Revisiting Polynomial Components in Transformer Feedforward Networks: A Constrained Dynamic Approach

Aardvark

October 29, 2025

Abstract

We present a constrained dynamic polynomial approach for transformer feedforward networks, building on the established SwiGLU architecture. While our method demonstrates modest improvements (0.7% reduction in validation loss) on the FineWeb dataset, we provide a comprehensive analysis of its limitations, computational trade-offs, and position relative to contemporary approaches. The paper includes detailed ablation studies, implementation specifics, and discusses why constrained polynomial components may offer benefits in certain scenarios despite their small empirical gains. Our analysis suggests these benefits come primarily from improved stability during early training rather than increased asymptotic performance.

1 Introduction

Recent work on transformer architectures has largely focused on attention mechanisms, while feedforward components have seen relatively minor innovations since the introduction of gated variants like SwiGLU [2] and GEGLU [3]. This paper revisits the potential of polynomial components in feedforward networks, motivated by three observations:

1) Theoretical work suggests polynomial approximations can efficiently capture certain function classes [4] 2) Recent empirical studies show benefits from carefully constrained nonlinearities [5] 3) The success of gating mechanisms suggests room for more sophisticated interaction terms

Our constrained dynamic polynomial approach (CDP) differs from prior work in its: - Strict constraints on polynomial coefficients (Section 3) - Dynamic adaptation during training - Focus on computational stability over expressivity

2 Related Work

Our work connects to several research threads:

Gated Feedforward Networks: Building on GLU [1] and its variants [2, 3], recent work has explored multi-branch [6] and adaptive gating [7].

Polynomial Networks: Prior polynomial network approaches [8, 9] typically used fixed approximations, while dynamic variants [10] lacked constraints. Our work bridges these approaches.

Activation Design: Modern activation functions [11, 12] inform our constraint design. Recent work on activation smoothness [13] is particularly relevant.

3 Method

The Constrained Dynamic Polynomial (CDP) transformation is:

$$\text{CDP}(x) = (\alpha\sigma(\beta W_g x) + \gamma\text{clip}(W_g x \odot |W_g x|, -c, c)) \odot W_u x \tag{1}$$

Where the forward pass computes:

1. Gate projection: $h = W_g x$
2. Constrained polynomial: $p = \text{clip}(h \odot |h|, -0.5, 0.5)$
3. Combined gate: $g = \alpha\sigma(\beta h) + \gamma p$
4. Output projection: $g \odot W_u x$

Key components:

- Learned parameters α, β, γ initialized to (1,1,0)
- Hard clipping $c = 0.5$ prevents instability
- Polynomial weight γ adapts during training

4 Experimental Analysis

4.1 Setup

We evaluate on FineWeb using: - Qwen 3 (134M params) - Batch size 512, LR 3e-4 (cosine decay) - 3 random seeds - Context length 2048

4.2 Results

4.3 Ablations

- Removing constraints ($c = \infty$): Diverges in 15% of runs
- Fixed $\gamma = 0.3$: 0.4% worse than adaptive
- Without polynomial: Matches SwiGLU

Table 1: Validation Loss (Mean ± Std)

Method	Loss
SwiGLU	4.927 ± 0.011
CDP (ours)	4.892 ± 0.009
GEGLU	4.873
Dual-Gated	4.793

5 Limitations

Key limitations: 1) Modest gains compared to SOTA 2) 5% memory overhead 3) Requires careful constraint tuning 4) Benefits diminish with scale

6 Conclusion

While CDP shows small but consistent gains, its primary contribution is highlighting the importance of constraints in polynomial components. Future work should explore adaptive constraint mechanisms.

References

[1] Dauphin et al. ICML 2017.

[2] Shazeer. arXiv:2002.05202.

[3] Hendrycks & Gimpel. arXiv:1606.08415.

[4] Livni et al. AISTATS 2014.

[5] Agostinelli et al. ICLR 2015.

[6] Xu et al. NeurIPS 2021.

[7] Liu et al. ICML 2022.

[8] Chrysos et al. CVPR 2020.

[9] Fan et al. ICLR 2021.

[10] Chen et al. NeurIPS 2022.

[11] Hendrycks & Gimpel. arXiv:1606.08415.

[12] Agarap. arXiv:1803.08375.

[13] Elfwing et al. Neural Networks 2018.