Stable-SPAM: How to Train in 4-Bit More Stably than 16-Bit Adam

Tianjin Huang[1,9]*, Haotian Hu[5]*, Zhenyu Zhang[6]*, Gaojie Jin[1], Xiang Li[2], Li Shen[7], Tianlong Chen[8], Lu Liu[1], Qingsong Wen[3], Zhangyang Wang[6], and Shiwei Liu[4,9]

[1]Department of Computer Science, University of Exeter, Exeter, UK
[2]Department of Computer Science, Reading University, Reading, UK
[3]AI Research Institute, Squirrel Ai Learning, Seattle, USA
[4]Mathematical Institute, University of Oxford, Oxford, UK
[5]Department of Computer Science, University of Science and Technology of China, Heifei, China
[6]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, USA
[7]The School of Cyber Science and Technology, Sun Yat-Sen University, Shenzhen, China
[8]Department of Computer Science, University of North Carolina at Chapel Hill, North Carolina, USA
[9]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, NL

Abstract

This paper comprehensively evaluates several recently proposed optimizers for 4-bit training, revealing that low-bit precision amplifies sensitivity to learning rates and often causes unstable gradient norms, leading to divergence at higher learning rates. Among these, SPAM, a recent optimizer featuring momentum reset and spike-aware gradient clipping, achieves the best performance across various bit levels, but struggles to stabilize gradient norms, requiring careful learning rate tuning. To address these limitations, we propose Stable-SPAM, which incorporates enhanced gradient normalization and clipping techniques. In particular, Stable-SPAM (1) adaptively updates the clipping threshold for spiked gradients by tracking their historical maxima; (2) normalizes the entire gradient matrix based on its historical l_2 norm statistics; and (3) inherits momentum reset from SPAM to periodically reset the first and second moments of Adam, mitigating the accumulation of spiked gradients. Extensive experiments show that Stable-SPAM effectively stabilizes gradient norms in 4-bit LLM training, delivering superior performance compared to Adam and SPAM. Notably, our 4-bit LLaMA-1B model trained with Stable-SPAM outperforms the BF16 LLaMA-1B trained with Adam by up to 2 PPL. Furthermore, when both models are trained in 4-bit, Stable-SPAM achieves the same loss as Adam while requiring only about half the training steps.

1 Introduction

Recently, several advanced optimizers have been proposed, claiming to either outperform the widely used Adam optimizer or achieve comparable performance at reduced costs in the context of Large Language Models (LLMs). Given the massive size of LLMs, reducing the memory footprint of Adam has become a key objective in this line of research (Shazeer & Stern, 2018; Chen et al., 2024; Zhang et al., 2024a; Zhao et al., 2024a; Zhang et al., 2024b; Ma et al., 2024). Another area of focus is addressing the challenges of instability in LLM training. For instance, Huang et al. (2025) proposed SPAM which incorporates momentum reset and spike-aware gradient clip (SpikeClip) to mitigate the adverse effects of loss spikes. Zhao et al. (2024b) studied the stability of various optimizers to hyperparameters with BF16. These optimizers are predominantly evaluated using the standard BF16 precision, which is a practical option for real-world LLM training (Touvron et al., 2023; Li et al., 2023). With the growing shift toward low-bit precisions such as FP8 and FP4 in LLMs due to their significant cost-saving potential (Liu et al., 2024; Lee et al., 2024; Peng et al., 2023; Xi et al., 2023), it is crucial to investigate whether their effectiveness persists under lower-bit precisions. For

[*]Equal Contribution

the newly proposed optimizers to be economical, their training with low-bit precisions should be similarly robust to hyperparameter choice as trained using higher precision.

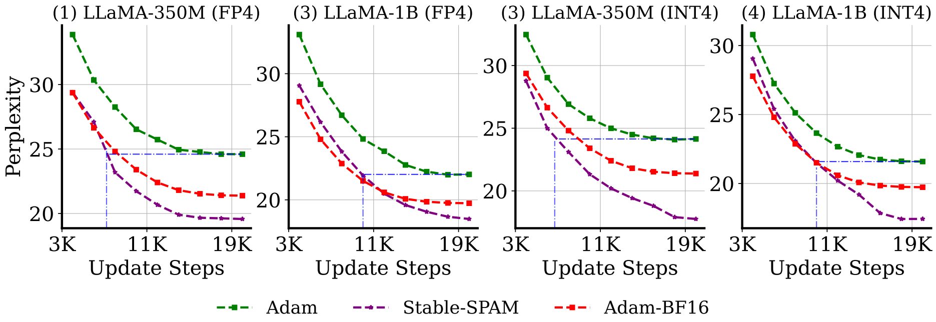

Figure 1: **Performance of 4-bit LLM training.** Experiments are conducted with LLaMA-130M/350M/1B models on C4 Dataset. **Adam-BF16** denotes that the model is trained with BF16 by Adam. Perplexity on validation set is reported.

This paper provides a comprehensive evaluation of the effectiveness and robustness of learning rate choices across various recent optimizers, including Adam (Kingma, 2014), Adafactor (Shazeer & Stern, 2018), Adam-mini (Zhang et al., 2024a), and SPAM (Huang et al., 2025), when training with 4-bit weights and activations. Our study reveals several key observations:

- ⋆ All evaluated optimizers exhibit increased sensitivity to learning rate choices during 4-bit training, often diverging quickly when larger learning rates are used as shown in Figure 2.
- ⋆ SPAM consistently achieves the lowest evaluation loss across various bit levels but requires careful learning rate tuning. Adafactor is surprisingly robust to learning rate choices, even outperforming Adam in this regard.
- ⋆ Our analysis of the training dynamics in Figure 4 reveals that 4-bit training often exhibits extremely unstable gradient norms, often accompanied by spikes, compared to BF16. This behavior can result in loss spikes and, in some cases, even training divergence with relatively larger learning rates.
- ⋆ While SpikeClip introduced in SPAM mitigates the unstable gradient norms caused by 4-bit training to a certain extent, it falls short of fully preventing training divergence, as shown in Figure 3.

