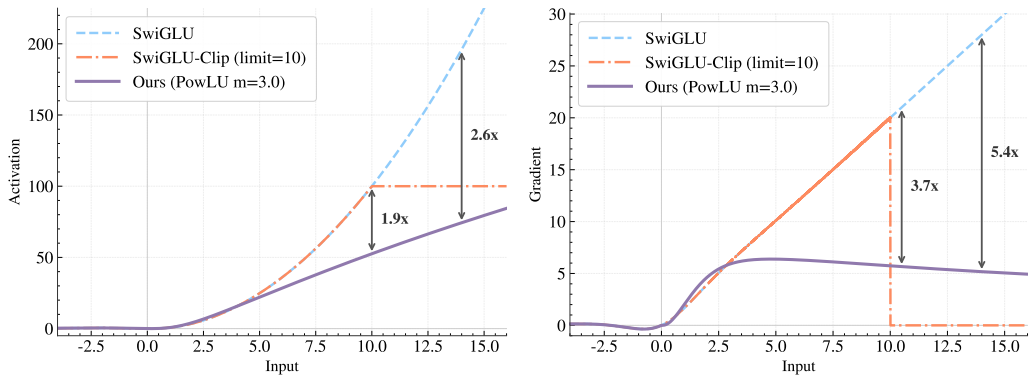
POWLU: AN ACTIVATION FUNCTION FOR STABLE PRE-TRAINING OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In contemporary large language models (LLMs), the swish-gated linear unit (SwiGLU) activation function is widely adopted to regulate the information flow and introduce non-linearity. In its scalar unary representation, SwiGLU approximates the quadratic function x^2 for large positive inputs, providing strong nonlinearity and expressive capacity. However, this property also causes numerical instability as the input or model scale increases, particularly in low-precision LLM training. The main reason is the approximate quadratic amplification reflected by its unary form, which enlarges the output range and exacerbates outliers. To address this issue, we propose a stable activation function, Power Linear Unit (PowLU), for large-scale LLM pre-training. Specifically, PowLU employs a rational power function to achieve adaptive nonlinearity, thereby improving representation ability and enabling stable training in spike regions. Moreover, we provide theoretical justification for several key properties of PowLU. Scaling law experiments confirm that the performance is consistent across model sizes, and further experimental results with the Ling-3.0 architecture (7.9B and 124B total parameters) demonstrate that PowLU achieves competitive results against SwiGLU and SwiGLU-Clip in large-scale training of LLMs. In addition, the experimental results also show that PowLU effectively improves the scalability of the large-scale training of LLMs.



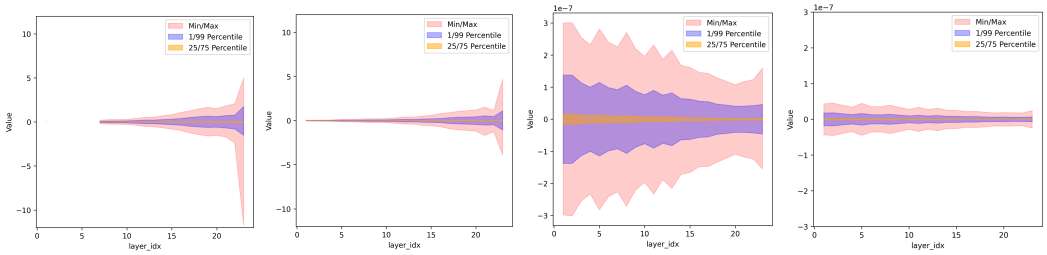
(a) Activation Functions (b) First-Order Derivatives of Functions

Figure 1: A visual comparison of the scalar unary forms of SwiGLU, SwiGLU-Clip, and PowLU in terms of function curves and first-order derivatives. Both the activation output and the first derivative of SwiGLU are significantly larger than those of PowLU. This gap continuously widens as the input increases, ultimately causing severe outliers.

1 INTRODUCTION

Large language models (LLMs) have emerged as a frontier in artificial intelligence research, demonstrating remarkable capabilities in natural language processing (Singh et al., 2025). In the architecture of LLMs, the feed-forward network layers within Transformer blocks (Vaswani et al., 2017)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) SwiGLU, Forward (b) PowLU, Forward (c) SwiGLU, Backward (d) PowLU, Backward

Figure 2: Visualization of the P99 distribution in the pre-training of 7.9B MoE LLM with SwiGLU or PowLU activation function. The results are recorded when 400B tokens are trained. The visualized layers are the linear layers in the routing experts of the LLM. In the visualizations, the red band spans from the minimum to the maximum values to show the full range including extreme outliers. The purple band spans from P1 to P99 to capture 98% of the distribution and indicate the effective dynamic range while excluding rare extremes. In addition, the orange band spans from P25 to P75 to represent the central 50% of the distribution where most values concentrate.

play a critical role in determining overall performance (Mirzadeh et al., 2023; Wei et al., 2024), primarily through the non-linearity enabled by the activation function between two linear transformation layers. Current studies and architectural designs indicate that the swish-gated linear unit (SwiGLU) activation function (Shazeer, 2020) has become a prevalent choice in recent LLMs (Team et al., 2023; Grattafiori et al., 2024; Yang et al., 2025), as it can effectively enhance the model expressiveness and generalization.

In the scalar unary representation, the SwiGLU activation function is formulated as the product of the input x and $\text{SiLU}(x)$, where $\text{SiLU}(x)$ is formulated as $x \cdot \text{sigmoid}(x)$. According to the form of the SwiGLU activation function, the value of $\text{sigmoid}(x)$ approaches 1 as the input x becomes large. As a result, the output of $\text{SwiGLU}(x)$ approaches x^2. The quadratic function x^2 has a strong amplifying effect on large inputs, resulting in a large number of outliers. Figure 2 presents the percentile-based numerical distributions of forward activations and backward gradients in the linear layers of routing experts. As shown in Figure 2a, the LLM with the SwiGLU activation function exhibits a wide red band that extends to large maximum values in the forward pass. A similar phenomenon can be observed in the backward pass in Figure 2c. These observations indicate that the SwiGLU activation function introduces severe tensor outliers during pre-training. Complementary numerical distribution analyses for the linear layers in shared experts are provided in Appendix D.1 (Figure 10), where the same trend is consistently observed. This problem will become much more severe due to the accumulation effect as the number of model layers increases, thus making the pre-training process unstable and prone to collapse. Furthermore, the stability of the pre-training can be further challenged if low-precision training techniques are adopted. This is because the low-precision training is more sensitive to the range of data distribution (Chavan et al., 2024; Lee et al., 2024; Hao et al., 2025), and the amplification effect of SwiGLU can cause some values to exceed the range of the corresponding precision, e.g., FP4 and FP8. As a result, the performance and stability of the training process will be limited.

