $x * softmax(z)$**; 3)** $\frac{(z-min(z_{\min},0))}{max(0,z_{max})-min(z_{min},0)} \cdot softmax(z)$**.**

G Implementation

In this section, we present the implementation of the proposed SA-Softmax module in `PyTorch` which allows for research purpose (Paszke et al., 2019).

```
import torch
import torch.nn as nn

class SA-Softmax(nn.Module):
  def __init__(self, operation_name):
    """


    Args:
      operation_name: "softmax","v1","v2","v3", or "v4"
    """
    super(SA-Softmax, self).__init__()


    self.operation_name= operation_name:


  def forward(self, attention: torch.Tensor, bias: torch.Tensor):
    """
    Args:
      attention: input sequence, which is q^T * k,
         shape [bsz, num_heads, seq_len, seq_len]
      bias: bias matrix, which can be generated by ALiBi, Kerple
      FIRE or other additive position encodings
         shape [1,num_heads, seq_len, seq_len]

    Returns:
      attention with SA-Softmax,
      shape [bsz, num_heads, seq_len, seq_len]
    """
    attention_probs = softmax(attention_scores, attention_mask)

    if self.gradient_name=="v1":
        attention_probs=attention_probs*attention_scores
        attention_probs=torch.tril(attention_probs)


    elif self.gradient_name=="v2":
        B, H, T, _ = attention_scores.shape
        # Create a mask for the lower triangular part (including diagonal)
        mask = torch.tril(torch.ones(T, T, dtype=torch.bool, device=
            attention_scores.device))
        # Apply mask to get lower triangular values, replace upper triangle
            with inf (so it doesn't affect min)
        x_lower_tri = attention_scores.masked_fill(~mask, float('inf'))
        # Get the minimum value along the last dimension
        min_attention_score, _ = x_lower_tri.min(dim=-1,keepdim=True)
        attention_scores=torch.tril(attention_scores)
        attention_probs=attention_probs*(attention_scores-min_attention_score)
        attention_probs=torch.tril(attention_probs)

    elif self.gradient_name=="v3":
        B, H, T, _ = attention_scores.shape

        # Create a mask for the lower triangular part (including diagonal)
        mask = torch.tril(torch.ones(T, T, dtype=torch.bool, device=
            attention_scores.device))
        # Apply mask to get lower triangular values, replace upper triangle
            with inf (so it doesn't affect min)
        x_lower_tri = attention_scores.masked_fill(~mask, float('inf'))
        # Get the minimum value along the last dimension
        min_attention_score, _ = x_lower_tri.min(dim=-1,keepdim=True)
        # Apply mask to get lower triangular values, replace upper triangle
            with inf (so it doesn't affect min)
```

```
        x_lower_tri = attention_scores.masked_fill(~mask, float('-inf'))
        max_attention_score, _ = x_lower_tri.max(dim=-1,keepdim=True)

        attention_probs=attention_probs*((attention_scores-min_attention_score)
            /(max_attention_score-min_attention_score+1e-10))
    elif self.gradient_name=="v4":
        attention_scores_tril_this=torch.tril(attention_scores)
        min_attention_score=torch.min(attention_scores_tril_this, -1,keepdim=
            True)[0]
        max_attention_score=torch.max(attention_scores_tril_this, -1,keepdim=
            True)[0]
        min_attention_score=torch.minimum(min_attention_score, torch.zeros(1,
            device= attention_scores.device))
        max_attention_score=torch.maximum(max_attention_score,torch.zeros(1,
            device= attention_scores.device))
        attention_probs=attention_probs*((attention_scores-min_attention_score)
            /(max_attention_score-min_attention_score+1e-10))

`   attention_probs=torch.tril(attention_probs)
    return attention_probs
```