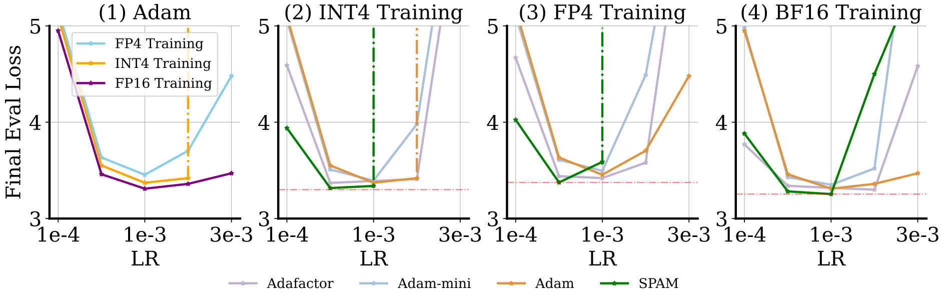

Figure 2: **Final validation loss when training LLaMA-130M on C4, sweeping across learning rates (LR).** The vertical dotted line indicates that the model cannot be trained further as increasing the learning rate, i.e. Training loss becomes NaN. Red dashed horizontal lines indicate the best performance achieved.

Despite its sensitivity to learning rate selection, SPAM consistently achieves the lowest evaluation loss across various bit levels, making it an ideal foundation for improvement. Building on this, we introduce `Stable-SPAM` to address the instability challenges associated with low-precision training of LLMs. Stable-SPAM retains the superior performance of SPAM[1] while improving stability, offering a significant advancement in low-precision optimization.

[1] Nevertheless, results in Table 3 show that our proposed techniques also improve the performance of other optimizers.

Specifically, beyond the original momentum reset operation in SPAM, `Stable-SPAM` introduces two key techniques: Adaptive Spike-Aware Clipping (`AdaClip`), which enables adaptive clipping of spiked gradients, followed by Adaptive Gradient Norm (`AdaGN`), which normalizes the entire gradient matrix based on its historical l_2 norm statistics. Our analysis demonstrates that these enhancements effectively stabilize the gradient norm of 4-bit training, achieving better performance than Adam and SPAM. Notably, our 4-bit LLaMA-1B model trained with `Stable-SPAM` outperforms the BF16 LLaMA-1B trained with Adam. Furthermore, when both models are trained in 4-bit, `Stable-SPAM` achieves the same loss as Adam while requiring only about half the training steps.

2 4-BIT TRAINING STABILITY INVESTIGATION

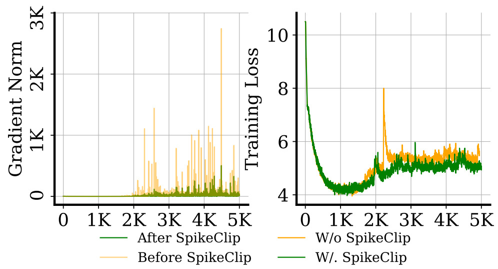

Figure 3: **Effect of SpikeClip Huang et al. (2025) on stabilizing training.** Left: gradient norms before and after performing gradient spike clip. Right: training loss with and without gradient spike clip. Models are trained by Adam optimizer based on LLaMA-130M and C4.

Recent studies (Zhao et al., 2024b; Wortsman et al., 2023b; Huang et al., 2025; Takase et al., 2023; Wortsman et al., 2023b) have investigated stability challenges in large language model (LLM) training, including issues such as learning rate instability, gradient spikes, and loss spikes. In this section, we extend the evaluation by analyzing the stability of various optimization algorithms under a 4-bit LLM training setting. Following the experimental setup outlined in (Wortsman et al., 2023b; Zhao et al., 2024b), we evaluate the final performance using a range of learning rates from 1e-4 to 3e-3. This evaluation includes two widely used optimizers, Adam (Kingma, 2014) and Adafactor (Shazeer & Stern, 2018), as well as two recently proposed methods, Adam-mini (Zhang et al., 2024a) and SPAM (Huang et al., 2025). Additionally, we monitor both the global gradient norm and training loss throughout the 4-bit LLM training process. The global gradient norm is defined as follows: $\sqrt{\sum_{i=0}^{N} \|g_i\|_2^2}$ where N is the number of layers in model and g_i denotes the gradient of i-th layer. The experiments are conducted on the LLaMA-130M/350M models using the C4 dataset and showed in Figure 2 and Figure 4. We observe:

① **Lower-bit training exhibits reduced learning rate stability.** As illustrated in Figure 2, the final evaluation loss for 4-bit training increases significantly with larger learning rates, whereas BF16 training exhibits a more stable performance across different learning rates. This indicates that 4-bit training is more sensitive and less stable in terms of learning rate.

② **Lower-bit training suffers more loss spikes and gradient norm spikes.** Figure 4 illustrates this phenomenon by comparing the training loss and gradient norm curves of LLaMA-130M and LLaMA-350M trained under BF16 and FP4 (E1M2) precision, using various learning rates. We observe that BF16 training remains stable, but FP4 training exhibits significant loss spikes, which occur on both model sizes. Furthermore, these loss spikes are consistently accompanied by gradient norm explosions.

③ **SPAM performs the best in 4-bit training but needs careful learning rate tuning.** As shown in Figure 2, SPAM achieves the lowest eval loss among various optimizers in INT4 or FP4 with the optimal learning rate. However, its validation loss either diverges to NaN or sharply increases as the learning rate rises. Additionally, we monitored the training loss and gradient norm after applying the spike clipping technique (SpikeClip) proposed in SPAM. SpikeClip detects and mitigates gradient outliers by leveraging the second moment of gradients. Specifically, it follows the expression: $g_i = \text{sign}(g_i) \cdot \sqrt{\theta V_i}$ under the condition $\frac{g_i^2}{V_i} > \theta$ where g_i, V_i, θ are the gradient, second moment and pre-defined threshold (5000 used by default in their paper) respectively. We found that merely SpikeClip can mitigate the loss spike to some extent but can not prevent the training divergence completely. One possible explanation is that SpikeClip operates on an element-wise basis and may use a threshold that is too high. If all gradient components increase simultaneously, SpikeClip may still allow a large overall gradient norm, as it focuses solely on clipping individual outliers and does not effectively handle uniformly large gradients. This is supported by the observation in Figure 3 that the gradient norm remains high even after SpikeClip is applied.

