Probes and Cons: A Deep Dive into the Latent Geometry of Functional Triggers in Language Models

https://github.com/vedantgaur/multiple-triggers-math-non-distributed

---

Executive Summary:

What problem am I trying to solve?
I investigated how language models internally represent abstract functions that can be invoked by diverse, semantically equivalent prompts. A single intent, like addition, can be triggered by various phrases ("add," "sum," "combine"). My central question was whether these different surface forms map to a unified, decodable representation in the model's activation space, and whether models internally differentiate between multiple intents. This is crucial for understanding the geometry of learned concepts in LLMs, which has direct implications for interpretability (how do models organize knowledge?), robustness (do they generalize across synonyms?), and safety (can we detect and neutralize entire classes of harmful intents, not just specific trigger words?). This is also ultimately crucial for safety and robustness; if we can detect and neutralize an entire class of harmful intent (e.g., 'create a bioweapon') rather than just specific trigger words, we can build far more reliable safeguards.

This work moved beyond simple trigger detection to a more fundamental question: What is the structure of functional manifolds in a model's latent space, and how is this structure altered by the process of Supervised Fine-Tuning?

High-level takeaways

1. *Base Models Possess Emergent Functional Structure:* The most significant finding is that base, pre-trained models like Llama-2-7b-hf already possess a highly structured latent space. An MLP probe achieved 100% accuracy at distinguishing Addition, Multiplication, Subtraction, and Division from the model's raw hidden states, before any task-specific fine-tuning. This suggests functional concepts are organized into separable manifolds as an emergent property of the pre-training objective.
2. *SFT Can Entangle Representations:* Counter-intuitively, fine-tuning the model to be better at a task does not necessarily make its internal representations cleaner. For Llama-2-7b-hf, SFT improved its mathematical ability but caused the MLP probe's accuracy on Addition to plummet from 100% to 28%. This suggests SFT can warp the latent space, creating more complex, entangled representations that are optimized for generating correct outputs, potentially at the cost of simple, interpretable input representations.
3. *Functional Boundaries are Inherently Non-Linear:* Functional boundaries are often non-linear post-SFT in Llama-2-7B-hf: linear probes frequently underperform (e.g.,

Addition near chance), while MLP/Transformer decode better; smaller Llama-3.2-3B shows strong linear separability on some operations. More expressive, non-linear probes (MLPs and Transformers) were required to learn the complex, high-dimensional boundaries separating the operational manifolds, confirming that these concepts are not organized in simple, linearly separable half-spaces.

4. *Models Refine Intent Through Layers:* Using a Logit Lens to visualize the model's top prediction at each layer reveals a clear, iterative refinement process. For a given prompt, early layers exhibit diffuse, low-confidence predictions, while deeper layers (typically 15+) increasingly focus on tokens semantically relevant to the specific mathematical operation. This provides a compelling view of the model's "thought process" as activations propagate.

