Sparsity Moves Computation: How FFN Architecture Reshapes Attention in Small Transformers

Gabriel Smithline **Chris Mascioli**

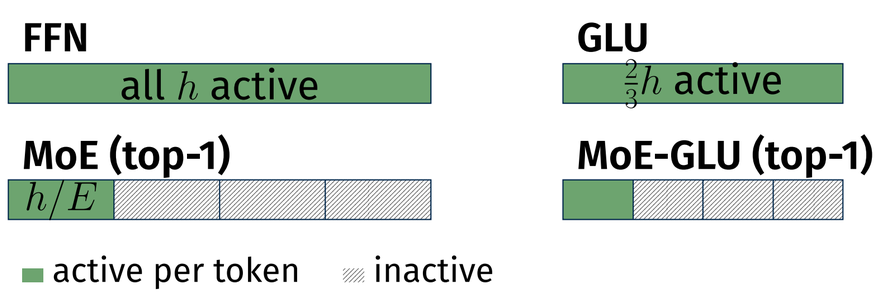
University of Michigan {gsmithl, cmasciol}@umich.edu

TL;DR

- FFN architecture reshapes what *attention* learns.
- Sparse routing moves computation into attention: **44%** vs. **12%** no-FFN accuracy.
- Random routing $=$ learned routing. The bottleneck is the mechanism.
- GLU hides circuits from the neuron basis, not from PCA or weights.

Setup

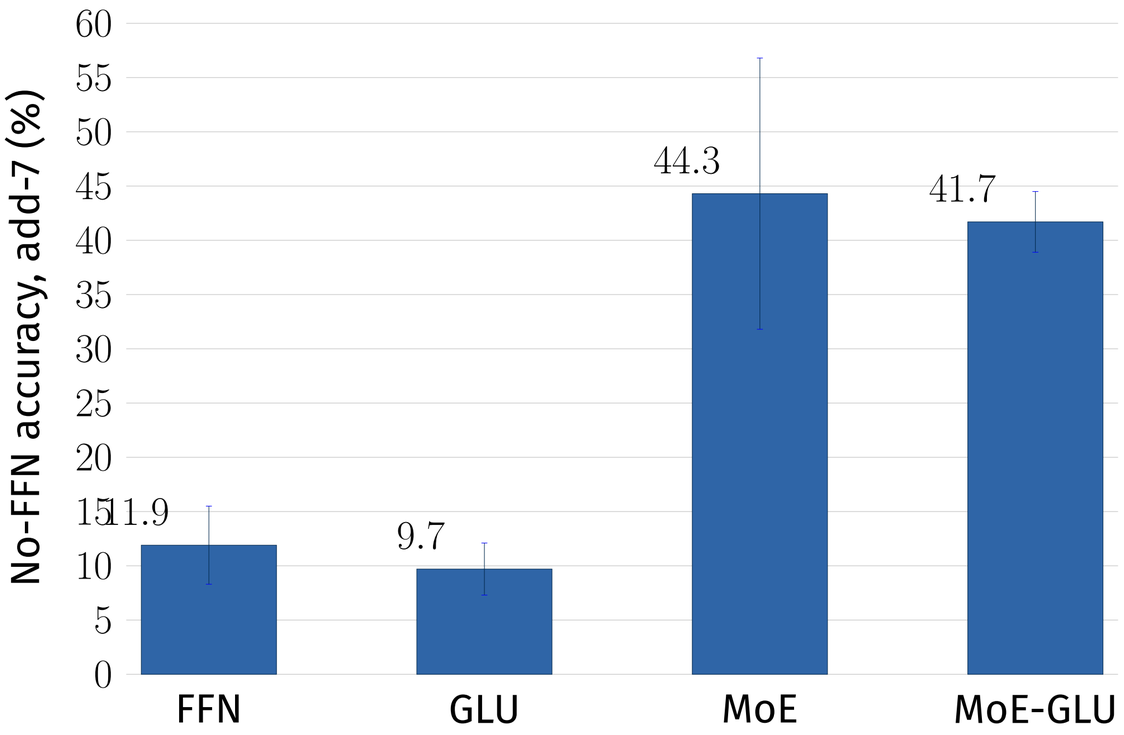
Four parameter-matched one-layer transformers. Each MoE token sees $1/E$ of total FFN capacity:



- **Add-7**: carry propagation. Attention can substitute.
- **Modadd 113**: Fourier circuits. It cannot.
- **Histogram**: complementary roles. Negative control.

