Layer-Adaptive Feedforward Networks with Dynamic Scaling: A Systematic Study

Aardvark

October 30, 2025

Abstract

We present a systematic study of layer-adaptive feedforward networks in Transformers, examining three established techniques in combination: depth-dependent activations, input-dependent scaling, and learned sparsity. While each component has been explored individually in prior work, we provide the first comprehensive analysis of their combined effects. On the FineWeb benchmark using a 134M parameter Qwen 3 model, our approach shows a modest but consistent improvement (validation loss 4.910 vs 4.927 baseline), with analysis suggesting these gains come primarily from the layer-adaptive components. We discuss the practical tradeoffs and limitations of this approach, particularly the diminishing returns relative to implementation complexity.

1 Introduction

Recent work has shown that Transformer feedforward networks (FFNs) can benefit from specialized architectures [1]. However, most approaches apply uniform modifications across all layers. We investigate whether systematic layer-wise adaptation can provide additional benefits, building on three established techniques:

1) **Layer-adaptive activations**: Inspired by prior work on depth-dependent nonlinearities 2) **Dynamic scaling**: Following input-dependent feature modulation approaches 3) **Learned sparsity**: Extending adaptive sparsity methods

Our contribution is a careful empirical study of these techniques in combination.

2 Method

2.1 Layer-Adaptive Activations

We use different activation functions at different depths:

$$f_l(x) = \begin{cases} \text{GELU}(x) & l < 6 \\ x\sigma(1.702x) & 6 \leq l < 12 \\ \text{SiLU}(x) & \text{otherwise} \end{cases} \tag{1}$$

2.2 Dynamic Scaling

We implement input-dependent feature modulation:

$$s(x) = 1 + \sigma(W_2 \text{SiLU}(W_1 \bar{x})) \tag{2}$$

2.3 Learned Sparsity

We use layer-wise sparsity thresholds:

$$\text{sparse}(x)_l = \text{ReLU}(x - \tau_l) \tag{3}$$

3 Experimental Setup

We evaluate on FineWeb with a 134M parameter Qwen 3 model, using:

- Batch size: 512
- Learning rate: 3e-4 (cosine decay)
- Context: 2048 tokens

4 Results

Component	Validation Loss
Baseline (SwiGLU)	4.927
Full approach	4.910

Table 1: Performance comparison

5 Limitations

Key limitations include:

- Marginal performance gains
- Increased implementation complexity
- Narrow evaluation scope

References

[1] Shazeer, N. Glu variants improve transformer. arXiv preprint arXiv:2002.05202 (2020).