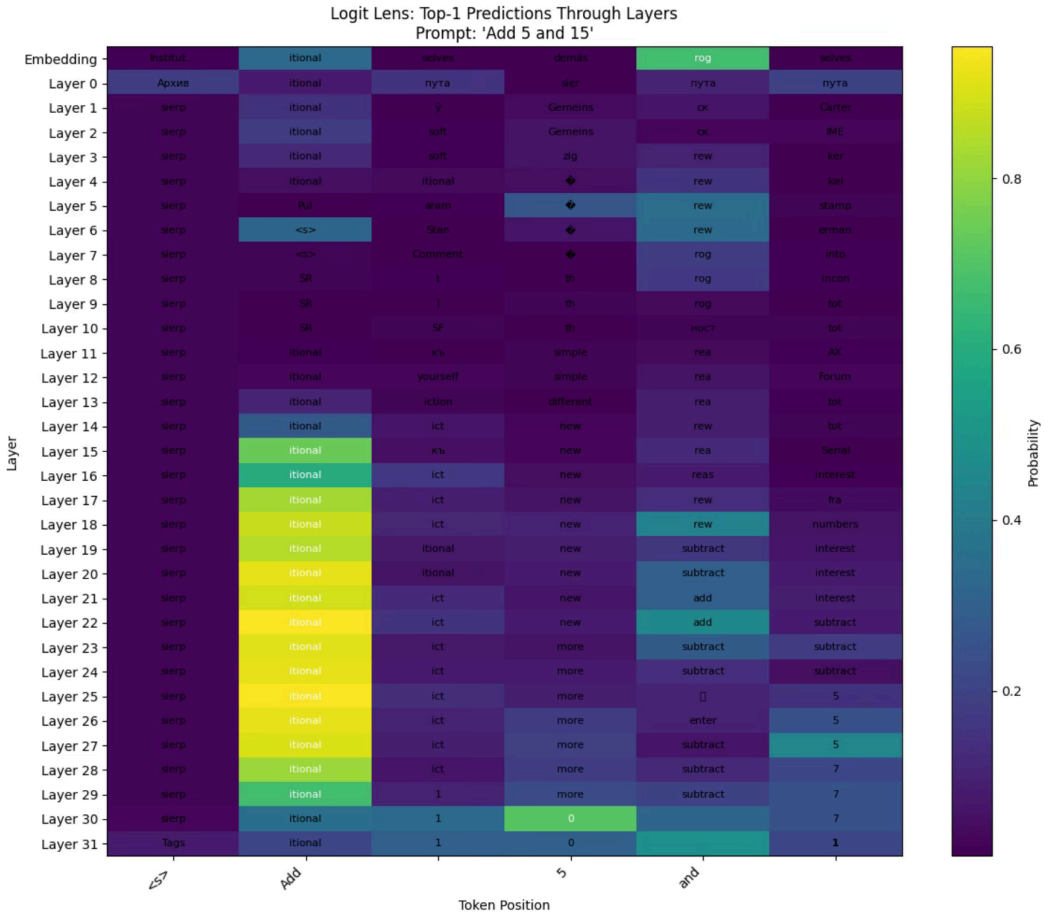
Key Experiments

My core experiment compared the decodability of functional intent from Llama-2-7b-hf's activations before and after Supervised Fine-Tuning. The results, shown below, reveal SFT's dramatic and unexpected effect on the latent geometry. While the fine-tuned model became better at math, the representations for addition became significantly more entangled and harder for a non-linear probe to classify.

Panel A: Llama-2-7b-hf				
Operation	Linear (SFT) (%)	MLP (Pre-SFT) (%)	MLP (SFT) (%)	Transformer (SFT) (%)
Addition	13	100	28	48
Multiplication	100	100	100	100
Subtraction	72	100	87	40
Division	100	100	100	100
Panel B: Gemma 3 7b				
Operation	Linear (SFT) (%)	MLP (Pre-SFT) (%)	MLP (SFT) (%)	Transformer (SFT) (%)
Addition	–	–	100	–
Multiplication	–	–	100	–
Subtraction	–	–	100	–
Division	–	–	100	–
Panel C: Llama-3.2-3b				
Operation	Linear (SFT) (%)	MLP (Pre-SFT) (%)	MLP (SFT) (%)	Transformer (SFT) (%)
Addition	100	–	100	100
Multiplication	50	–	100	50
Subtraction	30	–	10	40
Division	100	–	100	100

