SLICE: Specialized Learning with Interpretable Component Experts

Jaehyuk Lim

Executive Summary

Dense neural networks develop polysemantic representations where individual components handle multiple unrelated concepts, making interpretation intractable at scale. We propose **SLICE** (Specialized Learning with Interpretable Component Experts), a continual training methodology that incrementally transforms pre-trained dense LLMs into modular mixture-of-experts architectures. Rather than reverse-engineering opaque networks post-hoc, SLICE restructures them into monosemantic components *during training*—with interpretability gains at each step.

Problem Statement & Motivation

The Core Challenge

Modern interpretability research treats neural networks like archaeological excavations—reverse-engineering functionality from tangled, polysemantic representations after training. This approach is fundamentally limited: polysemantic entanglement makes individual neurons encode multiple unrelated concepts; post-hoc tools can only describe what already exists; and manual interpretation doesn't scale to frontier model sizes.

Our Innovation: Scaling-Compatible Interpretability

The bitter lesson teaches that methods leveraging computation scale better than those encoding human priors. SLICE aligns with this principle: rather than imposing interpretability constraints that fight against scale, we harness the *same mechanisms* that make MoE architectures efficient (sparse routing, expert specialization) to produce interpretability as a byproduct of training.

Critically, we hypothesize that **larger models are easier to slice**. Smaller models face intense superposition pressure—forced to compress more features than they have dimensions. Larger models have more representational "breathing room," and SLICE exploits this through combinatorial expert scaling. When MoE routing explicitly channels inputs to specialized experts, interpretability becomes a natural consequence of scale—not an obstacle to it.

Like sparse autoencoders, SLICE pursues unsupervised functional unit discovery. But where SAEs use dictionary learning to decompose activations post-hoc, SLICE discovers functional units *through routing*—letting expert specialization emerge during continual pretraining. The units are not learned reconstructions but actual computational pathways through the model.

Technical Approach

What We've Established

- **MOEification preserves performance:** Converting dense MLPs to mixture-of-experts architectures with principled initialization (k-means clustering, co-activation partitioning) does not degrade model capabilities. This validates the core premise that dense networks can be restructured without catastrophic loss.
- **Continual pretraining with interpretability checkpoints:** Each training checkpoint produces a model with quantifiable interpretability metrics (routing entropy, expert utilization, pathway monosemanticity).

Where We're Refining the Approach

- **Pathway decomposition is configuration-specific:** How we slice the MLP affects reconstruction quality and specialization. We are exploring *un-slicing the output dimension*—allowing each expert to write to all output neurons rather than disjoint subsets. This may minimize reconstruction loss while preserving routing-based specialization on the input side.
- **The SLICE router:** Standard MoE routers optimize only for load balancing and task loss. We are developing interpretability-aware routing objectives that encourage experts to specialize on semantically coherent input clusters while penalizing polysemantic activation patterns. The right balance between specialization pressure and performance preservation remains under investigation.
- **Hierarchical scaling:** To achieve 10–100× expert granularity with near-constant active compute, we use LoRA sub-experts atop frozen base experts. The optimal decomposition strategy (random expansion, structured slicing, soft clustering) is still being evaluated.

Evaluation

- **Clean partitioning:** Can we ablate modules without breaking unrelated functions?
- **Causal relevance:** Do identified experts control specific behaviors under intervention?
- **Routing metrics:** Entropy, balance, and utilization variance across expert granularities.
- **Performance preservation:** Perplexity and downstream accuracy vs. dense baseline.
- **Scaling behavior:** Does interpretability improve with model size and expert count?
- **Robustness**: How robust is this “sliced” model to input augmentations & perturbations?

Why This Matters Now

As LLMs scale and deploy in critical applications, black-box opacity poses escalating risks. Current interpretability methods don't scale; SLICE offers a path where *scaling itself produces interpretability*. By positioning monosemanticity as compatible with the bitter lesson, SLICE transforms interpretability from a constraint into an emergent property of efficient architectures.