To tackle this issue, we introduce a novel activation function called Power Linear Unit (PowLU) in this work. As shown in Figure 1, PowLU suppresses the excessive positive-branch amplification caused by the gate input, reducing the unary growth from the approximately quadratic behavior of SwiGLU to an approximately linear behavior, while maintaining the non-linearity needed for the representation ability of the whole LLM. Moreover, the form of PowLU is the same as that of SwiGLU for $x \leq 0$. In PowLU, we introduce a rational power function of the square root of the input, to adaptively adjust the non-linearity level of the activation function. Furthermore, we also integrate the sigmoid function into PowLU to further enhance the degree of non-linearity. Owing to the above core designs, the rate of outliers is reduced compared with that of SwiGLU as shown in Figures 2b and 2d. Our contributions can be summarized as follows:

- We introduce a stable activation function, i.e., PowLU, which can effectively suppress gate-induced amplification when the gate input is large and ensure the non-linearity at the same

time. As a result, the shortcoming of SwiGLU regarding excessive output-range expansion can be effectively alleviated.

- We theoretically demonstrate key properties of the proposed PowLU activation function. Specifically, the continuity, differentiability, positive-branch monotonicity of the scalar unary form, and the bounded positive-gate property of the practical gated form are theoretically justified, thus supporting the rationale behind the design of PowLU.
- We have conducted scaling law experiments to demonstrate the consistent validity of PowLU. Furthermore, we also validate the effectiveness of PowLU in the large-scale pre-training for LLMs with 7.9 billion (B) and 124B parameters. The experimental results demonstrate that PowLU achieves competitive results compared with the widely used SwiGLU or SwiGLU-Clip activation function. Furthermore, the experimental results also show that PowLU can maintain the stability of the pre-training while reducing loss spikes.

2 Related Works

2.1 Activation Functions

Activation functions are essential for introducing non-linearity into neural networks (Kunc & Kléma, 2024) and are a key factor in the representation capability of Transformer-based LLMs (Minaee et al., 2024). Early Transformer architectures adopted the rectified linear unit (ReLU) (Vaswani et al., 2017; Nair & Hinton, 2010), which is simple and computationally efficient but has limited smoothness. GeLU (Hendrycks & Gimpel, 2016) was later introduced as a smoother alternative and became widely used in BERT (Devlin et al., 2019) and GPT-style models (Radford et al., 2018). Despite their effectiveness, these point-wise activations process each neuron independently and do not explicitly model multiplicative interactions between hidden features (Bai, 2022). To improve expressiveness, gated activation functions have been introduced into neural networks and LLMs. The gated linear unit (GLU) uses a gating mechanism to adaptively modulate information flow (Dauphin et al., 2017; Ramachandran et al., 2017). Building on this design, SwiGLU replaces the sigmoid gate with SiLU and has become a prevalent activation function in modern LLMs because of its strong empirical performance and generalization ability (Shazeer, 2020). However, in its scalar unary representation, the positive branch of SwiGLU grows approximately as a quadratic function for large inputs. This strong amplification can enlarge activation ranges and aggravate tensor outliers, especially in large-scale and low-precision training. In contrast, PowLU is designed to suppress excessive positive-branch growth while retaining non-linearity, thereby improving numerical stability without sacrificing expressive capacity.

2.2 Stable Training of LLMs

Training stability is a critical issue in the pre-training of LLMs, particularly as model size, training length, and numerical efficiency requirements continue to increase (Takase et al., 2025; Wortsman et al., 2024). Instability commonly appears in the form of loss spikes or gradient spikes, which may degrade final performance or even lead to training collapse (Molybog et al., 2023; Zhang et al., 2024). Existing approaches address this problem from several directions, including optimization, architecture design, and activation function design. From the optimization perspective, gradient clipping is a standard technique for preventing gradient explosions and stabilizing training dynamics (Pascanu et al., 2013). More recent spike-aware Adam variants further improve robustness by detecting unstable updates and resetting or adjusting optimizer states (Huang et al., 2025b;a). From the architectural perspective, residual connections and layer normalization have become fundamental components of Transformer-based LLMs for stabilizing signal and gradient propagation (Vaswani et al., 2017). Activation function design provides another direct way to improve stability by controlling the magnitude of intermediate representations. SwiGLU-Clip (Agarwal et al., 2025) improves stability by clamping the linear component and capping the gate component, effectively reducing activation outliers. However, hard clipping may discard useful information once activations exceed predefined thresholds. SiTU-GLU (Team et al., 2026) instead applies smoother caps to the gated branches, but its saturating transformation may still over-compress large activations and weaken informative variations. PowLU differs from these methods by gradually suppressing activation growth through a rational power form, providing a smoother trade-off between outlier control and information preservation.

3 POWLU ACTIVATION FUNCTION

3.1 FORMULATION

Unary Function Expression. We first introduce the scalar unary representation of PowLU, denoted by $f_m(x)$. This form is obtained by tying the two multiplicative inputs of the gated unit, i.e., setting $x_1 = x_2 = x$, and is used for visualization and theoretical analysis throughout this paper. The unary form is given by

$$f_m(x) = \begin{cases} x \cdot x^{\frac{m}{\sqrt{x}+1}} \cdot \text{sigmoid}(x), & x > 0, \\ x^2 \cdot \text{sigmoid}(x), & x \leq 0, \end{cases} \tag{1}$$

where x denotes the scalar input, m is a hyperparameter satisfying $0 < m < 10$, and $\text{sigmoid}(x) = 1/(1+e^{-x})$ denotes the sigmoid function. For $x \leq 0$, this unary expression is identical to the scalar unary form of SwiGLU.

Linear Unit Form. In practical implementations, PowLU is used as a gated linear unit with two projected inputs, while Eq. (1) serves as its scalar unary representation for analysis. Specifically, the same input $\boldsymbol{h}$ is mapped by two linear projections with independent parameters: $\boldsymbol{x}_1 = \text{Linear}_1(\boldsymbol{h})$ and $\boldsymbol{x}_2 = \text{Linear}_2(\boldsymbol{h})$. The Linear Unit form of PowLU is then defined as

$$\text{PowLU}(\boldsymbol{h}) = \boldsymbol{x}_1 \odot \begin{cases} \boldsymbol{x}_2^{\frac{m}{\sqrt{\boldsymbol{x}_2}+1}} \odot \text{sigmoid}(\boldsymbol{x}_2), & \boldsymbol{x}_2 > 0, \\ \boldsymbol{x}_2 \odot \text{sigmoid}(\boldsymbol{x}_2), & \boldsymbol{x}_2 \leq 0, \end{cases} \tag{2}$$

where Linear_1 and Linear_2 have independent learnable parameters. All operations and conditions in Eq. (2) are applied element-wise, and $\odot$ denotes element-wise multiplication. The reference implementation is provided in Appendix A.

