TECHNICAL WHITE PAPER · MODELCARTOGRAPHY

Model Cartography: A Universal Platform for Neural Network Interpretability, Attribution, and Surgical Intervention

A universal platform for neural network interpretability, attribution, and surgical intervention.

Mechanistic Interp. | SAE | MoE Routing | AI Act | Cross-Platform

ABSTRACT

Model Cartography is a universal SwiftUI application for macOS, iPadOS, and tvOS that operationalizes the field of mechanistic interpretability — turning neural network architectures into navigable maps rather than opaque computational boxes. The application implements a five-stage interpretability pipeline (Instrument → Capture → Attribute → Map → Intervene) across five distinct model architectures through a single, capability-aware adapter interface. Key techniques include logit-lens probing, sparse autoencoder (SAE) feature decomposition, gradient-based saliency, per-expert attribution in Mixture-of-Experts models, and surgical ablation with collateral damage detection. This paper surveys the academic foundations of interpretability research, analyzes the design innovations that enable honest capability degradation across architectures, and makes the case for on-device interpretability as a prerequisite for responsible AI deployment.

01 Introduction

The deployment of machine learning models in consequential settings — medical diagnosis, credit decisioning, legal document analysis, autonomous systems — demands more than a model that produces correct outputs on average. It demands a model whose behavior can be understood, audited, and refined without destroying capabilities that are known to be correct. This is the core problem of machine learning interpretability.

The field has produced a rich set of techniques — saliency methods, probing classifiers, circuit analysis, feature attribution, logit lens, sparse autoencoders — but most of these techniques exist as Python library functions or Jupyter notebook demonstrations, inaccessible to practitioners without a research computing background. Model Cartography materializes these techniques in a native, interactive, cross-platform application that runs entirely on-device.

The conceptual innovation is geographic: rather than asking "what does this model do?" in the abstract, Model Cartography asks "where in this model does a given behavior live?" — and provides the cartographic tools to answer that question with confidence. Regions (neurons, experts, residual stream directions) are mapped, attributed to the inputs that activate them, verified under ablation, and intervened upon. The result is a workflow that makes model understanding as concrete and manipulable as map reading.

02 Scientific Background

2.1 Mechanistic Interpretability

Mechanistic interpretability is the subfield of AI safety and ML research concerned with reverse-engineering the internal algorithms that neural networks implement [1]. Unlike behavioral interpretability — which characterizes input-output relationships — mechanistic interpretability traces the internal computational graph: which neurons activate for which inputs, which circuits implement which behaviors, and how information is routed from input to output.

The field's landmark results include:

- **Induction heads** [2]: A two-layer attention mechanism that implements in-context sequence completion, found reliably across transformer models.
- **Superposition** [3]: The finding that neural networks represent more features than they have dimensions by encoding features in directions that are nearly orthogonal, allowing polysemantic neurons (neurons that activate for multiple semantically unrelated inputs).
- **Features as directions** [4]: The hypothesis, supported by extensive empirical work, that the meaningful computational units in large neural networks are not individual neurons but linear directions in activation space. Sparse autoencoders recover these directions.
- **Universal circuits** [5]: The conjecture that sufficiently similar architectures converge to similar internal algorithms for similar tasks, implying that interpretability insights transfer across model families.

2.2 Saliency Methods

Saliency methods attribute the output of a model to its inputs. The gradient-based family — vanilla gradient [6], integrated gradients [7], GradCAM [8] — computes the partial derivative of the output with respect to each input feature and interprets the magnitude of this derivative as the feature's importance. Occlusion-based methods [9] instead measure the change in output when each input region is masked, avoiding the approximation errors inherent in gradient computation for discontinuous or piecewise-linear functions.

Model Cartography implements both approaches:

- **Gradient saliency** (MockNetwork adapter): True analytical gradients computed through the pure-Swift forward-backward pass of the demo network. Supported because the application owns the model graph.
- **Occlusion saliency** (Core ML adapter): Forward-only occlusion because Core ML is a black box that does not expose internal gradients. The adapter advertises `.saliency` but explicitly does not advertise `.attribution`, reflecting the honest-capability-declaration design.

2.3 Logit Lens

The logit lens technique [10] applies the final classifier head to the residual stream at every intermediate layer, projecting intermediate representations into the model's output vocabulary. For a language model, this reveals how the model's "best guess" at the next token evolves across layers. For a classifier, it shows how class probabilities sharpen as information propagates through the network.