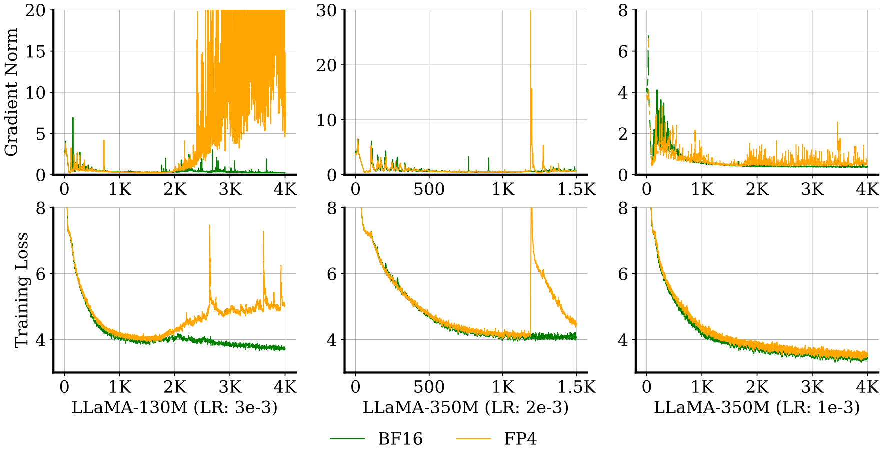

Figure 4: **Training loss and gradient norm of Adam using various learning rates with BF16 and FP4 precision.** Experiments are conducted under the same training configuration with LLaMA-130M/350M.

3 Stable-SPAM

To address the training instability in 4-bit LLM training, we propose `Stable-SPAM`, a stabilized spike-aware Adam optimizer. Apart from the momentum reset inherited from the original SPAM, `Stable-SPAM` introduces two techniques: Adaptive Gradient Norm (`AdaGN`) and Adaptive Spike-Aware Clipping (`AdaClip`), which we will explain in detail. The pseudocode is provided in Appendix B.

Adaptive Gradient Norm (`AdaGN`). As we can observe in Figures 4 and 3, spikes in training loss and instances of training divergence usually align with abrupt surges in the gradient norm, consistent with findings in Takase et al. (2023); Huang et al. (2025). To address these training instabilities, we propose `AdaGN`, a method that stabilizes gradients by adaptively scaling them based on their historical l_2 norm statistics. To better track the dynamics of the gradient norm during training, we leverage the idea of Adam by maintaining moving averages of both the first and second moments of the gradient norm. Concretely, we compute and update the moving averages of the gradient norm (m_{norm}, v_{norm}), then use them to derive a normalized gradient:

$$g_{\text{norm}} = \|g_t\|_2, \tag{1}$$

$$m_{\text{norm}} = \gamma_1 \cdot m_{\text{norm}} + \big(1-\gamma_1\big) \cdot g_{\text{norm}}, \tag{2}$$

$$v_{\text{norm}} = \gamma_2 \cdot v_{\text{norm}} + \big(1-\gamma_2\big) \cdot g_{\text{norm}}^2, \tag{3}$$

$$\hat{g}_t = \frac{g_t}{g_{\text{norm}}} \cdot \frac{m_{\text{norm}}}{\sqrt{v_{\text{norm}}} + \epsilon}. \tag{4}$$

where $\hat{g}_t$ is the normalized gradient, γ_1 and γ_2 are momentum coefficients and ϵ is small constant for numerical stability. By rescaling g_t with a ratio of its historical mean norm m_{norm} to the square root of its historical second moment $\sqrt{v_{norm}}$, `AdaGN` mitigates abrupt gradient norm spikes. Note that as the gradient norm g_{norm} is essentially a scalar for an entire layer, the additional parameter overhead introduced by `AdaGN` is negligible, i.e., two extra parameters per layer.

Adaptive Spike-Aware Clipping (`AdaClip`). Different from the spike gradient clipping technique in Huang et al. (2025), which sets a fixed clipping threshold, we propose an adaptive clipping approach, i.e., `AdaClip`. The core idea is to dynamically adjust the clipping threshold by tracking the maximum gradient magnitude observed over time, rather than relying on a pre-defined fixed value. Concretely, let g_t be the gradient at time step t. We first compute g_{max}, the maximum absolute gradient value across all parameters. Then, we update the threshold $T_{\text{threshold}}$ with an exponential moving average that incorporates $g_{\max}$. Finally, any entries of g_t that exceed $T_{\text{threshold}}$ are rescaled to maintain stability. The procedure is formally expressed as follows:

$$g_{\max} = \max_i(|g_t[i]|), \tag{5}$$

$$T_{\text{threshold}} = \gamma_3 \cdot T_{\text{threshold}} + (1 - \gamma_3) \cdot g_{\max}, \tag{6}$$

$$\text{Mask}_{\text{spikes}} = (g_t > T_{\text{threshold}}), \tag{7}$$

$$g_t[\text{Mask}_{\text{spikes}}] = \frac{g_t[\text{Mask}_{\text{spikes}}]}{g_{\max}} \times T_{\text{threshold}}, \tag{8}$$

where $\gamma_3 \in [0, 1]$ controls the weight of the moving average. When γ_3 is large, $T_{\text{threshold}}$ responds more slowly to new gradient maxima, leading to more stable updates. When γ_3 is small, it adapts more quickly to sharp changes in gradient magnitude.

Momentum Reset (`MoRet`). Following Huang et al. (2025), we adopt momentum reset (`MoRet`) to periodically reset the accumulated first and second moments in Adam. The effectiveness of `MoRet` lies in addressing the negative effects of gradient spikes, which can inflate the first and second moments of Adam. Since Adam uses exponential moving averages to track their historical information, these inflated values caused by spiked gradients can have prolonged detrimental effects (Huang et al., 2025) on moments. By resetting the momentum terms at fixed intervals (ΔT), `MoRet` mitigates the lasting influence of unusually large gradients, enabling more stable and consistent optimization.