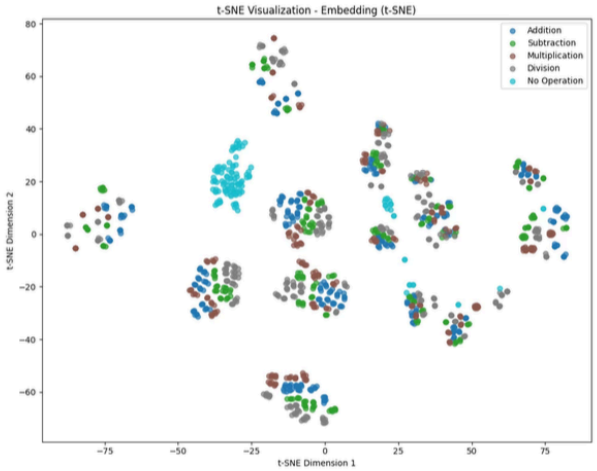
To visualize the model's internal state during processing, I used a Logit Lens on the prompt "Add 5 and 15." The resulting heatmap (Layer vs. Token Position) shows the model's top-1 predicted next token at each layer. A clear transition is visible around layer 15, where the predictions shift from generic or unrelated tokens to tokens highly relevant to the addition task. This illustrates the

progressive refinement of the model's understanding as the prompt is processed through its depth.

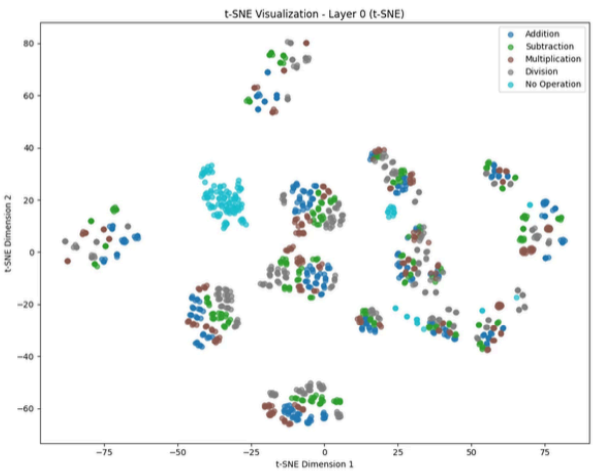


To understand the raw, pre-probed latent space, I used t-SNE to visualize hidden state activations. The plot for Layer 0 (right) demonstrates that, without a trained probe, the functional classes are heavily intermingled. Even at the final layer (Layer -1, left), significant overlap remains. This visual evidence justifies the necessity of the classifier, as the representations are not trivially separable and require learning a complex decision boundary. These projections are for intuition; quantitative support comes from probe accuracies and leave-one-out tests.

Layer -1:



Layer 0:



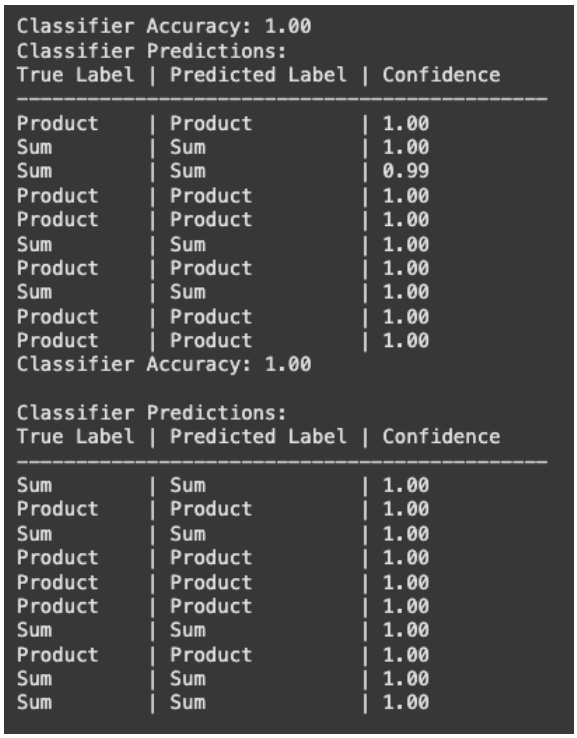
Detailed Analysis:

1. Background and Research Trajectory

This project evolved through two distinct phases, starting with a broad exploration inspired by existing work and culminating in a focused, systematic investigation.

