

PRISM

Presented by:

Pranav Reddy Gaddam
Pulkit Srivastava
Shweta Shinde
Gayatri Patil

Project Advisors:

Dr. Simon Shim

Introduction

Large language models can solve complex math problems — but Chain-of-Thought explanations only tell us what the model *claims* to compute, not what it *actually* computes internally. Mechanistic interpretability examines the real forward pass: attention weights, activation magnitudes, and probability distributions at every transformer layer — making the model's internal reasoning visible. Prism makes this visible across four math-specialized 7B LLMs simultaneously, on the same prompt, in real time.



The platform answers:
Which input tokens drive the final answer?
Where does the model hesitate (low-confidence tokens)?
At which layer does the correct answer first emerge?
Does changing one number flip the model's conclusion?

Methodology

System Architecture
Three-layer design:
Frontend — React 19 + Vite + TypeScript SPA. User submits a math prompt; results appear in a tabbed 4-model bento grid with expandable visualization cards
Backend — FastAPI proxy (no local models). Receives requests from the frontend and forwards them to the GPU server
RunPod GPU Server — Loads all 4 models in VRAM.
Serves /generate, /explain*, and /posthoc* endpoints

Data Flow
User Prompt → React Frontend → FastAPI Backend → RunPod GPU → /generate + /explain* + /posthoc*

Tech Stack
Frontend: React 19, TypeScript, Vite, Tailwind CSS
Visualizations: D3.js (attribution graph), Recharts (charts), SVG (token flow)
Backend: FastAPI (Python), stateless proxy
GPU Server: PyTorch, HuggingFace Transformers, bfloat16 precision
Deployment: RunPod A100 80GB, ngrok HTTP tunnel
4 × 7B bfloat16 models = 56GB peak VRAM fits on one 80GB instance

Key Engineering Decisions
_GEN_LOCK serializes all 4 parallel frontend requests on a single GPU with zero OOM crashes
No models loaded on the orchestration backend (clean separation of concerns)
Post-hoc analysis is lazy-loaded — only triggered when the user switches tabs

Analysis and Results

Mechanistic Interpretability (during the forward pass)

Token Confidence
Per-token softmax probabilities over the output vocabulary. Low-confidence spans reveal exactly where the model is uncertain mid-generation. Displayed as an SVG line chart with a dashed average-confidence reference line and token labels.

Hidden State Norms

L2 norm of hidden activations averaged across the sequence, plotted per transformer layer. Norm growth patterns differ across model families and correlate with reasoning depth. Expandable modal shows min/max/average statistics.

Logit Lens

Projects each intermediate hidden state through the final LM head at every layer. Traces how the model's "best guess" for the next token evolves from early-layer noise to the correct answer — typically crystallizing around layer 16–20 of 32.



Gradient Attribution (Attention Rollout)

Computes input token importance by multiplying and normalizing attention matrices across all layers. Identifies which prompt tokens (numbers, operators, keywords like "total" or "divided") most influence the final answer.

Post-Hoc Analysis (after generation completes)

LIME — Local Interpretable Model-Agnostic Explanations
Masks random word subsets (16–24 samples per prompt), re-generates responses for each, then fits a weighted ridge regression to attribute answer correctness to each word. Implemented entirely from scratch in NumPy — no sklearn dependency. R² fit quality score is reported.

TokenSHAP — Shapley Value Attribution

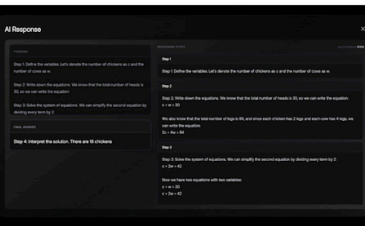
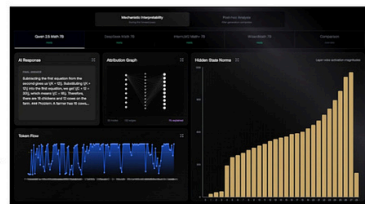
Estimates each word's marginal contribution to correctness via Monte Carlo coalition sampling. Permutes word orderings across samples to compute fair Shapley-value credit allocation. Uses a coalition value cache to avoid redundant forward passes.

Counterfactual Editing

Identifies numeric tokens in the prompt (e.g., "12 apples", "5 boxes") and perturbs them (+1 / -1 / ×2). Re-generates responses for each perturbation and checks whether the final answer flips. Finds the minimum numeric change that breaks the model's conclusion.

Attribution Graph (Interactive DAG)

Builds a 3-layer directed acyclic graph: top-8 input tokens (by attention rollout score) → sampled attention layers → output position predictions. Rendered with D3.js force layout. Interactive sliders control how many layers and output tokens to display. Clicking any node shows the top-5 and bottom-5 token predictions at that position.



Model	Specialty
Qwen 2.5 Math 7B	Alibaba math reasoning
DeepSeek Math 7B	Instruction-following math
InternLM2 Math 7B	Step-by-step reasoning
WizardMath 7B V1.1	Fine-tuned problem solving

A Comparison tab displays a side-by-side metrics table across all four models token count, average confidence, max attribution score, average hidden state norm, and logit lens layer count — with the best value per metric highlighted. And since not everyone is a machine learning expert, we added a plain-English AI Interpretation box that automatically explains what each result actually means, so anyone can understand what the model was doing without knowing any math or AI.

Summary/Conclusions

Technical Contributions

- First platform to compare mechanistic interpretability across four math-specialized LLMs simultaneously on the same prompt
- Custom LIME, TokenSHAP, and counterfactual implementations entirely in NumPy/PyTorch — no sklearn or SHAP library
- Interactive attribution graph (D3) with live layer/output sliders, built from real attention rollout data
- GPU scheduling enabling 4 × 7B models on a single 80GB instance with zero OOM crashes

Key Findings

- Models with sleeper hidden-state norm growth in middle layers (12–20) produce more structured, step-by-step reasoning traces
- Logit lens reveals correct answers often crystallize as early as layer 16/32 — later layers refine confidence rather than change the prediction
- Counterfactual analysis shows DeepSeek Math is most sensitive to numeric perturbations
- LIME attribution consistently highlights the final numeric value and operation keyword ("divided", "total") as the top-attributed words in the prompt

Key References

- [1] Elhage et al. (2021). *A Mathematical Framework for Transformer Circuits*. Anthropic Technical Report.
- [2] Ribeiro, Singh & Guestrin (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. ACM KDD.
- [3] Lundberg & Lee (2017). *A Unified Approach to Interpreting Model Predictions*. NeurIPS.
- [4] Nostalgebraist (2020). *Interpreting GPT: the Logit Lens*. LessWrong.
- [5] Wiegrefe & Pinter (2019). *Attention is not Explanation*. EACL.
- [6] Abnar & Zuidema (2020). *Quantifying Attention Flow in Transformers*. ACL.

Acknowledgements

Thanks to RunPod for GPU infrastructure access, and to the Applied Data Science faculty for guidance throughout this project. Special thanks to the open-source communities behind HuggingFace Transformers, React, D3.js, and PyTorch.

Interested in partnering with us on a capstone project or exploring other ways to collaborate with the College of IDS? Contact CIDS_CAP@sjsu.edu to discuss potential projects and partnership opportunities.