Table 1: **Comparison of various optimizers of INT4 and FP4 training of LLaMA models on C4.** Perplexity is reported.

	INT4 Training			**FP4 Training**		
	130M	**350M**	**1B**	**130M**	**350M**	**1B**
Adam	26.4	24.14	21.59	28.9	24.59	22.01
Adam+GradClip	26.30	21.64	19.74	28.27	20.84	20.25
Adafactor	25.11	20.45	20.65	26.89	20.53	20.03
SPAM	25.03	20.19	19.98	26.78	20.35	19.74
`Stable-SPAM`	**24.33**	**17.76**	**17.42**	**26.31**	**19.49**	**18.48**
Adam (BF16)	24.53	21.38	19.73	24.53	21.38	19.73
Training Tokens	2.2B					

4 Experiments

To demonstrate the efficacy of the proposed `Stable-SPAM`, we conduct extensive experiments with various sizes of the LLaMA model on the C4 dataset. Please refer to the Appendix A for the detailed experimental settings.

To evaluate the performance of `Stable-SPAM` in 4-bit LLM training, we conduct experiments using both FP4 (E1M2: 1-bit exponent, 2-bit mantissa) and INT4 (4-bit integer) quantization-aware training strategies. The training curves of various LLaMA models on the C4 dataset are presented in Figure 1, and the final perplexity results are summarized in Table 1.

We observe that ❶ 4-bit training leads to a significant performance drop compared to BF16 training. As shown in Table 1, the perplexity gap between BF16 (Adam) and INT4/FP4 (Adam) exceeds 1.5 across all model sizes, highlighting the challenges of reduced precision. ❷ Figure 1 shows that `Stable-SPAM` consistently outperforms Adam by a significant margin in 4-bit scenarios, even surpassing the performance of 16-bit Adam. Table 1 further demonstrates that `Stable-SPAM` outperforms other advanced optimizers, such as Adafactor and SPAM. Among the baselines, incorporating GradClip reduces perplexity, while Adafactor and SPAM both outperform the simple application of GradClip. ❸ `Stable-SPAM` is able to match Adam's performance with half the tokens in 4-bit training. As illustrated in Figure 1, `Stable-SPAM` achieves the same perplexity as Adam in approximately half the training steps. ❹ Notably, `Stable-SPAM` performs particularly well with larger models, such as LLaMA-350M and LLaMA-1B, showcasing its strong potential for large-scale training. This is likely because large-scale, low-precision training is more susceptible to instability issues (Fishman et al., 2024), making stabilized training approaches like `Stable-SPAM` especially beneficial.

5 Conclusion

This paper presents a comprehensive study on the training instability challenges of 4-bit quantization in large language models. We find that while low-precision training significantly reduces memory and computational costs, it also amplifies the sensitivity to learning rates, and increases the likelihood of gradient and loss spikes. To address these issues, we propose `Stable-SPAM`, an optimizer that combines three key techniques: `AdaClip`, `AdaGN`, and `MoRet`. Empirical results on LLaMA models of various sizes demonstrate that `Stable-SPAM` not only stabilizes 4-bit training but also achieves better performance compared to existing optimizers, sometimes even surpassing BF16 performance. We additionally show that these stabilization strategies are broadly applicable, benefiting other optimizers like Lion and Adam-mini.

References

Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. Halo: Hadamard-assisted lossless optimization for efficient low-precision llm training and fine-tuning. *arXiv preprint arXiv:2501.02625*, 2025.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2024.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, et al. Cogview: Mastering text-to-image generation via transformers. *Advances in neural information processing systems*, 34:19822–19835, 2021.

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

Ian Goodfellow. Deep learning, 2016.

Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. Spam: Spike-aware adam with momentum reset for stable llm training. *arXiv preprint arXiv:2501.06842*, 2025.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pp. 4475–4483. PMLR, 2020.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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.

Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pp. 766–775, 2023.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2023.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory. *Advances in Neural Information Processing Systems*, 35:22941–22954, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Chao Ma, Wenbo Gong, Meyer Scetbon, and Edward Meeds. Swan: Preprocessing sgd enables adam-level performance on llm training with significant memory reduction. *arXiv preprint arXiv:2412.13148*, 2024.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

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.

Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. Fp8-lm: Training fp8 large language models. *arXiv preprint arXiv:2310.18313*, 2023.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, et al. What language model to train if you have one million gpu hours? *arXiv preprint arXiv:2210.15424*, 2022.

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

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.

Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. Stable and low-precision training for large-scale vision-language models. *Advances in Neural Information Processing Systems*, 36:10271–10298, 2023a.

Mitchell Wortsman, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, et al. Small-scale proxies for large-scale transformer training instabilities. *arXiv preprint arXiv:2309.14322*, 2023b.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168, 2023.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. Coat: Compressing optimizer states and activation for memory-efficient fp8 training. *arXiv preprint arXiv:2410.19313*, 2024a.

Haocheng Xi, Yuxiang Chen, Kang Zhao, Kai Jun Teh, Jianfei Chen, and Jun Zhu. Jetfire: Efficient and accurate transformer pretraining with int8 data flow and per-block quantization. *arXiv preprint arXiv:2403.12422*, 2024b.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pp. 40770–40803. PMLR, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024a.

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. Q-galore: Quantized galore with int4 projection and layer-adaptive low-rank gradients. *arXiv preprint arXiv:2407.08296*, 2024b.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024a.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024b.

A Experimental Setting

Baselines. We adopt five popular optimizers as our baselines including Adam (Kingma, 2014), Adafactor (Shazeer & Stern, 2018), Lion (Chen et al., 2024), Adam-mini (Zhang et al., 2024a) and SPAM (Huang et al., 2025). Among these, Adam and Adafactor are well-established and widely used, while Adam-mini and SPAM have been introduced more recently. Besides, we also include gradient clipping (Goodfellow, 2016) (GradClip) in conjunction with Adam as an additional baseline.