Motivations. In the case of $x > 0$, $\sqrt{x}$ is included to slow down the rate at which the activation function degenerates into a linear form as x increases. To be more specific, in the unary form, when x is increasing, $m/(\sqrt{x}+1)$ approaches 0 and $\text{sigmoid}(x)$ approaches 1, and the PowLU function is approximately equal to x. In this case, $\sqrt{x}$ can decrease the speed at which $m/(\sqrt{x}+1)$ approaches 0, thus reducing the speed at which the unary function degenerates to x. Meanwhile, $\sqrt{x}$ can further enhance the non-linearity of the PowLU activation function compared with x.

In addition, the constant 1 in the denominator is introduced to improve numerical stability in computation. With this design, the exponent $m/(\sqrt{x}+1)$ remains finite near the origin and approaches m as $x \to 0^+$. If the denominator were $\sqrt{x}$ alone, the exponent $m/\sqrt{x}$ would diverge for tiny positive inputs, and it can make the power operation numerically ill-conditioned, leading to underflow or unstable backward computation in training. Therefore, the added constant prevents the exponent from becoming arbitrarily large near zero and makes the computation more robust.

3.2 THEORETICAL ANALYSIS

Unless otherwise stated, the following theoretical analysis is conducted on the scalar unary representation $f_m(x)$ defined in Eq. (1). This unary form is used to study the point-wise shape properties of PowLU, including continuity, differentiability, and positive-branch monotonicity. For the stability-related boundedness property, we analyze the positive-branch gate in the practical gated form, rather than the full unary product $f_m(x)$. Since the practical linear unit form in Eq. (2) applies the nonlinear gate element-wise, these analyses characterize the corresponding scalar nonlinear transformation rather than the full multivariate mapping from $\boldsymbol{h}$ to $\text{PowLU}(\boldsymbol{h})$.

Continuity. The unary PowLU function f_m is composed of elementary functions when $x \neq 0$. Therefore, it is continuous in this case. When $x = 0$, the function value $f_m(0) = 0$. Meanwhile, the left-hand limit $\lim_{x\to 0^-} f_m(x) = 0^2 \cdot 1/(1+e^0) = 0$. Moreover, the right-hand limit can be calculated as $\lim_{x\to 0^+} 0^{1+m} \cdot 1/2$. Because the hyperparameter m satisfies $m > 0$, the right-hand limit is 0. Since the left-hand limit, the function value, and the right-hand limit are all equal to 0, the unary PowLU function f_m is continuous at $x = 0$.

Differentiability. Because the unary PowLU function f_m is composed of differentiable elementary functions when $x \neq 0$, it is also differentiable in these cases. When $x = 0$, we analyze the left-hand and right-hand derivatives to show differentiability. Specifically, the left-hand derivative

can be written as $f'_{m,-}(0) = \lim_{h\to 0^-}(f_m(h) - f_m(0))/h = \lim_{h\to 0^-} h \cdot \text{sigmoid}(h) = 0$. Furthermore, the right-hand derivative can be written as $f'_{m,+}(0) = \lim_{h\to 0^+}(f_m(h) - f_m(0))/h = \lim_{h\to 0^+} h^{m/(\sqrt{h}+1)} \cdot \text{sigmoid}(h)$. Because $m > 0$, $f'_{m,+}(0)$ is equal to 0. Since the left-hand derivative is equal to the right-hand derivative, the unary PowLU function f_m is differentiable at $x = 0$.

Positive-Branch Monotonicity. In the case of $x > 0$, the positive branch of the unary PowLU function f_m is monotonically increasing when the hyperparameter m satisfies $0 < m < 10$. The detailed analysis can be found in Appendix B.1.

Bounded Positive Gate. The scalar unary positive branch can be written as $f_m(x) = xq_m(x)$, where $q_m(x) = x^{m/(\sqrt{x}+1)}\sigma(x)$ for $x > 0$. The unary function f_m itself is not bounded in the positive direction. Indeed, $f_m(x) \sim x$ as $x \to +\infty$. The bounded component is the positive-branch gate q_m. Specifically, for any $m > 0$, we have $0 < q_m(x) \le C_m$, where $C_m = \exp\big(2mW(e^{-1})\big)$ and W denotes the Lambert W function. As a concrete example, setting $m = 3$ gives $C_m \approx 5.32$. Therefore, the practical PowLU satisfies $|\,\text{PowLU}(x_1, x_2)| \le C_m|x_1|$ for $x_2 > 0$, which means that the gate input x_2 cannot cause unbounded amplification. In contrast, the SwiGLU gate satisfies $x_2\sigma(x_2) \sim x_2$ as $x_2 \to +\infty$, so no constant independent of x_2 can provide such a bound. The proof is provided in Appendix B.2.

4 Experiments

4.1 Experimental Setup

Training Settings. To evaluate the performance of different activation functions (i.e., SwiGLU (Shazeer, 2020), SwiGLU-Clip (Agarwal et al., 2025; Xu et al., 2026), and PowLU), we place them between two linear layers in routing experts and shared experts in Mixture-of-Experts (MoE) models following the Ling architecture (inclusionAI, 2026b;a). We adopt a series of LLMs with a small number of parameters for scaling law experiments. Meanwhile, we have also trained larger models with 7.9B (Ling-3.0-tiny (inclusionAI, 2026b)) and 124B (Ling-3.0-flash (inclusionAI, 2026a)) parameters for a large number of GPU hours. Detailed model architecture and training hyperparameters are provided in Appendix C. Furthermore, we use $m = 3.0$ in the main experiments. The corresponding multi-scale parameter study is presented in Section 4.4.1.

Evaluation Benchmarks. Following conventions (Team et al., 2025), we choose evaluation benchmarks from three categories:

World Knowledge: AGIEval (Zhong et al., 2024), MMLU (Hendrycks et al., 2021a), MMLU-Pro (Wang et al., 2024), MMMLU (OpenAI, 2024), C-Eval (Huang et al., 2023), CMMLU (Li et al., 2024), SuperGPQA (Du et al., 2025), TriviaQA (Joshi et al., 2017), ARC-challenge (Clark et al., 2018).

Language & Reasoning: BBH (Suzgun et al., 2023), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021).

Math & Code: HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), MGSM (Shi et al., 2023), CMATH (Wei et al., 2023).