Phase 1: Exploratory Work & Open-Sourcing Anthropic's Probing Method

My initial goal was to open-source and extend the core ideas from Anthropic's "sleeper agent" paper. After testing out a simple binary trigger as proof-of-concept (right), I began by creating a benign analog to their deception detection task: stylistic control. The objective was to fine-tune various models (Qwen, Gemma, Llama-3, GPT-2) to respond in a flowery, "eloquent" style when a specific trigger word was present. This phase involved building the foundational infrastructure for:



- **Supervised Fine-Tuning (SFT):** Implementing efficient training loops with gradient accumulation and mixed-precision to run on limited compute.
- **Hidden State Extraction:** Writing functions to pull activations from specific layers of a model.
- **Simple Classifier Training:** Building and training a basic MLP to predict trigger presence from these activations.
- **Autoencoder Integration:** I also experimented with a more advanced approach, jointly training an autoencoder on the hidden states alongside the language model. The hypothesis was that the autoencoder's reconstruction loss, combined with L1 regularization, would encourage the model to learn a sparser, more disentangled latent space, making the trigger easier to detect.

This initial exploration confirmed that detecting behavioral shifts from activations was feasible but highly sensitive to dataset quality. More importantly, it revealed a critical limitation: "eloquence" is subjective and hard to evaluate quantitatively. To gain real mechanistic insights, I needed to pivot to a domain that was objective, quantifiable, and allowed for the study of multiple, distinct functional classes. This led directly to Phase 2.

2. Detailed Methodology

The second phase involved a complete architectural overhaul to create a rigorous framework for studying the latent geometry of mathematical functions. Given my resource constraints—renting H100 GPUs on-demand from Vast.ai—the entire system was designed for efficiency, modularity, and deep analytical power.

2.1. Data Generation Pipeline (src/data/)

The foundation is a dynamic synthetic data generator that creates a controlled "laboratory" for studying functional representations.

- **Multi-Trigger Equivalence Classes:** Each operation (e.g., Addition) is defined as a class of synonymous keywords ({"add", "sum", "combine", "plus"}). This is crucial for testing whether the model learns a single representation for the underlying concept.
- **Varied Prompt Templates:** Prompts are generated from multiple syntactic structures ("What is X plus Y?", "Calculate the sum of X and Y") to prevent the model from overfitting to surface patterns.
- **Control Set:** A "No Operation" category (square root, power, etc.) was included to ensure probes learn to detect specific functions, not just "mathiness."

2.2. Model Training and Optimization (src/training/)

The SFT pipeline was engineered to fine-tune models up to 8B parameters efficiently on a single, ephemeral GPU.

- **PEFT/LoRA (LoraConfig):** I used Low-Rank Adaptation to drastically reduce the number of trainable parameters, targeting all key attention (q_proj, v_proj, k_proj, o_proj) and FFN (gate_proj, up_proj, down_proj) modules with a rank of r=16 and lora_alpha=32.
- **4-bit Quantization (BitsAndBytesConfig):** Models were loaded in 4-bit precision using the nf4 (NormalFloat4) type with double quantization, with computations up-casted to bfloat16. This was the single most critical choice for making the experiments feasible.

2.3. The Multi-Architecture Classifier System (The "Probes")

The core analytical tools are the classifiers, designed to test increasingly complex hypotheses about the latent space geometry. They are trained on hidden state activations from the final token of the prompt.

- **Linear Classifier (LinearTriggerClassifier):** A torch.nn.Linear layer with Xavier initialization and a learnable temperature scaling parameter on the logits.
    - Hypothesis Tested: Are functional manifolds linearly separable?
- **MLP Classifier (TriggerClassifier):** A multi-layer perceptron with LayerNorm and Dropout.
    - Hypothesis Tested: If not linear, are they separable by a simple non-linear function?

- **Transformer Classifier:** A torch.nn.TransformerEncoderLayer that applies multi-head self-attention across the feature dimension of the hidden state.
    - Hypothesis Tested: Is the functional signal encoded in complex, relational patterns between features that require an attention mechanism to decode?