Experimental Setup. Following Lialin et al. (2023); Zhao et al. (2024a), we train LLaMA-based architectures ranging from 60M to 1B parameters. Each architecture is configured with RMSNorm (Shazeer, 2020) and SwiGLU activations (Zhang & Sennrich, 2019). For every model size, we keep the same set of hyperparameters across methods and vary only the learning rate. Specifically, we sweep over learning rates from 1×10^{-4} to 1×10^{-3} , incrementing by 2×10^{-4} for each optimizer.

Following the settings in Takase et al. (2023); Huang et al. (2025), we set the threshold to 1 for the GradClip baseline. For Adafactor, we adopt the hyperparameters from the original paper (Shazeer & Stern, 2018), where $\epsilon_1 = 10^{-30}$, $\epsilon_2 = 10^{-3}$, and $d = 1.0$. The hyperparameters for SPAM are configured based on the settings in Huang et al. (2025), with reset intervals set to 500, learning rate warmup steps to 150, and the GSS threshold to 5000. For `Stable-SPAM`, we set $\gamma_1 = 0.7$, $\gamma_2 = 0.9$ and $\theta = 0.999$ for 4-bit LLM training and $\gamma_1 = 0.85$, $\gamma_2 = 0.9999$ and $\gamma_3 = 0.999$ for BF16 training. Detailed descriptions of our task setups and hyperparameters are provided in the Appendix E.

B PSEUDOCODE

The pseudocode is presented in Alogrithm 1.

Algorithm 1: `Stable-SPAM`

```
Input: A layer weight matrix $w \in \mathbb{R}^{m\times n}$, learning rate $\alpha$, decay rates $\beta_1 = 0.9$, $\beta_2 = 0.999$, initial
    parameters $w_0$, $\gamma_1 = 0.7, \gamma_2 = 0.9$ for AdaGN and $\gamma_3 = 0.999$ for AdaClip, momentum reset
    interval $\Delta T$, small constant $\epsilon = 1 \times 10^{-6}$, and total training steps $T$.
Output: Optimized parameters $w_T$.
while $t < T$ do
    $g_t \in \mathbb{R}^{m\times n} \leftarrow -\nabla_w \phi_t(w_t)$        // Gradient of the objective at step $t$.
    $g_{\max} \leftarrow \mathtt{Max}(\mathtt{abs}(g_t))$
    $T_{\text{threshold}} \leftarrow T_{\text{threshold}} \cdot \theta + (1-\theta)\, g_{\max}$
    $\widehat{T}_{\text{threshold}} \leftarrow \frac{T_{\text{threshold}}}{1-\theta^t}$        // Bias correction for threshold
    $\text{Mask}_{\text{spikes}} \leftarrow \big(\mathtt{abs}(g_t) > \widehat{T}_{\text{threshold}}\big)$
    if $\mathit{sum}\big(\mathit{Mask}_{\text{spikes}}\big) > 0$ then
        $g_t[\text{Mask}_{\text{spikes}}] \leftarrow \frac{g_t[\text{Mask}_{\text{spikes}}]}{g_{\max}} \times \widehat{T}_{\text{threshold}}$
    $g_{\text{norm}} \leftarrow \|g_t\|_2$
    $m_{\text{norm}} \leftarrow \gamma_1\, m_{\text{norm}} + (1-\gamma_1)\, g_{\text{norm}}$
    $v_{\text{norm}} \leftarrow \gamma_2\, v_{\text{norm}} + (1-\gamma_2)\, g_{\text{norm}}^2$
    $\widehat{m}_{\text{norm}} \leftarrow \frac{m_{\text{norm}}}{1-\gamma_1^t}, \widehat{v}_{\text{norm}} \leftarrow \frac{v_{\text{norm}}}{1-\gamma_2^t}$        // Bias-corrected norm estimates
    $\text{adaptive\_norm} \leftarrow \frac{\widehat{m}_{\text{norm}}}{\sqrt{\widehat{v}_{\text{norm}}} + \epsilon}$
    $g_t \leftarrow \frac{g_t}{g_{\text{norm}}} \times \text{adaptive\_norm}$
    if $(\mathit{Mod}(t, \Delta T) = 0)$ then
        $\boldsymbol{m} \leftarrow \mathtt{zeros\_like}(\boldsymbol{m})$
        $\boldsymbol{v} \leftarrow \mathtt{zeros\_like}(\boldsymbol{v})$
    $\boldsymbol{m}_t \leftarrow \beta_1\, \boldsymbol{m}_{t-1} + (1-\beta_1)\, g_t$
    $\boldsymbol{v}_t \leftarrow \beta_2\, \boldsymbol{v}_{t-1} + (1-\beta_2)\, g_t^2$
    $\widehat{\boldsymbol{m}}_t \leftarrow \frac{\boldsymbol{m}_t}{1-\beta_1^t}$        // bias correction
    $\widehat{\boldsymbol{v}}_t \leftarrow \frac{\boldsymbol{v}_t}{1-\beta_2^t}$        // bias correction
    $w_t \leftarrow w_{t-1} - \alpha\, \frac{\widehat{\boldsymbol{m}}_t}{\sqrt{\widehat{\boldsymbol{v}}_t} + \epsilon}$
    $t \leftarrow t + 1$
return $w_T$.
```

C EXTRA EXPERIMENTS

C.1 PERFORMENCE OF BF16 LLM TRAINING

To further evaluate the efficacy of `Stable-SPAM`, we conducted experiments on various LLaMA model sizes using standard BF16 training. The experiments are based on C4 dataset. The training curves and final perplexity values are presented in Figure 5 and Table 2, respectively. Table 2 highlights that Stable-SPAM consistently delivers superior performance across different model sizes, surpassing the second-best optimizer with significant improvements. Furthermore, Figure 5 illustrates that Stable-SPAM achieves the same performance as Adam in only half the training steps or even fewer for LLaMA-350M and LLaMA-1B, validating its ability to match Adam's performance while requiring significantly fewer tokens under BF16 LLM training. The above results demonstrate that the promise of Stable-SPAM not only holds for low-precision LLM training but also holds for the standard BF16 training.