4.2 Main Results

4.2.1 Scaling Law Experiments

We have conducted scaling law experiments on a series of MoE LLMs. The details of LLMs and training settings are presented in Table 1. As shown in Figure 3, the loss scaling curves of SwiGLU and PowLU almost overlap, indicating that SwiGLU and PowLU have roughly the same performance on MoE LLMs with 26 million (M) to 368M activated parameters.

4.2.2 Results on 7.9B and 124B Models

We evaluate PowLU on LLMs of two different scales. For the 7.9B-parameter setting, the LLMs are pre-trained with 600B tokens and integrated with SwiGLU, SwiGLU-Clip, or PowLU activation

Table 1: Model configurations and hyperparameters for scaling law experiments. "Act. Params." refers to the number of activated parameters in MoE models. All models are trained with a sequence length of 4,096.

Act. Params.	Layer	Hidden	lr	Batch Size
26M	10	512	0.00156	128
47M	12	640	0.00130	192
92M	16	768	0.00110	256
199M	20	1024	0.00091	448
368M	24	1280	0.00077	640

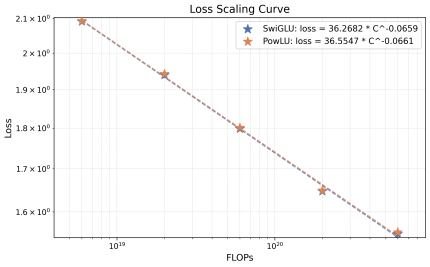


Figure 3: The fitted scaling law curves for SwiGLU and PowLU.

Table 2: Comparisons among SwiGLU, SwiGLU-Clip, and PowLU activation functions on Ling-3.0-tiny and Ling-3.0-flash. Ling-3.0-tiny is an MoE model with 7.9B total parameters and is pre-trained with 600B tokens, while Ling-3.0-flash is an MoE model with 124B total parameters and is pre-trained with 800B tokens. For Ling-3.0-tiny, the highest results are in bold font and the second-highest results are underlined. For Ling-3.0-flash, the highest results are in bold font.

Types	Benchmarks	Ling-3.0-tiny (7.9B)			Ling-3.0-flash (124B)	
		SwiGLU	SwiGLU-Clip	PowLU	SwiGLU	PowLU
	Trained Tokens	600B	600B	600B	800B	800B
World Knowledge	AGIEval	**31.75**	30.23	31.13	53.03	**53.75**
	MMLU	53.95	54.12	**54.92**	69.10	**69.14**
	MMLU-Pro	21.56	21.79	**24.00**	**40.75**	40.12
	MMMLU	31.77	32.40	**32.61**	46.27	**48.10**
	C-Eval	51.99	52.62	**52.96**	**71.74**	71.56
	CMMLU	52.69	50.24	**52.82**	**71.58**	71.41
	SuperGPQA	**17.67**	17.14	17.02	25.93	**26.16**
	TriviaQA	47.86	**48.87**	48.18	66.31	**66.91**
	ARC-challenge	51.53	51.86	**55.93**	77.29	**83.05**
Language & Reasoning	BBH	38.82	38.22	**38.96**	62.07	**63.36**
	HellaSwag	66.24	66.31	**66.46**	76.23	**76.24**
	WinoGrande	63.14	64.56	**65.90**	**75.45**	73.72
Math & Code	HumanEval	25.61	23.17	**26.83**	54.27	**55.49**
	GSM8K	30.40	32.30	**33.74**	**70.81**	69.90
	MATH	22.98	22.64	**23.98**	42.22	**44.98**
	MGSM	15.93	17.40	**18.40**	54.00	**54.80**
	CMATH	63.21	**63.39**	63.11	80.69	**83.33**

functions. For the larger-scale 124B-parameter setting, the LLMs are pre-trained with 800B tokens and integrated with either SwiGLU or PowLU activation functions. The experimental results are presented in Table 2.

For the 7.9B model, PowLU achieves competitive performance across the 17 selected evaluation benchmarks. More specifically, PowLU enhances training stability while retaining desirable non-linear transformation properties, thereby effectively preserving the model's expressive capacity. For the 124B model, the PowLU-based LLM also delivers competitive overall performance. These results demonstrate that integrating PowLU can effectively maintain the performance of LLMs at different scales while enhancing training stability, further confirming the effectiveness and scalability of PowLU.

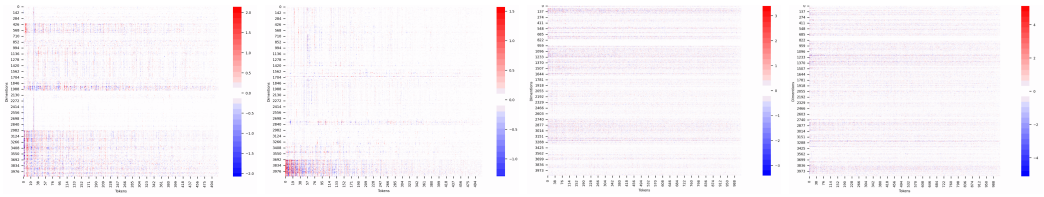
4.3 Stability Evaluation

4.3.1 Loss Spike Analysis

In the training processes of larger-scale LLMs with SwiGLU or SwiGLU-Clip activation functions, we have faced the loss spike problem which makes the training process unstable. In contrast, the



Figure 4: Visualization of the training loss curves of large-scale LLMs with SwiGLU, SwiGLU-Clip, or PowLU activation functions. **Blue Line:** SwiGLU-based LLM trained with BF16 precision. **Orange Line:** SwiGLU-based LLM trained with FP8 precision. **Green Line:** SwiGLU-Clip-based LLM trained with FP8 precision. **Red Line:** PowLU-based LLM trained with FP8 precision.



(a) SwiGLU, Forward (b) PowLU, Forward (c) SwiGLU, Backward (d) PowLU, Backward

Figure 5: Visualization of the outlier channels in the low-precision training of 7.9B MoE LLMs with SwiGLU activation function or PowLU activation function. The visualized layers are the linear layers in the routing experts of the LLM.

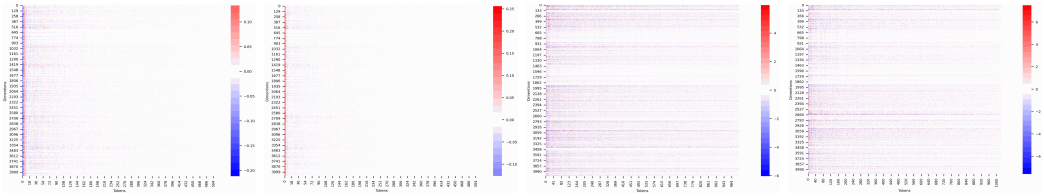
proposed PowLU activation function shows improved stability in this case. In this part, we have visualized the loss curves of the above training processes in Figure 4 for further analysis.

