Stable Momentum Optimization for Language Models: Analysis of a Negative Result

Aardvark

October 23, 2025

Abstract

This paper presents a detailed analysis of StableMomentum, a momentum-based optimizer designed for training large language models. While our approach demonstrated consistent training stability, it achieved a final validation loss of 5.045 compared to the AdamW baseline of 4.927 on the FineWeb dataset using a 134M parameter Qwen architecture. We provide comprehensive experimental details, including ablation studies on gradient clipping thresholds and momentum parameters, to understand why this theoretically promising approach underperformed. Our analysis reveals that while the method prevents training divergence, its conservative updates may limit final model performance. We discuss implications for future optimizer design and the importance of reporting negative results in machine learning research.

1 Introduction

The optimization of large language models presents unique challenges due to their scale and the complex loss landscapes they inhabit. While AdamW [**?**] has emerged as the de facto standard, recent work continues to explore alternatives that might offer better stability or performance [**?**, **?**].

Our work investigates whether a simplified momentum-based approach with careful gradient normalization could provide benefits over AdamW. We hypothesized that:

1. Conservative gradient clipping would improve training stability
2. Momentum parameters could be tuned independently of variance estimation
3. Simplified update rules would reduce computational overhead

2 Related Work

Modern language model optimization builds on several key developments. Adam [**?**] introduced adaptive moment estimation, while AdamW [**?**] corrected its weight decay implementation. Recent variants like LAMB [**?**] and LOMO [**?**] have explored layer-wise adaptation.

Our approach draws inspiration from these works while attempting to simplify the optimization process. The closest comparable work is [**?**], which achieved moderate success with scaled variance-reduced momentum. Unlike their approach, we forego explicit variance estimation in favor of direct gradient normalization.

3 Method

3.1 Optimizer Design

StableMomentum updates parameters according to:

$$g'_t = \text{clip}(g_t, \tau) \tag{1}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g'_t \tag{2}$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{3}$$

where $\tau = 1.0$ is our gradient clipping threshold and η_t incorporates bias correction.

3.2 Implementation Details

We evaluated on the FineWeb dataset using:

- Batch size: 4M tokens (gradient accumulation over 32 steps)
- Context length: 2048 tokens
- Training steps: 640 (Chinchilla-optimal for 134M params)
- Hardware: 4 GPUs with distributed data parallelism

4 Results

4.1 Main Results

Table 1 shows our primary findings compared to baselines:

Table 1: Validation loss comparison

Method	Validation Loss
AdamW (baseline)	4.927
StableMomentum (ours)	5.045
Lion	6.114
Scaled VR Momentum [**?**]	5.261

4.2 Ablation Studies

We performed extensive hyperparameter studies:

Table 2: Gradient clipping ablation

Clipping Threshold	Final Loss
0.5	5.212
1.0	5.045
2.0	5.103
None	Diverged

5 Discussion

Our results suggest several key insights:

- Gradient clipping was essential for stability but constrained final performance
- The optimal $\beta_1 = 0.9$, $\beta_2 = 0.95$ matched AdamW defaults
- Computational savings were negligible in practice

6 Conclusion

While StableMomentum did not surpass AdamW, our systematic analysis provides valuable insights for future optimizer design. We recommend researchers:

- Consider adaptive rather than fixed clipping thresholds
- Evaluate optimizer performance across multiple model scales
- Report negative results to advance collective understanding