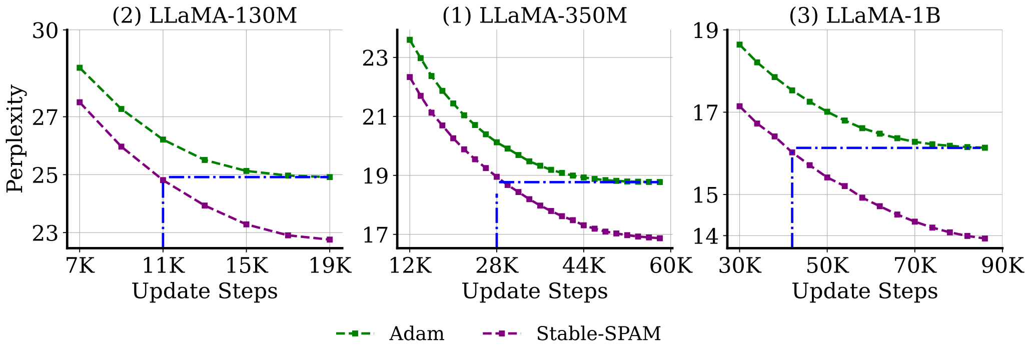

Figure 5: **Performance of BF16 training with various model sizes.** Experiments are based on LLaMA models trained on C4 Dataset.

Table 2: **Comparison among various optimizers on BF16 training.** Perplexity is reported.

Optimizer	60M	130M	350M	1B
Adam-mini	34.10	24.85	19.05	16.07
Adam	34.09	24.91	18.77	16.13
Adam + GradClip	33.33	24.88	18.51	15.22
Adafactor	32.57	23.98	17.74	15.19
SPAM	30.46	23.36	17.42	14.66
Stable-SPAM	**28.84**	**22.21**	**16.85**	**13.90**
Training Tokens	1.1B	2.2B	6.4B	11.6B

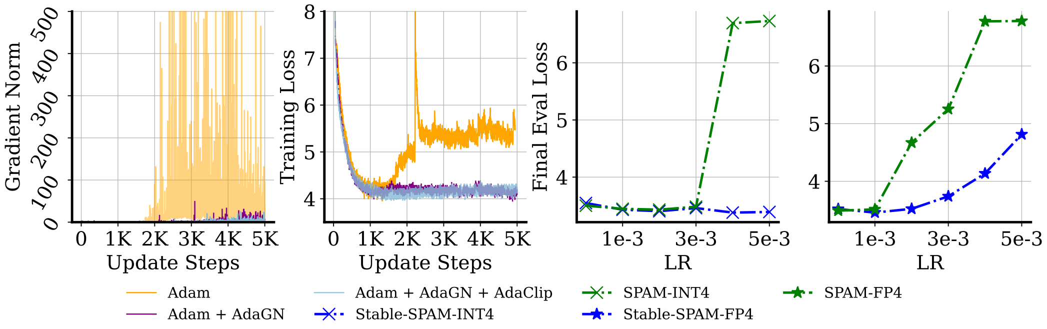

Figure 6: **Effect of AdaGN and AdaClip on stabilizing FP4 LLM training.** The left two figures use LLaMA-130M (LR = 3e-3), and the right two figures use LLaMA-60M.

C.2 Integration with Other Optimizers

Although AdaGN and AdaClip are proposed specifically for Stable-SPAM, one may wonder, "Can AdaGN and AdaClip also be compatible with other optimizers?" To answer this question, we applied AdaGN and AdaClip to two recently published optimizers: Lion Chen et al. (2024) and Adam-mini Zhang et al. (2024a). We conducted comparative experiments using Lion and Adam-mini alone, as well as in combination with AdaGN and AdaClip, under a 4-bit training setting. These experiments were performed on LLaMA-60M/130M models with the C4 dataset.

The results in Table 3 show that AdaGN and AdaClip consistently enhance the performance of both Lion and Adam-mini under FP4 and INT4 training settings, across LLaMA-60M and LLaMA-130M model sizes. Notably, on LLaMA-130M with INT4 training, Lion achieves a perplexity improvement of up to 5.88, and Adam-mini on LLaMA-60M under FP4 training sees an improvement of 1.72. These improvements underscore the broad applicability and effectiveness of the proposed AdaGN and AdaClip methods.

Table 3: **Performence of AdaGN and AdaClip on Lion and Adam-mini optimizers.** Experiments are based on LLaMA-60M/130M with 4-Bit training.

Optimizers	INT4 Training		FP4 Training	
	60M	**130M**	**60M**	**130M**
Lion	39.36	35.28	39.89	34.20
Lion+AdaGN+AdaClip	**38.49**	**29.40**	**36.75**	**31.63**
Adam-mini	34.84	29.79	36.37	32.95
Adam-mini+AdaGN+AdaClip	**34.61**	**29.65**	**34.65**	**32.39**
Training Tokens	1.1B			

C.3 Effect on Stabilizing Training

To validate the effectiveness of our proposed AdaGN and AdaClip techniques in stabilizing the LLM training process, Firstly, we compared the training loss and gradient norm curves across three settings: using Adam alone, using Adam with AdaGN, and using Adam with both AdaGN and AdaClip. Our experiments employed LLaMA-130M with a learning rate of 3e-3 under an FP4 training setting. As shown in Figure 6, training solely with Adam leads to divergence in the training loss and frequent spikes in the gradient norm. However, once AdaGN is introduced, the training loss converges, and the gradient norm is noticeably reduced. Adding AdaClip on top of AdaGN further decreases the gradient norm and yields a smoother training loss curve. Secondly, we present the final performance across a range of learning rates, from 5×10^{-4} to 5×10^{-3}, evaluated on LLaMA-60M under both FP4 and INT4 training settings. The results in Figure 6 show that Stable-SPAM produces a significantly flatter curve, highlighting its stability across varying learning rates. These results demonstrate the effectiveness of the proposed AdaGN and AdaClip techniques in achieving a more stable and consistent training process.