In our experiments, loss spikes mainly appear after 76,200 training steps. As shown in Figure 4, the visualized loss curves demonstrate a divergence in stability. Specifically, the SwiGLU-based LLM trained with BF16 precision (blue line) achieves relatively low loss values and stable training process. This phenomenon mainly comes from two aspects. First, the training precision of this experiment (BF16) is higher than that of other experiments (FP8), thus the loss values of this experiment are lower than those of others. Second, the experiments for SwiGLU-Clip and PowLU are conducted via replacing the SwiGLU activation function after 75,000 training steps. As a result, there is a loss recovery process and the loss values are higher. Moreover, the SwiGLU-Clip activation function (green line) delays the timing of loss spikes compared to the SwiGLU baseline with FP8 training (orange line), but it also suffers from the loss spike around training step 77,000. In addition, the proposed PowLU activation function (red line) maintains a consistently smooth and low loss trajectory, hovering around 1.32 without significant deviations. The above empirical observations indicate that the proposed PowLU activation function effectively mitigates the occurrence of loss spikes, enhancing the training stability of large-scale LLMs compared to commonly used SwiGLU and SwiGLU-Clip activation functions.

4.3.2 Outlier Channels Analysis

To further characterize tensor outliers in routing experts and shared experts, we visualize their outlier channels (i.e., hidden dimensions). We consider the same tensor types as those used in the percentile-based numerical distribution analyses in Figure 2 and Appendix D.1. For each tensor, we compute the L_2 norm along the channel axis and sort all channels in descending order according to their norm magnitudes to facilitate visual inspection. The results are presented in Figures 5 and 6.

The visualizations in Figure 5 demonstrate the efficacy of the proposed PowLU activation function in suppressing outlier channels compared to the standard SwiGLU activation function. For the



(a) SwiGLU, Forward (b) PowLU, Forward (c) SwiGLU, Backward (d) PowLU, Backward

Figure 6: Visualization of the outlier channels in the low-precision training of 7.9B MoE LLMs with SwiGLU activation function or PowLU activation function. The visualized layers are the linear layers in the shared experts of the LLM.

visualizations of SwiGLU in Figures 5a and 5c, there is a prevalence of high-intensity red and blue regions, indicating that specific channels exhibit large magnitudes during both forward and backward propagation. In contrast, the corresponding visualizations for PowLU display a more uniform color distribution along with a compressed dynamic range. Moreover, we can make similar observations from Figure 6. The analysis above indicates that PowLU effectively mitigates the extreme outliers, resulting in smoother tensor distributions which are inherently more stable for the large-scale LLM training process.

4.4 Ablation Studies

4.4.1 Parameter Studies

The hyperparameter m controls the degree of non-linearity on the positive branch of PowLU. To examine its sensitivity across model scales, we sweep $m \in \{2, 2.5, 3, 3.5\}$ on MoE models with 26M to 368M activated parameters. Figure 7 summarizes both the observed grid minima and the continuous optima estimated from quadratic fits.

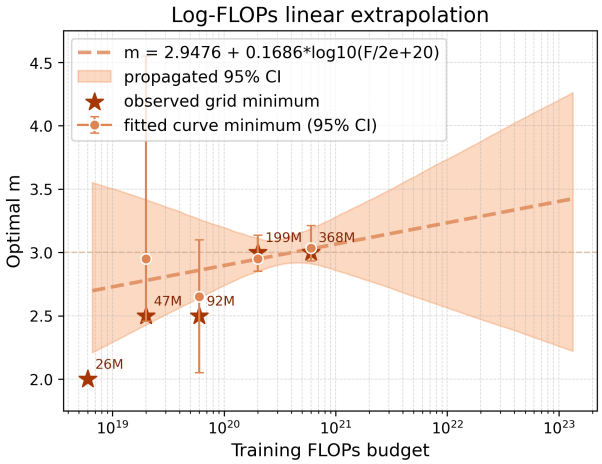
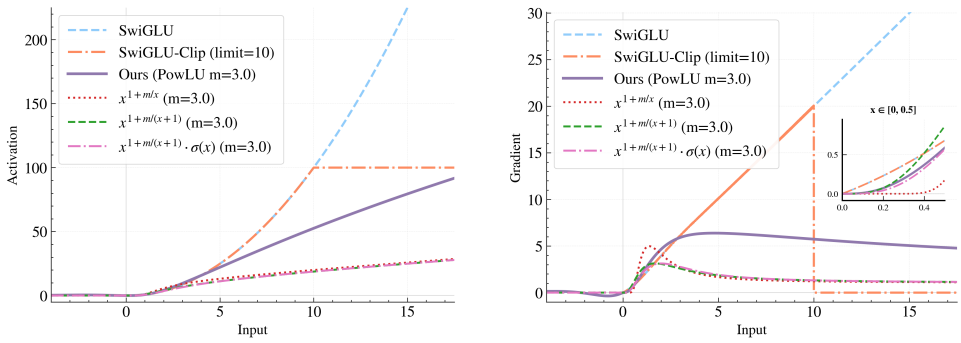


Figure 7: Multi-scale selection of the PowLU hyperparameter m as a function of training FLOPs.

The preferred value exhibits only a mild increase with training FLOPs. The observed grid optimum is $m = 2.5$ for the 47M and 92M models and shifts to $m = 3$ for the 199M and 368M models. Moreover, the loss range across all tested values of m is at most 0.005 at every scale, indicating that PowLU is not sensitive to moderate changes in this hyperparameter. Considering its performance at the larger observed scales and the uncertainty of the extrapolated trend, we use $m = 3.0$ as a robust default in the main experiments. The complete raw results and fitting details are provided in Appendix D.2.

4.4.2 Ablation Studies for Core Components

In order to validate the appropriateness of the activation function design, we conduct ablation studies in terms of the core components in PowLU. Specifically, the ablation study mainly focuses on the term 1 of the denominator in the exponent, the sigmoid function, and the term $\sqrt{x}$ of the denominator in the exponent. To simulate different activation functions in the case of $x > 0$, we design three additional activation functions, i.e., $x^{1+m/x}$, $x^{1+m/(x+1)}$, and $x^{1+m/(x+1)} \cdot \text{sigmoid}(x)$. In the



(a) Activation Functions (b) First-Order Derivatives of Functions

Figure 8: Visualizations of curves and first-order derivatives for different activation functions.

