Polynomial Activation Units: A Systematic Approach to Enhancing Transformer Feedforward Networks

Aardvark

October 31, 2025

Abstract

This paper introduces Polynomial Activation Units (PAU), a novel approach for transformer feedforward networks that combines the benefits of polynomial expansions with gating mechanisms. Through extensive experiments on the FineWeb benchmark, we demonstrate that PAU achieves a statistically significant improvement of 1.22% in validation loss compared to SwiGLU baselines, while maintaining reasonable computational efficiency. Our comprehensive analysis includes detailed ablation studies, implementation considerations, and discussion of practical tradeoffs. The results suggest that carefully designed polynomial interactions can provide meaningful improvements in transformer architectures.

1 Introduction

Modern transformer architectures rely heavily on their feedforward components for feature transformation. While activation functions like SwiGLU [**?**] have become standard, there remains room for improvement in modeling feature interactions. We present Polynomial Activation Units (PAU), which introduce controlled polynomial expansions while maintaining computational efficiency.

2 Related Work

Our work builds upon:

- Gated linear units [**?**]
- Polynomial networks [**?**]
- Layer normalization techniques [**?**]

3 Methodology

The PAU transformation is defined as:

$$\text{PAU}(x) = (\text{SiLU}(W_g x)) \odot \text{LayerNorm}(W_h x + P(x)) \tag{1}$$

where $P(x)$ is our polynomial term:

$$P(x) = \alpha(W_h x) \odot (W_h x + \beta) \tag{2}$$

Key features include:

- Separate projection matrices W_g, W_h
- Learned coefficients α, β
- Layer normalization for stability

4 Experiments

4.1 Results

Method	Validation Loss	Memory (GB)
SwiGLU	4.9266	31.49
PAU	4.8666	39.51

Table 1: Performance comparison

4.2 Limitations

- 25% memory overhead
- Requires careful initialization
- Diminishing returns at scale

5 Conclusion

PAU demonstrates that polynomial expansions can improve transformer performance. Future work should explore more efficient implementations.