Revisiting Sparse Gating in Transformer Feedforward Networks: An Empirical Study

Aardvark

October 30, 2025

Abstract

This paper presents a systematic investigation of sparse gating mechanisms in transformer feedforward networks (FFNs). While recent work has demonstrated the effectiveness of sparsity in attention layers, its application to FFNs remains understudied. We evaluate a novel sparse gated FFN architecture combining key-value memory structures with selective activation patterns and residual gating. Our comprehensive experiments on a 134M parameter language model reveal that while the approach reduces theoretical FLOPs by 85%, it results in a 3.3% increase in validation loss compared to the SwiGLU baseline. We analyze the failure modes through extensive ablations and provide insights for future sparse FFN designs.

1 Introduction

Transformer architectures have revolutionized natural language processing, with the feedforward network (FFN) component accounting for up to two-thirds of parameters in modern models. Recent work has reinterpreted FFNs as key-value memory systems, while others have explored sparsity in attention mechanisms. Our work bridges these approaches through rigorous empirical evaluation.

2 Related Work

Our work builds on several research threads:

Sparse Transformers: Building on sparse attention patterns, we extend sparsity to FFN layers while maintaining fixed compute budgets.

FFN as Memory: Following key-value memory interpretations of FFNs, we decompose projections while analyzing sparsity effects.

Gating Mechanisms: Our residual gating draws from both Highway Networks and recent adaptive computation approaches.

3 Method

3.1 Architecture Overview

Our Sparse Gated FFN combines three components:

$$FFN(x) = Gate(x) \cdot (W_{down}(SparseAct(W_k x) \circ W_v x)) + (1 - Gate(x)) \cdot x \quad (1)$$

3.2 Key Components

Key-Value Decomposition: We split the FFN into separate projections:

$$K = GeLU(W_k x), \quad V = W_v x \quad (2)$$

Selective Activation: We implement sparsity via:

$$SparseAct(z) = z \circ mask(z, k = 0.15 d_{hidden}) \quad (3)$$

where *mask* selects the top-k values.

Residual Gating: A learned mechanism balances computation:

$$Gate(x) = \sigma(LayerNorm(W_g x)) \quad (4)$$

4 Experimental Setup

We evaluate on the FineWeb dataset using:

- Model: Qwen architecture (134M params)
- Training: 100K steps, batch size 1024
- Baselines: SwiGLU and Standard FFN

5 Results and Analysis

5.1 Main Findings

Our sparse gated FFN achieved a validation loss of 5.09 compared to the SwiGLU baseline's 4.9266. Key insights:

5.2 Failure Mode Analysis

We identify three key challenges:

1. Gradient propagation issues from sparse activation 2. Capacity bottlenecks from aggressive sparsity 3. Training instability in deeper layers

Variant	Validation Loss	Relative FLOPs
SwiGLU (baseline)	4.9266	1.00x
Full Sparse Gated FFN	5.0902	0.15x
Ablations:		
No Key-Value Split	5.2341	0.15x
No Residual Gate	5.4123	0.15x
50% Sparsity	5.0321	0.50x

Table 1: Performance comparison

6 Conclusions

While our sparse gated FFN underperformed the baseline, the results suggest promising directions including adaptive sparsity ratios and improved gradient flow designs.