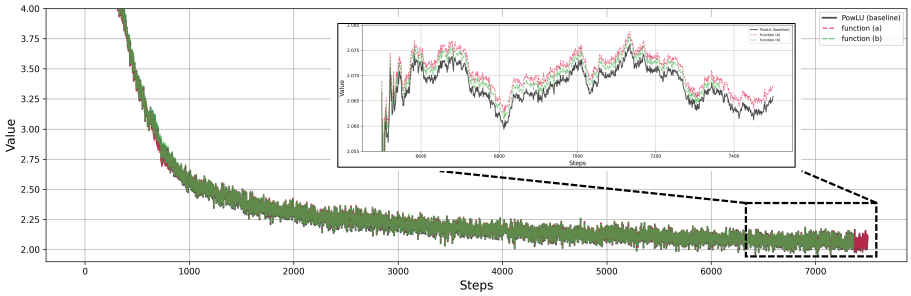


Figure 9: Visualizations for the loss difference between different activation functions and SwiGLU.

$x \leq 0$ part, the formulation of these activation functions is the same as SwiGLU. The function curves and the corresponding first-order derivatives are presented in Figure 8.

As can be observed from Figure 8a, the growth rate of PowLU is between that of SwiGLU and the three functions designed when x becomes larger. Moreover, the three functions designed have similar growth trends. As for the first-order derivatives in Figure 8b, the derivatives of PowLU and three functions designed tend to plateau after rising. Meanwhile, the derivatives of PowLU are larger than that of the three designed functions. In addition, the subfigure on the right shows that the gradient of function $x^{1+m/x}$ is almost invisible in interval $x \in (0, 0.5]$. This is mainly because the term m/x approaches $+\infty$ as $x \to 0^+$. To this end, we introduce the term 1 of the denominator in the exponent to avoid the above problem of numerical instability.

In addition, the training loss curves in terms of different activation functions are presented in Figure 9. Please note that the function (a) and function (b) in this figure refer to activation functions $x^{1+m/(x+1)}$ and $x^{1+m/(x+1)} \cdot \text{sigmoid}(x)$, respectively. As can be seen from Figure 9, the proposed PowLU activation function achieves the lowest loss value compared to other two activation functions. These results support the validity of introducing terms $\sqrt{x}$ and $\text{sigmoid}(x)$ in the formulation of PowLU.

5 Conclusion

In this work, we identify the numerical instability caused by the commonly-used SwiGLU activation function in the LLM training. To tackle this issue, we propose the PowLU activation function as a solution. PowLU bounds the positive-branch gate via a rational power function, thereby suppressing gate-induced amplification and effectively mitigating outliers. Experimental results demonstrate that the PowLU-based LLM achieves competitive performance against the SwiGLU-based or SwiGLU-Clip-based LLMs across knowledge, reasoning, and code benchmarks. In addition, the analysis and ablation studies show that PowLU can effectively enhance the training stability of LLMs.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs and providing critical ingredients for proving mathematical claims. We have not used generative AI tools for implementing methods, generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, and interpreting results. We have reviewed all AI-assisted work. In particular, all AI-assisted mathematical arguments were independently verified, and all AI-assisted code was reviewed and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Yuhan Bai. ReLU-function and derived function review. In *SHS Web of Conferences*, volume 144, pp. 02006, 2022.

Arnav Chavan, Raghav Magazine, Shubham Kushwaha, Mérouane Debbah, and Deepak Gupta. Faster and lighter LLMs: A survey on current challenges and way forward. *arXiv preprint arXiv:2402.01799*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pp. 933–941. PMLR, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

Xeron Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *Advances in Neural Information Processing Systems*, 38, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhiwei Hao, Jianyuan Guo, Li Shen, Yong Luo, Han Hu, Guoxia Wang, Dianhai Yu, Yonggang Wen, and Dacheng Tao. Low-precision training of large language models: Methods, challenges, and opportunities. *arXiv preprint arXiv:2505.01043*, 2025.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

Tianjin Huang, Haotian Hu, Zhenyu Zhang, Gaojie Jin, Xiang Li, Li Shen, Tianlong Chen, Lu Liu, Qingsong Wen, Zhangyang Wang, and Shiwei Liu. Stable-SPAM: How to train in 4-bit more stably than 16-bit Adam. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025a. URL https://openreview.net/forum?id=Gk2pBIAMUl.

Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. SPAM: Spike-aware Adam with momentum reset for stable LLM training. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=L9eBxTCpQG.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36: 62991–63010, 2023.

inclusionAI. Ling-3.0-flash. https://huggingface.co/inclusionAI/Ling-3.0-flash, 2026a.

inclusionAI. Ling-3.0-tiny. https://huggingface.co/inclusionAI/Ling-3.0-tiny, 2026b.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Vladimír Kunc and Jiří Kléma. Three decades of activations: A comprehensive survey of 400 activation functions for neural networks. *arXiv preprint arXiv:2402.09092*, 2024.

Joonhyung Lee, Jeongin Bae, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. To FP8 and back again: Quantifying the effects of reducing precision on LLM training stability. *arXiv preprint arXiv:2405.18710*, 2024.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11260–11285, 2024.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on Adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

OpenAI. Multilingual massive multitask language understanding (MMMLU). https://huggingface.co/datasets/openai/MMMLU, 2024.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. https://www.mikecaptain.com/resources/pdf/GPT-1.pdf, 2018.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=fR3wGCk-IXp.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=52YBEzcI0l.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Ling Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation. *arXiv preprint arXiv:2510.22115*, 2025.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. CMATH: Can your language model pass chinese elementary school math test? *arXiv preprint arXiv:2306.16636*, 2023.

Xiuying Wei, Skander Moalla, Razvan Pascanu, and Caglar Gulcehre. Building on efficient foundations: Effective training of LLMs with structured feedforward layers. *Advances in Neural Information Processing Systems*, 37:4689–4717, 2024.

Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie E Everett, Alexander A Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-Dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=d8w0pmvXbZ.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5HCnKDeTws.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, 2024.

APPENDIX

A PYTORCH IMPLEMENTATION OF POWLU

Listing 1 provides a reference implementation of PowLU in PyTorch. Following the implementation used in our experiments, x_1 and x_2 denote two linear projections of the same input. The activation is computed as $\mathrm{PowLU}(x_1, x_2) = x_1 \cdot f(x_2)$, where $f(x_2) = x_2^{m/(\sqrt{x_2}+1)}\sigma(x_2)$ for $x_2 > 0$, and $f(x_2) = x_2\sigma(x_2)$ otherwise. Setting $x_1 = x_2 = x$ recovers the single-variable formulation in Eq. (1). This recovered single-variable formulation is the scalar unary form used in our theoretical analysis.