C.4 Ablation Study

To validate the effectiveness of the three components, MoRet, AdaGN, and AdaClip, in Stable-SPAM, we conduct a comprehensive ablation study. Specifically, we take two approaches: **(1)** We iteratively incorporate MoRet, AdaGN, and AdaClip into the Adam optimizer to measure their individual and combined improvements under both FP4 and BF16 training settings. **(2)** We replace AdaClip with SpikeClip (Huang et al., 2025) and AdaGN with GradClip (Goodfellow, 2016) to further assess the unique contributions of our proposed components. The results, summarized in Table 4, reveal the following observations: ❶ MoRet consistently improves performance across both FP4 and BF16 settings. ❷ Under both FP4 training, AdaGN alone shows limited improvement. However, when combined with AdaClip, it substantially reduces final perplexity. ❸ Conversely, in the BF16 setting, AdaGN alone yields considerable performance gains, but adding AdaClip offers limited improvement. This discrepancy may stem from the higher frequency of extreme element-wise gradient spikes in this FP4 training experiments, which necessitates AdaClip to correct biased update directions effectively. Finally, replacing AdaClip with SpikeClip (Huang et al., 2025) and AdaGN with GradClip (Goodfellow, 2016) results in increased perplexity, further validating the efficacy of our proposed AdaGN and AdaClip.

Table 4: **Ablations on Stable-SPAM.** Experiments are based on LLaMA-60M and C4.

Optimizer	FP4	BF16
Adam	35.47	34.09
Adam + MoRet	32.4	31.47
Adam + MoRet + AdaClip	31.97	30.29
Adam + MoRet + AdaGN	32.26	28.96
Adam + MoRet + AdaGN + AdaClip (**Stable-SPAM**)	**31.40**	**28.84**
Adam + MoRet+AdaGN+SpikeClip Huang et al. (2025)	32.01	28.90
Adam + MoRet+ GradClip Goodfellow (2016)+AdaClip	31.95	29.87
Adam + MoRet+AdaGN+AdaClip (**Stable-SPAM**)	**31.40**	**28.84**
Training Tokens	1.1B	

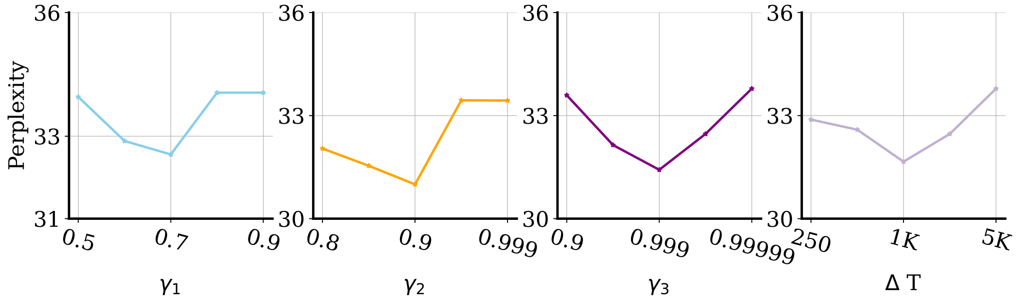

Figure 7: **Hyper-parameter Analysis.** Experiments are conducted with FP4 training on LLaMA-60M and C4 with 1.1B tokens.

C.5 Hyper-Parameter Analysis

Stable-SPAM introduces four hyperparameters: γ_1, γ_2, γ_3, and ΔT, which extend the functionality of Adam. Among these, γ_1 and γ_2 serve a similar purpose to β_1 and β_2 in Adam, controlling the smoothness of updates to the first moment m_{norm} and the second moment v_{norm}. Larger values of γ_1 and γ_2 result in smoother updates, placing greater emphasis on historical gradient norm statistics when adapting the current gradient norm. Similarly, γ_3 plays a role in determining the threshold for identifying gradient spikes. A larger γ_3 leads to a smoother and more conservative threshold, resulting in a higher proportion of gradients being classified as spike gradients. To investigate the impact of these hyperparameters, we plot the final perplexity curve while varying γ_1 from 0.5 to 0.9, γ_2 from 0.8 to 0.999, γ_3 from 0.9 to 0.999, and ΔT from 250 to 5000. The experiments are conducted using LLaMA-60M, trained on 1.1B C4 tokens under the FP4 training setting. The results in Figure 7 demonstrate that overly small or excessively large values of these hyperparameters can degrade performance. However, the intuitive interpretations of these hyperparameters make them straightforward to tune, and they typically require minimal adjustments. In this paper, we adopt the optimal values $\gamma_1 = 0.7$, $\gamma_2 = 0.9$, $\gamma_3 = 0.999$, and $\Delta T = 1000$, which work effectively for all 4-bit training scenarios.

D Related Work

Instability of Training Large Language Models. The instability of large language model (LLM) training, which are marked by loss spikes and catastrophic divergence(Chowdhery et al., 2023; Molybog et al., 2023), has driven extensive research into stabilization techniques. These methods generally fall into three main categories: (1) gradient preprocessing, (2) architectural modifications, and (3) initialization strategies. Gradient preprocessing typically involves scaling and clipping gradients at the start of the optimization process to stabilize the training. A well-known example is gradient clipping (Goodfellow, 2016), which globally rescales the gradient norm to a fixed value. Later, Adafactor (Shazeer & Stern, 2018) introduced capping the norm of the parameter updates instead of the raw gradients. More recently, SPAM (Huang et al., 2025) proposed detecting and clipping anomalous gradients based on historical gradient statistics. However, a common drawback of these methods is that they require manually setting a predefined threshold. Architecturally, Xiong et al. (2020) showed that Post-LayerNorm (Post-LN) amplifies gradients, causing instability with large learning rates, while Pre-LayerNorm (Pre-LN) preserves gradient norms for stable training. Embed LayerNorm (Embed LN) normalizes embeddings(Dettmers et al., 2021), though it may impact performance(Scao et al., 2022), while Embed Detach(Ding et al., 2021; Zeng et al., 2022) reduces loss spikes by truncating gradients. DeepNorm(Wang et al., 2024) scales residual connections to stabilize ultra-deep models, and αReparam(Zhai et al., 2023) prevents attention entropy collapse via spectral-normalized parameterization. Initialization strategies offer complementary stability benefits. Scaled Embed(Takase et al., 2023) stabilizes LayerNorm gradients, while Scaled Initialization(Nguyen & Salazar, 2019) reduces variance using $\mathcal{N}(0, \sqrt{2/(5d)}/\sqrt{2N})$. Fixup(Zhang et al., 2019; Huang et al., 2020) eliminates LayerNorm entirely, inspiring norm-free architectures. Though ongoing advancements refine these approaches, training stability remains a key challenge in LLM development.