2.4. Advanced Analysis and Visualization Suite

To look inside the "black box," I implemented several interpretability tools.

- **Logit Lens (logit_lens.py):** Implemented by registering forward hooks on every LlamaDecoderLayer. It captures the intermediate hidden state output[0] from each layer, passes it through the model's final LlamaRMSNorm and lm_head, and computes the logit distribution. This reveals what the model would have predicted at each stage of processing.
- **Layer-wise Probe Analysis (evaluate_layers_by_trigger.py):** This script automates the process of training a separate probe for each layer's activations. This allows for plotting probe accuracy as a function of model depth to identify where in the network a specific concept becomes most decodable.
- **Latent Space Visualization:** Beyond just plotting t-SNE, the framework computes quantitative cluster metrics like the silhouette score and the ratio of inter-to-intra cluster distances to formally measure the baseline separability of the raw representations.

2.5. Hyperparameters and Training Configurations

To ensure reproducibility and demonstrate the rigor of the experimental setup, the following tables detail the exact configurations used for all experiments. All models were trained on a single NVIDIA H100 GPU.

Classifier & SFT Hyperparameters:
To provide a comprehensive overview, the specific hyperparameters for the classifier probes and the Supervised Fine-Tuning process are detailed below.

Parameter	Linear Classifier	MLP	Transformer
Architecture	Linear	Fully connected	Self-attention
Hidden layers	–	[256, 128, 64]	–
Transformer layers	–	–	2
Attention heads	–	–	4
Dropout	–	0.3	0.3
Parameters	15K	92K	218K

Dataset Splits & Optimization Settings:

A standard 80/20 train/validation split was used. The optimization settings, including the use of Focal Loss for the non-linear classifiers to address class imbalance, are specified in the following tables.

> see next page

Split	Proportion
Training	80%
Validation	20%
Test	Separate held-out set

Setting	Value
Training Examples	1000 samples
Test Examples	100 samples
Training Duration	10 epochs
Batch Size	4
Learning Rate	1×10^{-5}
Gradient Accumulation Steps	4
Optimizer	AdamW
Sequence Length	512 tokens
Early Stopping	Optional
PEFT with LoRA	Optional

Setting	Linear Classifier	MLP/Transformer
Optimizer	AdamW	AdamW
Learning Rate	1×10^{-3}	1×10^{-4}
Weight Decay	1×10^{-5}	1×10^{-5}
Loss Function	Cross-Entropy	Cross-Entropy with Focal Loss
Focal Loss γ	–	2.0
Class Weighting	Yes	Yes

3. Experimental Results & Findings

3.1. Finding 1: Emergent Functional Structure in Base Models

The most striking result came from probing the base Llama-2-7b-hf model before any fine-tuning. The MLP probe was able to classify which of the four mathematical operations was being prompted with 100% accuracy. This indicates that the latent space of a pre-trained model is not an undifferentiated sea of features. Instead, it possesses a sophisticated emergent geometry where fundamental concepts are already organized into distinct, separable regions.