For numerical safety, the implementation replaces non-positive values by one only when evaluating the inactive positive branch. This avoids applying the square-root and power operations to invalid inputs without changing the output selected by `torch.where`.

```python
import torch
import torch.nn as nn


class PowLU(nn.Module):
    """Power Linear Unit used in a gated feed-forward network.

    Args:
        m: Power coefficient. The paper uses m = 3.0.
           The monotonicity analysis assumes 0 < m < 10.
    """

    def __init__(self, m: float = 3.0):
        super().__init__()
        if not 0.0 < m < 10.0:
            raise ValueError("PowLU requires 0 < m < 10.")
        self.m = float(m)

    def forward(
        self,
        x1: torch.Tensor,
        x2: torch.Tensor,
    ) -> torch.Tensor:
        # x1 and x2 are two projections of the same input.
        positive = x2 > 0

        # The positive branch contains sqrt(x2) and x2 ** exponent.
        # Replacing inactive values by one prevents invalid operations
        # for x2 <= 0, while leaving the selected result unchanged.
        safe_x2 = torch.where(
            positive,
            x2,
            torch.ones_like(x2),
        )

        exponent = self.m / (torch.sqrt(safe_x2) + 1.0)
        sigmoid_x2 = torch.sigmoid(x2)

        positive_gate = torch.pow(safe_x2, exponent) * sigmoid_x2
        negative_gate = x2 * sigmoid_x2

        gate = torch.where(
            positive,
            positive_gate,
            negative_gate,
        )
        return x1 * gate
```

Listing 1: Reference PyTorch implementation of the PowLU activation function.

B Theoretical Analysis

B.1 Monotonicity

In this section, we analyze the monotonicity of the positive branch of the scalar unary PowLU function when $x > 0$. For ease of representation, we denote this positive branch as $f(x) = x^{1+m/(\sqrt{x}+1)}\sigma(x)$. In this expression, $\sigma(x) = 1/(1+e^{-x})$ denotes the sigmoid function. Because the function $f(x)$ satisfies $f(x) > 0$, analyzing the monotonicity of $f(x)$ is equivalent to analyzing the monotonicity of $g(x) = \ln f(x)$. The specific form of $g(x)$ is shown in Eq. (3).

$$g(x) = \left(1 + \frac{m}{\sqrt{x}+1}\right)\ln x + \ln\sigma(x) \tag{3}$$

Let $t = \sqrt{x} > 0$, the first-order derivative of $g(x)$ can be calculated as Eq. (4):

$$g'(x) = \frac{(t+1)^2 + m\cdot\phi(t)}{t^2(t+1)^2} + \frac{1}{1+e^{t^2}} \tag{4}$$

where $\phi(t) = t + 1 - t\ln t$.

Based on the above transformation process, we further analyze the properties of the auxiliary function $\phi(t)$. The first-order derivative of $\phi(t)$ can be written as $\phi'(t) = -\ln t$. According to the expression of $\phi'(t)$, we can know that $\phi(t)$ is monotonically increasing when $0 < t < 1$ and monotonically decreasing when $t > 1$. When $t = 1$, $\phi(t)$ achieves the global maximum value $\phi(1) = 2$. Furthermore, because $\lim_{t\to 0^+}\phi(t) = 1$, there exists a unique zero point $t_0 \approx 3.59$ of $\phi(t)$. Obviously, the function $\phi(t)$ satisfies $\phi(t) < 0$ when $t > t_0$ and $\phi(t) > 0$ when $0 < t < t_0$.

According to the above analysis, we analyze the monotonicity of $g(x)$ in the case of $m > 0$. When t satisfies $0 < t \le t_0$, function $\phi(t)$ satisfies $\phi(t) \ge 0$, thus we have Eq. (5) based on the scaling method.

$$g'(x) \ge \frac{(t+1)^2}{t^2(t+1)^2} + \frac{1}{1+e^{t^2}} = \frac{1}{t^2} + \frac{1}{1+e^{t^2}} > 0 \tag{5}$$

Furthermore, when $t > t_0$, the function $\phi(t)$ satisfies $\phi(t) < 0$. Because $1/(1+e^{t^2}) > 0$, a sufficient condition for $g'(x) > 0$ is $(t+1)^2 + m\phi(t) \ge 0$. This condition can be further transformed as shown in Eq. (6).

$$(t+1)^2 + m\phi(t) \ge 0 \Leftrightarrow m \le \frac{(t+1)^2}{t\ln t - t - 1} \equiv M(t) \tag{6}$$

Then, we find the minimum value by differentiating $M(t)$, and we can know that the stationary point of $M(t)$ satisfies $\ln t = 2 + 4/(t-1)$. The solution t^* to the transcendental equation is $t^* \approx 11.02$, thus the minimum value of $M(t)$ is approximately equal to 10.02. Therefore, the upper bound of m is approximately equal to 10.02. In summary, PowLU is monotonically increasing when the hyperparameter m satisfies $0 < m < 10$.

B.2 Bounded Positive Gate

In this section, we prove that the positive-branch gate of PowLU is bounded. Specifically, the bounded quantity is the gate $q_m(x) = x^{m/(\sqrt{x}+1)}\sigma(x)$ for $x > 0$, rather than the scalar unary positive branch $f_m(x) = xq_m(x)$. The latter satisfies $f_m(x) \sim x$ as $x \to +\infty$ and is therefore unbounded. For any $m > 0$, we show that $0 < q_m(x) \le \exp\big(2mW(e^{-1})\big)$, where W denotes the Lambert W function.

Let $t = \sqrt{x} > 0$. Then, $x^{m/(\sqrt{x}+1)} = \exp(m\ln x/(\sqrt{x}+1)) = \exp(2m\ln t/(t+1))$. Define $h(t) = 2\ln t/(t+1)$ for $t > 0$. Its derivative is $h'(t) = 2(1+1/t-\ln t)/(t+1)^2$. Since $1+1/t-\ln t$ is strictly decreasing on $(0,\infty)$, there exists a unique $t^\star > 1$ satisfying $\ln t^\star = 1 + 1/t^\star$. Therefore, $h(t)$ attains its global maximum at $t = t^\star$.

Using the stationary equation, we obtain $h(t^\star) = 2\ln t^\star/(t^\star+1) = 2/t^\star$. Moreover, $\ln t^\star = 1+1/t^\star$ implies $(1/t^\star)e^{1/t^\star} = e^{-1}$. By the definition of the Lambert W function, $1/t^\star = W(e^{-1})$, and hence $\max_{t>0} h(t) = 2W(e^{-1})$. It follows that, for every $x > 0$, $x^{m/(\sqrt{x}+1)} \le \exp\big(2mW(e^{-1})\big)$. Since $0 < \sigma(x) < 1$, we further obtain

$$0 < q_m(x) = x^{\frac{m}{\sqrt{x}+1}}\sigma(x) \le C_m, \qquad C_m = \exp\big(2mW(e^{-1})\big)\,. \tag{7}$$

This proves the boundedness of the positive-branch gate.

