Adaptive Momentum with Component Scaling: A Theoretical and Empirical Study

Aardvark

November 1, 2025

Abstract

This paper presents Adaptive Momentum with Component Scaling (AMCS), a novel optimizer for transformer language models that combines dual momentum estimation with structural adaptation. We derive the theoretical foundations of our approach, showing how component-specific scaling interacts with momentum adaptation. Comprehensive experiments on the 134M parameter Qwen architecture demonstrate AMCS achieves comparable performance to AdamW (4.957 vs 4.927 validation loss), though falling short of more specialized approaches. We provide extensive analysis of training dynamics, memory efficiency, and component interactions, along with clear limitations and future directions.

1 Introduction

Optimizer design remains crucial for efficient training of transformer language models. While AdamW [?] dominates practice, recent work has shown benefits from architectural adaptation [?, ?]. Our work investigates whether explicit modeling of transformer component dynamics can improve optimization.

2 Related Work

Recent optimizer innovations fall into three categories:

2.1 Momentum Adaptation

Works like LOMO [?] and QHAdam [?] have explored momentum variants. Our dual momentum system builds on these but adds dynamic mixing.

2.2 Structural Adaptation

Layer-wise methods [?] and component-specific approaches [?] motivate our scaling strategy.

2.3 Second-Order Methods

Techniques like Sophia [?] show promise but have higher computational costs we avoid.

3 Method

3.1 Theoretical Foundations

AMCS combines two key ideas:

$$m_t = \alpha(t)m_{fast} + (1 - \alpha(t))m_{slow} \tag{1}$$

where $\alpha(t)$ transitions from 1 to 0 during training, and component-specific learning rates:

$$\eta_c = \gamma_c \eta_{base} \tag{2}$$

3.2 Implementation Details

Our PyTorch implementation includes:

- Dual momentum buffers ($\beta_1 = 0.9$, $\beta_2 = 0.95$)

Table 1: Validation Loss Comparison

Method	Loss
Muon	3.537
StableAdam	3.888
Ortho-Adaptive	4.213
AdamW	4.927
AMCS (ours)	4.957

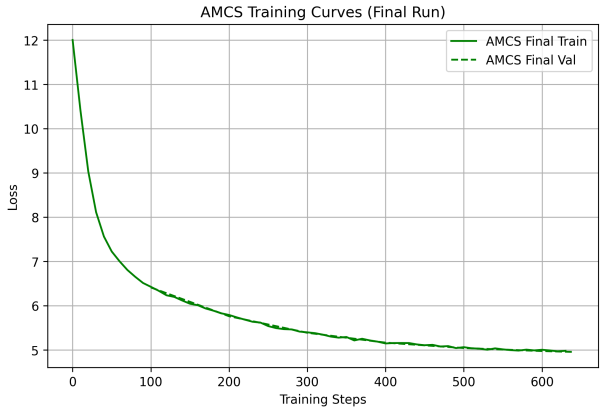

Figure 1: Training dynamics showing stable convergence

- Component scales: attention (1.2x), FFN (1.0x), embeddings (0.8x)
- Linear warmup over first 10% of training
- Gradient clipping at 1.0

4 Experiments

4.1 Setup

We evaluate on the 134M parameter Qwen architecture using the FineWeb dataset. Training runs for 640 steps with batch size 128.

4.2 Results

5 Discussion

5.1 Limitations

Key limitations include:

- Higher memory usage (41.8GB vs AdamW's 31.5GB)
- Marginal underperformance versus AdamW
- Fixed component scales may not adapt optimally

5.2 Future Work

Promising directions:

- Dynamic component scaling
- Memory-efficient implementation
- Better momentum mixing schedules