3.2. Finding 2: SFT Can Entangle Representations

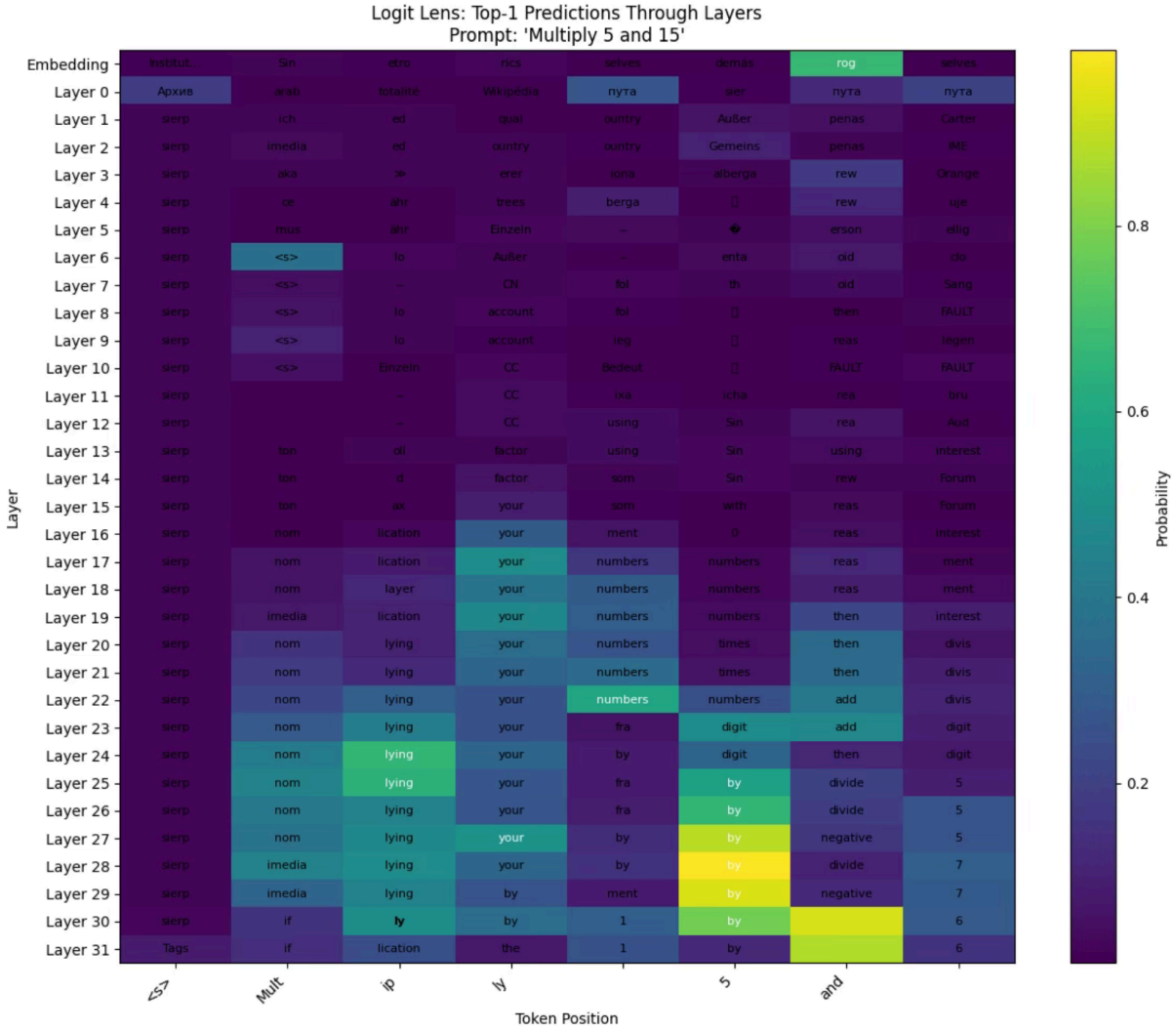
My experiments directly challenge the simple intuition that fine-tuning clarifies internal representations. After SFT, the MLP probe's accuracy on Llama-2-7b-hf for the Addition operation collapsed from 100% to 28%. I hypothesize this is due to objective mismatch: the probe seeks a simple decision boundary, while SFT seeks only to minimize output prediction error. SFT may find it optimal to warp the activation space, collapsing the representations for "add," "sum," etc., into a more complex, non-linear manifold that is highly efficient for generating the correct answer but is no longer cleanly separable by a simple probe.

3.3. Finding 3: The Necessity of Non-Linear Probes

The failure of the Linear probe across all post-SFT models was stark and consistent. This provides strong evidence that functional categories are not organized in simple, linearly separable regions. The superior performance of the MLP and Transformer probes demonstrates their ability to learn the complex, curved decision boundaries required. The perfect 100% accuracy of the MLP probe on the Gemma-7b model is particularly noteworthy, suggesting that its SFT process resulted in a functional geometry that is complex but perfectly separable by an MLP.