Model Cartography implements logit lens over the Dense Text adapter's residual network, rendering the class probability trajectory as a depth-resolved visualization in the Trace tab. This makes the progressive refinement of model predictions concrete and observable — a qualitative change from knowing the final answer to understanding how the answer was reached.

2.4 Sparse Autoencoders and Feature Discovery

Superposition makes individual neurons difficult to interpret: a single neuron may activate for inputs from multiple, semantically unrelated domains. Sparse autoencoders (SAEs) decompose the activation space into an overcomplete dictionary of directions — "features" — that are sparse (most are inactive for any given input) and monosemantic (each represents a coherent concept) [11].

An SAE is trained with the objective:

```
minimize ||x - W_dec · ReLU(W_enc · x + b_enc)||² + λ · ||ReLU(W_enc · x + b_enc)||₁
```

The L1 penalty on the encoder output enforces sparsity. The decoder matrix `W_dec` contains the recovered feature directions in activation space. Model Cartography trains a SAE on the Dense Text adapter's residual stream and auto-labels each recovered feature by finding the inputs that most activate it, yielding natural-language feature annotations (e.g., `finance: invest, portfolio, yield`).

2.5 Mixture-of-Experts Routing

Mixture-of-Experts (MoE) architectures [12] replace the dense feed-forward sublayer of standard transformers with N independent expert networks and a learned router. For each token at each layer, the router computes a softmax score over all experts; the top-K experts are activated, and their outputs are combined by gate weight. MoE models achieve the capacity of a large dense model (many parameters) with the compute cost of a smaller dense model (only a fraction of parameters activate per token).

This architecture is relevant to interpretability for a specific reason: **routing is interpretable by construction**. The router's decision — which expert to activate for this token — is a discrete, observable assignment that can be attributed to input features. Model Cartography's MoE adapter makes routing the primary interpretability surface: the cortex visualization renders experts as regions, gate weights as activation intensities, and the realized routing path through the network as a traced trajectory. Expert attribution — labeling each expert with the domain that preferentially routes to it — is the MoE equivalent of feature discovery in dense models.

03 Architecture: The ModelAdapter Interface

3.1 The Core Design Principle

Model Cartography never interacts with a model directly. All interactions are mediated by a `ModelAdapter` protocol that declares the capabilities the model can honestly support. The UI reads these capabilities at runtime and renders only what the underlying model can provide — hiding features it cannot support rather than pretending or computing approximate substitutes.

This honest-degradation design has a profound practical implication: it is safe to add any model to Model Cartography, including models that support only a subset of interpretability techniques. A Core ML classifier that exposes no internal activations can still be usefully analyzed for saliency and verification. A routing log imported from an external MoE engine that has no live inference capability can still yield a mapped, labeled cortex and verification metrics.

3.2 Capability Matrix

The `Capabilities` option set defines the available interpretability techniques:

Capability	Core ML	MockNet	Dense+SAE	MoE	Routing Log
Internal activations		✓	✓	✓	✓
Saliency	✓	✓			
Map B (features)			✓		
Logit lens			✓	✓	
Attribution			✓	✓	
Routing path		✓	✓	✓	✓
Ablation / pruning		✓		✓	
Steering			✓		
Verification	✓	✓	✓	✓	✓