For the practical PowLU activation, $\mathrm{PowLU}(x_1, x_2) = x_1 q_m(x_2)$ when $x_2 > 0$. Consequently, $|\mathrm{PowLU}(x_1, x_2)| \leq C_m |x_1|$. For the setting $m = 3$ used in this paper, $2W(e^{-1}) \approx 0.55693$, and thus $C_3 = \exp\big(6W(e^{-1})\big) \approx e^{1.6708} \approx 5.32$. Therefore, increasing the gate input x_2 cannot amplify the PowLU output relative to $|x_1|$ without bound.

By contrast, the SwiGLU gate satisfies $x_2\sigma(x_2) \sim x_2$ as $x_2 \to +\infty$. Hence, there is no constant C independent of x_2 such that $|x_1 x_2 \sigma(x_2)| \leq C|x_1|$ holds for all $x_2 > 0$.

C Model Architecture and Training Hyperparameters

Table 3 summarizes the model architecture and training hyperparameters used for Ling-3.0-tiny (inclusionAI, 2026b) and Ling-3.0-flash (inclusionAI, 2026a). The table is provided for reproducibility.

Table 3: Model architecture and training hyperparameters of Ling-3.0-tiny (inclusionAI, 2026b) and Ling-3.0-flash (inclusionAI, 2026a) used in the main experiments.

Hyperparameter	**Ling-3.0-tiny**	**Ling-3.0-flash**
Total Parameters	7.9B	124B
Active Parameters	1.3B	5.1B
Layers	24	42
Routed Experts	128	512
Active Experts	8	8
Hidden Size	1536	2560
Experts Hidden Size	512	768
Number of Attention Heads	16	32
Attention Dimension	128	128
Sequence Length	8,192	8,192
Vocab Size	157,184	157,184
Batch Size	8,192	7,680
Base Learning Rate	0.000339	0.000212
Wight Decay	0.1	0.1

D Additional Visualizations and Analysis

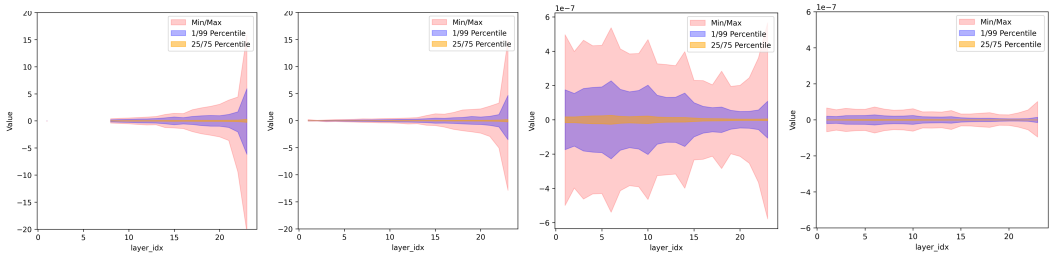
D.1 Numerical Distribution of Tensors in Shared Experts

In this section, we provide complementary numerical distribution analyses for the shared experts. These analyses complement the results for routing experts presented in Figure 2. We use percentile-based distribution plots to monitor tensor values across layers and diagnose potential training instabilities.

Specifically, we collect six percentile-based statistics for two tensor types in the linear layers of shared experts. For the forward pass, we record the input activation tensor of the second linear layer after the activation function. For the backward pass, we record the gradient tensor of the first linear layer before the activation function. The corresponding visualizations are presented in Figure 10.

As shown in Figure 10, SwiGLU exhibits a wider red band that extends to substantially higher maximum values than PowLU in the forward pass, as demonstrated by Figures 10a and 10b. This difference becomes more pronounced in the backward pass, as shown in Figures 10c and 10d. In particular, the gradients of the SwiGLU-based LLM display a highly volatile and expansive red band, indicating the presence of severe gradient outliers that may destabilize the training process.

In contrast, the PowLU-based LLM exhibits a more constrained red band and lower maximum values during backward propagation, suggesting that PowLU effectively suppresses extreme values. Overall, PowLU maintains a tighter range between the minimum and maximum values during both forward and backward propagation. These results are consistent with the observations for routing ex-



(a) SwiGLU, Forward (b) PowLU, Forward (c) SwiGLU, Backward (d) PowLU, Backward

Figure 10: Visualization of the percentile-based tensor distributions in the low-precision training of 7.9B MoE LLMs with SwiGLU or PowLU. The statistics are recorded after training on 400B tokens. The visualized layers are the linear layers in the shared experts. The red band spans from the minimum to the maximum values, the purple band spans from P1 to P99, and the orange band spans from P25 to P75.

perts in Figure 2 and further demonstrate that PowLU provides a more stable numerical distribution for large-scale low-precision pre-training.

D.2 Raw Results and Fitting Details for the PowLU Hyperparameter

Table 4 reports the raw losses used in the multi-scale hyperparameter study summarized in Figure 7. We evaluate $m \in \{2, 2.5, 3, 3.5\}$ on MoE models with 26M to 368M activated parameters. The reported training FLOPs correspond to the computational budget used at each model scale.

Table 4: Raw losses for the multi-scale sweep of the PowLU hyperparameter m. The lowest grid loss at each scale is highlighted in bold. For the 26M scale, the minimum is tied at $m = 2.0$ and $m = 3.5$. This scale is reported but excluded from continuous-optimum estimation and trend fitting because its response has tied boundary minima and is non-convex on the tested grid.

Act. Params.	**Training FLOPs**	$m = 2$	$m = 2.5$	$m = 3$	$m = 3.5$
26M	6×10^{18}	**2.073**	2.078	2.077	**2.073**
47M	2×10^{19}	1.931	**1.928**	1.930	1.929
92M	6×10^{19}	1.730	**1.729**	1.730	1.730
199M	2×10^{20}	1.641	1.640	**1.639**	1.640
368M	6×10^{20}	1.558	1.557	**1.556**	1.557

For each scale, the observed grid optimum is defined as the tested value of m with the lowest loss. For the scales with a reliable interior minimum, we additionally fit a quadratic function over the four tested values and use its vertex as the estimated continuous optimum. We then fit a linear relationship between these estimated optima and $\log_{10}(\text{FLOPs})$. The 26M scale is excluded from the latter two steps because the tested losses have tied minima at the boundaries and do not provide a reliable convex interior optimum. For visualization, the lower tied grid minimum, $m = 2.0$, is shown for this scale in Figure 7.

Because only four values of m are evaluated at each scale and the loss differences are small, the fitted trend is intended as supporting evidence rather than a definitive hyperparameter scaling law.