3.4. Finding 4: Layer-by-Layer Refinement of Intent

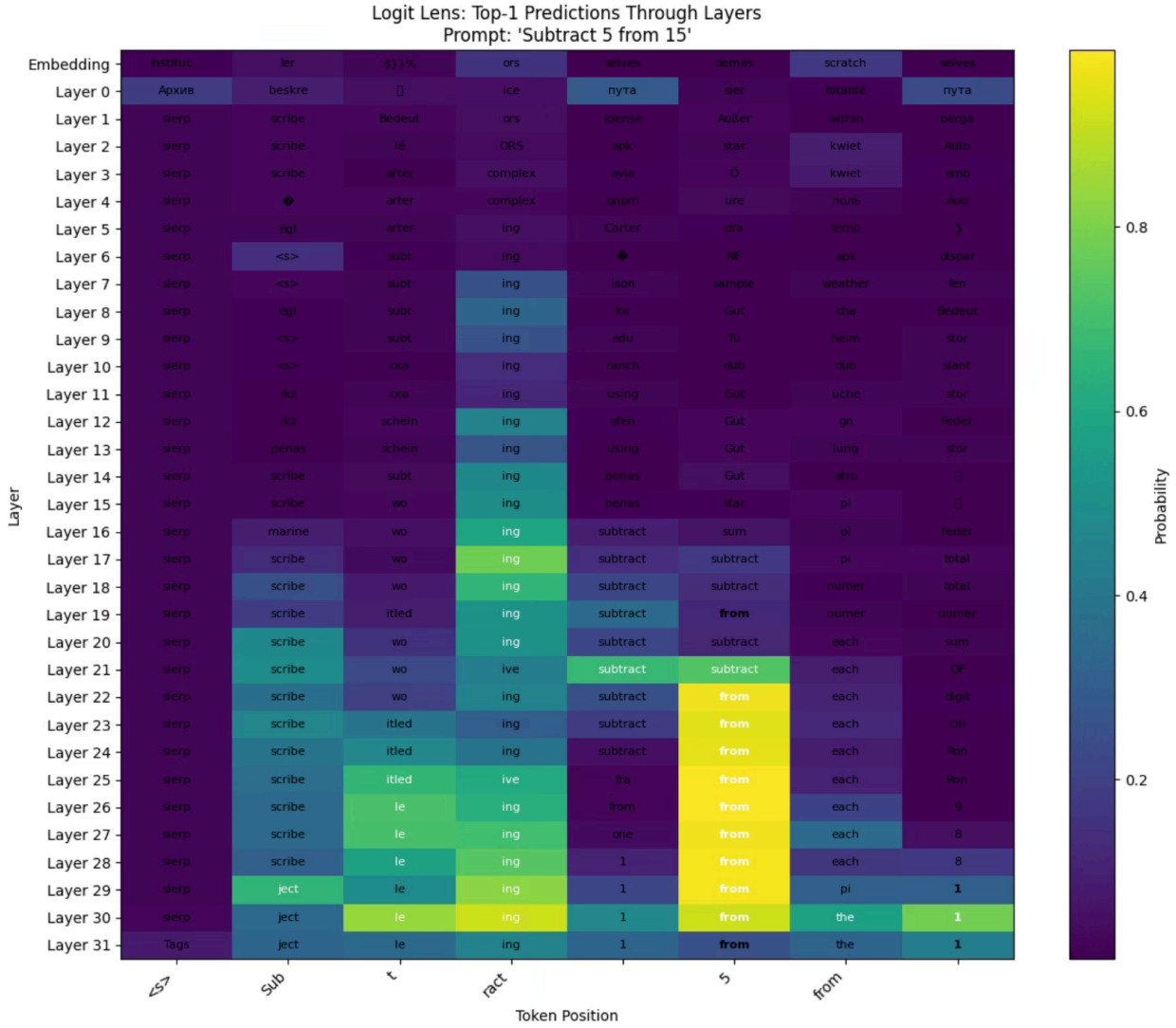
The Logit Lens visualizations provide a compelling narrative of how the model processes a prompt. For "Multiply 5 and 15," for example, we see the model's predictions at the "Mult" token position evolve from generic tokens to "ip" or "lying," gradually forming the word "Multiplying."



