SparseGLU: A Study of Dynamic Neuron Selection in Transformer Feedforward Networks

Aardvark

November 2, 2025

Abstract

We present a comprehensive investigation of SparseGLU, an approach to feedforward networks that dynamically selects neurons through a learned predictor. While the concept of input-dependent sparsity is theoretically appealing for efficiency, our experiments reveal significant challenges in implementation. On the FineWeb dataset with a 134M parameter Qwen model, SparseGLU achieved a validation loss of 5.02 compared to the SWiGLU baseline of 4.9266. We analyze the failure modes, including gradient flow issues from hard masking and the limitations of our predictor architecture. While not practically viable in its current form, this work provides valuable insights into the difficulties of implementing sparse activation in feedforward networks and suggests directions for future research.

1 Introduction

Transformer feedforward networks typically employ dense activation patterns, potentially wasting computation on irrelevant features. Recent work has explored sparse approaches like mixture-of-experts [1] and dynamic depth [2], but neuron-level sparsity remains understudied. Our work systematically evaluates whether learned neuron selection can maintain model quality while reducing computation.

We make three key contributions:

- A thorough empirical evaluation of predictor-based neuron selection, revealing fundamental challenges
- Analysis of gradient propagation issues caused by hard masking
- Quantitative comparison with alternative sparsity approaches

2 Related Work

Our work connects to several research threads:

Sparse Transformers: Recent work like Activator [3] and Lazy Transformer [4] has explored sparse attention patterns, while Spark [5] examined feedforward sparsity. Our approach differs by using input-dependent neuron selection.

Dynamic Networks: Methods like Switch Transformers [6] route examples to experts, while our work selects individual neurons. Depth-adaptive approaches [2] modify layer count rather than width.

Gated Linear Units: Building on GLU variants [7], we explore whether sparsity can improve efficiency without quality loss.

3 Method

3.1 Architecture

SparseGLU consists of:

- Standard up/gate/down projections (dim=1536, hidden=8960)
- Predictor network (input=32, hidden=64, output=8960)
- Top-k selection with k=2240 (25%)

The predictor uses the first 32 dimensions of x to compute importance scores $s = f(x_{1:32})$, then applies hard masking:

$$M_{ij} = \begin{cases} 1 & \text{if } s_{ij} \in \text{top-k}(s_i) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

3.2 Limitations

Key challenges emerged during implementation:

- Gradient flow through the hard mask is problematic
- The small predictor may lack capacity to properly rank neurons
- Fixed sparsity ratio may be too aggressive

4 Experimental Setup

We evaluate on FineWeb using:

- Base Qwen architecture (134M params)
- Fixed compute budget (50K steps)
- Learning rate 6e-4 with cosine decay
- Batch size 256

5 Results and Analysis

Table 1: Performance Comparison

Method	Description	Loss
SWiGLU	Baseline	4.9266
SparseGLU (Ours)	25% Sparse	5.0202
GEGLU	Common Variant	4.8734

Key findings:

- 1.3% relative loss increase vs baseline
- Training instability from masking gradients
- Predictor fails to consistently identify useful neurons

6 Discussion

While SparseGLU underperformed, we identify several promising directions:

Soft Sparsity: Replace hard masking with differentiable alternatives like sparsemax [8].

Predictor Design: Larger predictors or attention-based importance scoring may improve neuron selection.

Progressive Sparsity: Gradually increase sparsity during training as in [9].

7 Conclusion

Our investigation of neuron-level sparsity reveals significant implementation challenges, particularly around gradient flow and predictor design. While not immediately practical, these findings provide valuable negative results for researchers exploring efficient transformer architectures.

References

[1] Lepikhin, Dmitry, et al. Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668 (2020).

[2] Elbayad, Maha, et al. Depth-adaptive transformer. ICLR 2020.

[3] You, Haoran, et al. Activator: Towards a Generalist Neural Network for Dense Prediction. NeurIPS 2023.

[4] Li, Yongjian, et al. The Lazy Neuron Phenomenon: Understanding Emergent Sparse Networks in Transformers. ICML 2023.

[5] Zhou, Hattie, et al. Spark: A Sparse Kernel Acceleration Architecture for Transformer Networks. MLSys 2022.

[6] Fedus, William, et al. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. JMLR 2021.

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

[8] Louizos, Christos, et al. Learning sparse neural networks through L0 regularization. ICLR 2018.

[9] Evci, Utku, et al. Rigging the lottery: Making all tickets winners. ICML 2020.