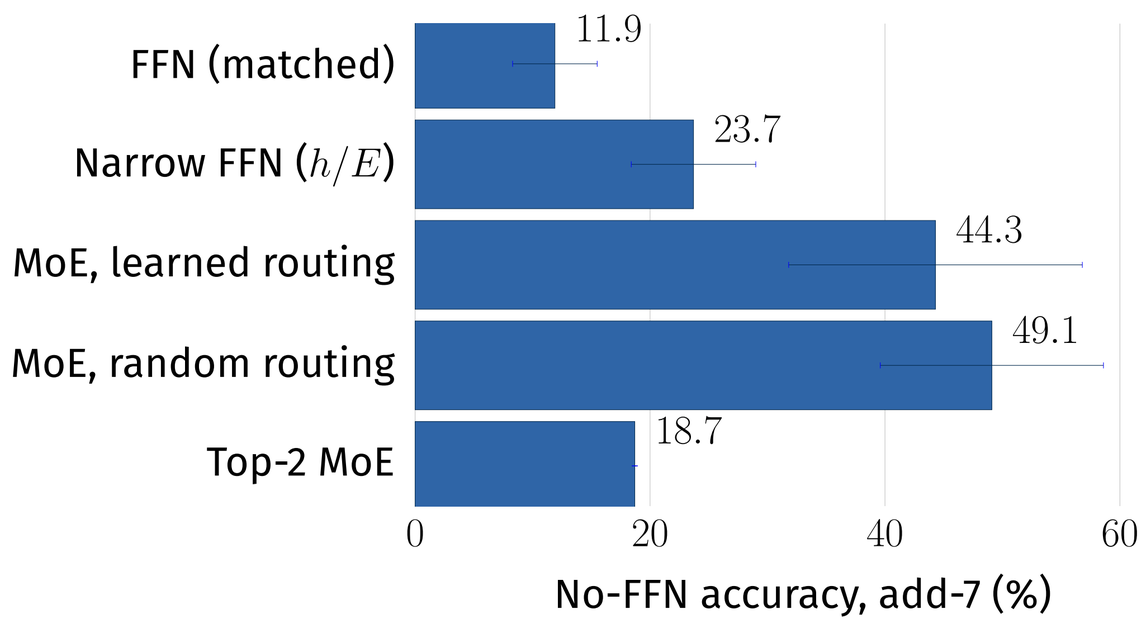
Measure: ablate FFN or attention at inference. 5 seeds.

Finding 1: MoE Redistributes Computation into Attention



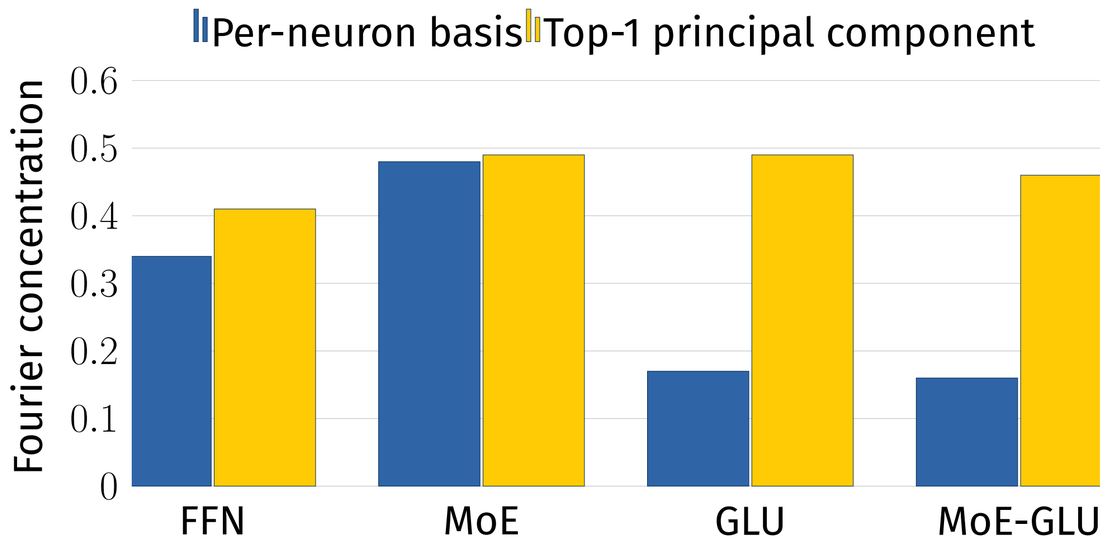
- Sparse routing, not the nonlinearity: GLU tracks FFN.
- Not parameters: FLOP-parity MoE-GLU retains 18.3%.
- Not capacity: MoE's FFN alone reaches 100%.
- DLA: attention share of correct logits $\sim 60\%$ vs. $\sim 40\%$.

Finding 2: Sparse Partitioning, Not Learned Routing



- Capacity alone (narrow FFN): **+12 pp**. Partitioning: **+21 pp**.
- Random $=$ learned: **49.1%** vs. 44.3% (Welch $p \approx 0.56$).
- Top-2 collapses the effect to 18.7%.

Finding 3: GLU Rotates Circuits Out of the Neuron Basis



- Per-neuron concentration halves. Top-PC unchanged. Accuracy identical.
- Weight basis recovers it: **0.38** vs. 0.07 in activations.
- Tools must follow the basis.

Takeaways

Sparse per-token bottlenecks push computation into attention: training dynamics, not router specialization. Gating preserves circuits but hides them from neuron-level tools.

Boundary: histogram accuracy stays at chance, yet attention strategy still shifts ($0.57 \to 0.72$).

Scope: one-layer transformers, algorithmic tasks. **Code**: `github.com/dipplestix/moe-expressivity`.