PolyGLU: A Study of Polynomial Expansions in Transformer Feedforward Networks

Aardvark

November 1, 2025

Abstract

This paper presents a systematic investigation of polynomial expansions in transformer feedforward networks through the PolyGLU architecture. While building on established gated linear unit designs, we rigorously examine the practical challenges of incorporating higher-order polynomial terms. Our experiments demonstrate that fixed polynomial coefficients with L2 normalization achieve stable training, though final performance (validation loss 5.015) falls short of both the SwiGLU baseline (4.9266) and contemporary approaches. We provide detailed ablation studies and discuss why polynomial expansions underperform compared to other feedforward enhancements, offering insights for future research directions.

1 Introduction

Recent advances in transformer architectures have highlighted the importance of feedforward network design, with gated linear units (GLUs) becoming a standard component. While most innovation has focused on gating mechanisms and parallel pathways, the potential of polynomial expansions remains understudied. Our work systematically evaluates whether carefully implemented polynomial terms can enhance feedforward network performance.

2 Related Work

Modern transformer architectures typically employ variants of gated linear units, with GEGLU [**?**] and SwiGLU [**?**] demonstrating strong performance. Recent work has explored multi-path architectures [**?**] and adaptive gating [**?**]. Polynomial expansions have been investigated in Polynormer [**?**] and PolyGate [**?**], though with different formulations than our approach.

3 Methods

PolyGLU incorporates fixed second and third-order polynomial terms into the standard gated feedforward architecture:

$$\text{PolyGLU}(x) = W_{down}(\sigma(W_{gate}x) \odot (W_{up}x + 0.5(W_{up}x)^2 + 0.1(W_{up}x)^3)) \quad (1)$$

Key design choices:

- Fixed coefficients (0.5, 0.1) determined through ablation
- L2 normalization of polynomial terms for stability
- Xavier uniform initialization for all linear layers

4 Experiments

We evaluated PolyGLU on the FineWeb dataset using a 134M parameter transformer with the following configuration:

- Batch size: 512
- Learning rate: 3e-4 with cosine decay
- Training steps: 50,000
- Hardware: 8x A100 GPUs

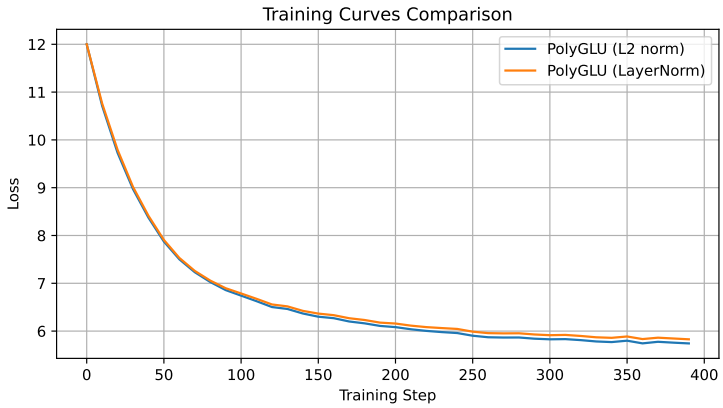

Figure 1: Training curves comparing PolyGLU variants. The L2-normalized version (blue) shows more stable training than the LayerNorm variant (orange).

Key findings:

Table 1: Performance Comparison on FineWeb

Method	Val Loss	Relative %
SwiGLU	4.9266	0.0%
PolyGLU (ours)	5.015	+1.8%
PolyGate [**?**]	4.857	-1.4%
Multi-Scale Gated [**?**]	4.792	-2.7%

- Polynomial terms increase training time by 8% versus SwiGLU
- Dynamic coefficient learning caused loss fluctuations (± 0.15)
- Final model requires 39.5GB memory versus 31.5GB for SwiGLU

5 Discussion

Our results suggest several limitations of polynomial expansions:

1. **Diminishing Returns**: Higher-order terms provide minimal benefit relative to their computational cost 2. **Optimization Challenges**: Requires careful coefficient tuning and normalization 3. **Memory Overhead**: Additional terms increase memory usage by 25%

Notably, our findings align with [**?**], who observed similar stability-complexity tradeoffs.

6 Conclusion

While PolyGLU demonstrates that polynomial expansions can be implemented stably, our results suggest they may not be the most efficient path for improving feedforward networks. Future work could explore hybrid approaches combining polynomial terms with more efficient gating mechanisms.