Low-precision LLM Training. Low-precision training (Wang et al., 2018; Lin et al., 2022; Xi et al., 2024a;b; Wortsman et al., 2023a) has emerged as a promising approach to improve both computational and memory efficiency during training. Among these methods, FP16 (Micikevicius et al., 2017) and BF16 (Kalamkar et al., 2019) are the most widely adopted precision formats. To push the efficiency further, 8-bit training has garnered increasing attention. For instance, LM-FP8 (Peng et al., 2023) enables training with FP8 precision While Fishman et al. (2024) demonstrates that as training scales up (larger than 250B tokens), the issue of activation outliers becomes more pronounced, posing challenges to the representation range of low-bit data formats. To address this challenge, Fishman et al. (2024) proposes a smoothing strategy, while Ashkboos et al. (2025) leverages Hadamard transformations to mitigate the impact of activation outliers. Furthermore, the choice of data format significantly influences training performance. The INT8 format is the most widely supported low-precision format, whereas FP8, available in NVIDIA's Hopper GPU architecture, provides specialized support. Additionally, the MX format (Rouhani et al., 2023) demonstrates superior representational capability, though it is rarely supported by current hardware. In this work, we investigate the training instability associated with low-precision training and propose enhancements through the design of optimizers. Our approach is compatible with existing techniques, providing a complementary solution to improve the stability of low-precision training.

E Architecture and Hyperparameters

We introduce details of the LLaMA architecture and hyperparameters used for 4-bit and BF16 pre-training, following Lialin et al. (2023); Zhao et al. (2024a). Table 5 shows the most hyperparameters of LLaMA models across model sizes. We use a max sequence length of 256 for all models, with a batch size of 512, with a batch size of 131K tokens. For all experiments, we adopt learning rate warmup of 2000 training steps, and use cosine annealing for the learning rate schedule, decaying to 10% of the initial learning rate.

Table 5: Configurations of LLaMA models used in this paper.

Params	Hidden	Intermediate	Heads	Layers
60M	512	1376	8	8
130M	768	2048	12	12
350M	1024	2736	16	24
1 B	2048	5461	24	32

For all methods across each model size (from 60M to 1B), we tune the learning rates from $1e-4$ to $1e-3$ with an increasing step of 2×10^{-4} for pre-training tasks, and the best learning rate is selected based on the validation perplexity. The detailed hyperparameter of `Stable-SPAM` on 4-bit training and BF16 training are reported in Table 6 and Table 7.

F Time Series Forescasting Task

We conducted additional experiments on time-series prediction tasks. In these experiments, we intentionally introduced anomalous data with a probability A=10% to simulate gradient anomalies.

Table 6: Hyperparameters of `Stable-SPAM` for 4-bit pre-training experiments in this paper.

Hyper-Parameters	LLaMA-130M	LLaMA-350M	LLaMA-1B
LR	$1e-3$	$4e-4$	$2e-4$
ΔT	1000	1000	1000
γ_1	0.7	0.7	0.7
γ_2	0.9	0.9	0.9
γ_3	0.999	0.999	0.999

Table 7: Hyperparameters of `Stable-SPAM` for BF6 pre-training experiments in this paper.

Hyper-Parameters	LLaMA-60M	LLaMA-130M	LLaMA-350M	LLaMA-1B
	Standard Pretraining			
LR	$1e-3$	$8e-4$	$4e-4$	$2e-4$
ΔT	1000	1000	1000	1000
γ_1	0.85	0.85	0.85	0.85
γ_2	0.99999	0.99999	0.99999	0.99999
γ_3	0.999	0.999	0.999	0.999

Experiments are conducted with 10 repeated runs on Weather time series data[2] using PatchTST (Nie et al., 2022) model. The results are presented in Figure 8.

The findings demonstrate that as the severity of anomalous data increases, `Stable-SPAM`'s performance advantage over Adam becomes more pronounced. Besides, `Stable-SPAM` consistently surpasses SPAM across all settings. These results further highlight the effectiveness of the proposed `Stable-SPAM`.

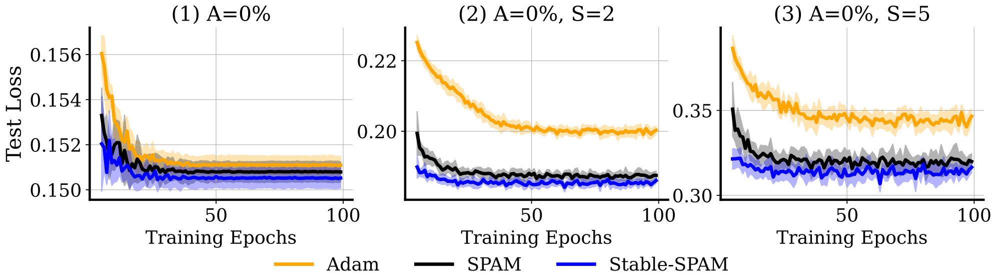

Figure 8: **Test Loss during Training Process on Weather Time-series Data.** Anomalous data is generated by adding Gaussian noise to 10% of randomly selected input values. Specifically, the anomalies data are conducted with $X = X + \texttt{Gaussin}(0, \texttt{Severity} * \texttt{Max}(X))$ where X is the inputs.

[2] `https://www.bgc-jena.mpg.de/wetter/`