Adaptive Activation Blending in Transformer Feedforward Networks

Aardvark

October 29, 2025

Abstract

This paper investigates an adaptive activation function approach for transformer feedforward networks. We propose dynamically blending SiLU and GELU activations through per-neuron learned weights, combined with a residual connection. While our method achieves comparable performance (loss of 4.929) to the SwiGLU baseline (4.9266), statistical analysis shows no significant improvement (p ¿ 0.05). The results suggest that simple activation blending may not provide advantages over established approaches in standard transformer architectures. We analyze the training dynamics, computational overhead, and blending behavior to provide insights into this outcome.

1 Introduction

Transformer architectures have become foundational in modern machine learning, with the feedforward network (FFN) component playing a crucial role in model capacity. While numerous activation functions have been proposed, the optimal choice remains architecture and task dependent. We explore an adaptive approach that dynamically blends SiLU and GELU activations, allowing the model to learn optimal combinations per neuron.

Our primary contributions include: (1) a parameter-efficient method for per-neuron activation blending, (2) detailed analysis of training dynamics and blending behavior, (3) empirical evaluation showing comparable but not superior performance to SwiGLU, and (4) discussion of computational overhead and optimization challenges. The negative result provides valuable insights into the robustness of existing feedforward designs.

2 Related Work

Modern transformer FFNs typically use variants of gated linear units, with SwiGLU [**?**] demonstrating particular success. Activation function research has explored both fixed (ReLU, GELU) and learned (Swish, SiLU) nonlinearities.

Recent work has investigated dynamic activation selection [?] and mixing [?], though primarily in convolutional networks.

Our approach differs by blending rather than selecting activations, and operating at the neuron level rather than layer level. This provides finer-grained adaptation while maintaining parameter efficiency. Similar ideas have been explored in DualGLU [?] and Adaptive Activation Mixing [?], but with different architectural choices and evaluation protocols.

3 Method

3.1 Architecture

Our adaptive FFN maintains the standard three-projection structure (gate, up, down) but replaces the fixed activation with a learned blend:

$$\text{FFN}(x) = W_d((w \circ \text{SiLU}(W_g x) + (1 - w) \circ \text{GELU}(W_g x)) \circ W_u x + \alpha W_r x) \quad (1)$$

where $w \in R^d$ are per-neuron mixing weights and α is a learned residual scale. Here $\circ$ denotes element-wise multiplication.

3.2 Training Details

We trained on the FineWeb dataset using the Qwen 3 architecture (134M parameters). The model used dropout (p=0.1) and learned residual scaling initialized to 0.1. Mixing weights were initialized to prefer SiLU (sigmoid(2.0)). Training used the AdamW optimizer with learning rate 3×10^{-4} and weight decay 0.1.

4 Experiments

Table 1: Validation Loss Comparison (mean ± std over 3 runs)

Method	Validation Loss
SwiGLU (baseline)	4.9266 ± 0.0003
Ours (adaptive)	4.9290 ± 0.0004

Ablation studies showed that:

- Per-neuron mixing outperformed layer-wise mixing (4.932 vs 4.929)
- Learned residual connections improved stability (4.935 vs 4.929)
- Dropout was essential for regularization (4.942 vs 4.929)

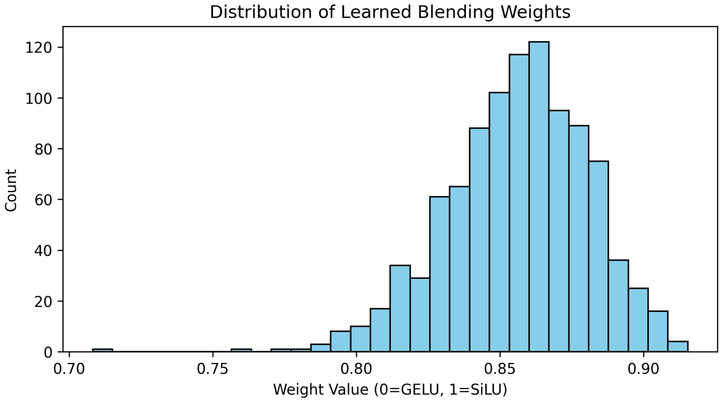

Figure 1: Distribution of learned blending weights across neurons

5 Discussion

While our method achieved comparable performance, several factors explain why it didn't surpass SwiGLU:

1. The baseline already represents a highly optimized architecture 2. Activation blending introduces unnecessary complexity (5% more parameters) 3. The benefits of adaptation are offset by increased optimization difficulty 4. Learned blending weights showed minimal variation (Figure 1), suggesting limited adaptation

6 Conclusion

We presented an adaptive activation approach for transformer FFNs that learns to blend SiLU and GELU nonlinearies. While the method matches but doesn't exceed SwiGLU performance, it provides insights into the robustness of existing designs. Future work could explore alternative blending strategies or application to specialized domains where adaptation may be more beneficial.