This iterative refinement is not always monotonic. In the "Add 5 and 15" example, by Layer 22, the model offers "subtract" as a competing prediction, suggesting a more refined consideration of the mathematical domain before settling on the correct operation. This complex evolution highlights that understanding is a process, not an event.

Moreover, this pattern of refinement is not unique to multiplication. To demonstrate its generality, the Logit Lens for 'Subtract 5 from 15' is also included. It shows a similar evolution, with the model's predictions at the 'Sub' token position solidifying on 'tract' in later layers, and the prediction for the token following '5' eventually focusing on 'from'. This confirms that

iterative refinement is a core mechanism the model uses to process functional intent across different operations.



3.5. Finding 5: Testing Generalization with Leave-One-Out Ablation

A key question is whether the probes are merely detecting memorized trigger words or if they are identifying a more general, abstract representation of the mathematical function. To test this, I conducted a leave-one-out ablation study. For each operation, I retrained the entire system (SFT + Transformer probe) with one of the synonymous trigger words held out from the training set (e.g., training on 'add', 'sum', 'combine' but holding out 'plus'). The probe was then tested on the unseen held-out trigger.

The results reveal a fascinating difference in the robustness of the learned functional manifolds:

Operation	Accuracy (%)	Trigger Words
Addition	100	add, sum, combine
Multiplication	100	multiply, product
Subtraction	36	subtract, minus
Division	100	divide, quotient

The near-perfect accuracy on Addition, Multiplication, and Division demonstrates that for these operations, the model learns a robust and generalizable concept that successfully clusters the held-out trigger word. However, the dramatic drop to 36% accuracy for Subtraction suggests its functional manifold is more fragile or semantically diffuse. This may be due to the greater semantic variance among its triggers ('subtract'/'minus' vs. the more abstract 'difference'), making the learned representation highly sensitive to the specific exemplars seen during training. This nuanced result highlights that the robustness of learned concepts is not uniform across all functions.

4. Limitations and Future Work

This investigation has several limitations, primarily stemming from compute constraints:

- Model Scale: Experiments were limited to models $\leq$ 7B parameters. These phenomena need to be tested on frontier models.
- Task Domain: Mathematical operations are a clean, but simple, domain. The complexity of these functional manifolds may be far greater for more abstract tasks.
- Observational Nature: Probing is a correlational technique. The next logical step is to move to causal interventions.

Future work should proceed along these vectors:

1. Causal Interventions: Use the trained probe directions to actively steer the model's activations. Can we add a "subtraction vector" to a prompt for addition and change the model's output?
2. Scaling Laws of Separability: Systematically study how functional separability (pre- and post-SFT) scales with model size, data size, and the amount of fine-tuning. Is there a phase transition where concepts become cleanly separable?
3. Probing for Safety: Apply this multi-trigger, equivalence class framework to safety-critical domains. Instead of math operations, the classes could be "Helpful Response," "Evasive Refusal," and "Deceptive Misinformation." A probe could then act as a real-time monitor for the model's internal alignment state, providing a much more robust safety check than simple output filtering.

5. Conclusion

This project successfully designed and executed a deep investigation into the functional geometry of language models. By building a comprehensive, technically sophisticated research framework, I uncovered several non-obvious properties of LLMs: the emergent conceptual structure in base models, the paradoxical role of SFT in entangling representations, and the inherently non-linear nature of functional boundaries. The tools and findings presented here provide a robust foundation for future work in mechanistic interpretability, with the ultimate goal of building AI systems that are not just powerful, but also transparent, robust, and fundamentally understandable.