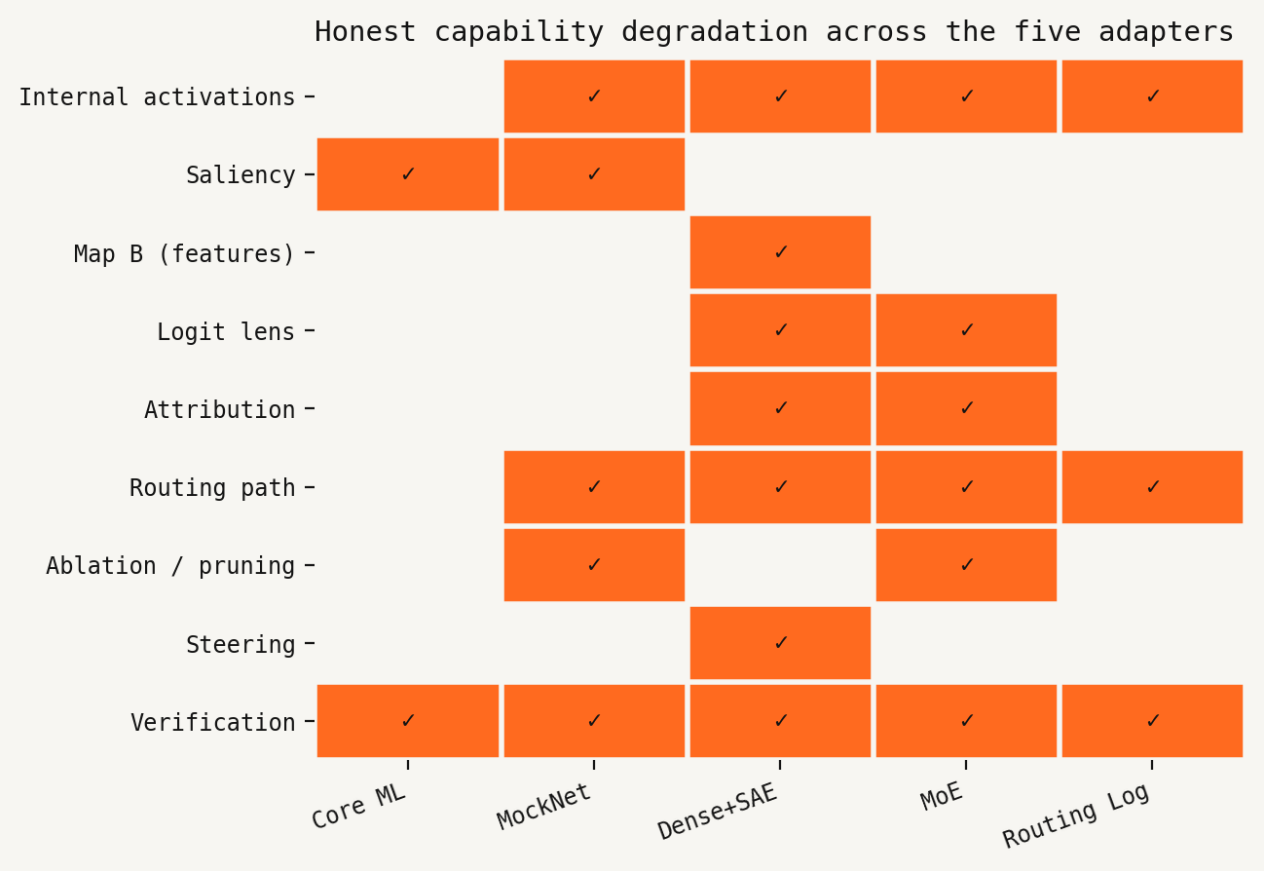

Figure 1. Honest capability degradation: each adapter exposes only the interpretability techniques it can genuinely support.

3.3 The Five-Stage Pipeline

Stage 1: Instrument. The adapter instruments the model — registering activation hooks, gradient checkpoints, or log parsers depending on the model type.

Stage 2: Capture. A forward pass is run on a user-selected input. Intermediate activations are captured at each instrumented point and stored in the `CartographyMap` normalized representation.

Stage 3: Attribute. The Attribution module labels each region with the class or domain it preferentially responds to, by correlating activation statistics across a corpus of labeled inputs.

Stage 4: Map. The `CartographyMap` — a data structure containing regions, edges, activation tensors, trace paths, and feature labels — is surfaced to the UI. The Cortex view renders regions as a grid; the Trace view renders the inference path.

Stage 5: Intervene. The user marks regions for ablation (zeroing their outputs) or applies steering vectors (adding a learned feature direction to the residual stream). The ablated or steered model is re-evaluated, and the Verification module computes before/after accuracy differentials and collateral damage flags.

04 Key Interpretability Contributions

4.1 Collateral Damage Detection

Standard ablation experiments in the literature report top-level accuracy change after ablation. Model Cartography adds collateral damage detection: when ablating a set of regions associated with Class A, the Verify tab checks whether accuracy on Classes B and C also changes. A significant accuracy drop on Class B following ablation of Class A's regions is a collateral damage flag — it indicates that the ablated regions contribute to Class B's correct classification as well, suggesting functional overlap or shared circuitry.

This extends the standard ablation analysis from "how much does this ablation harm the target class?" to the richer question "what else does this ablation break?" — critical information for any surgical intervention intended to remove a specific capability without degrading others.

4.2 Expert Utilization and Cold Expert Pruning

For MoE models, Model Cartography tracks the aggregate routing utilization of each expert across a labeled corpus. High-utilization (hot) experts are core to the model's performance on their domain; low-utilization (cold) experts are candidates for pruning — removing them from the model reduces memory footprint without significantly impacting inference quality.

The Intervene tab surfaces a ranked list of experts by utilization. Marking cold experts and re-verifying provides empirical measurement of the accuracy impact, turning expert pruning from a heuristic operation into a measurable, reversible experiment. The empirical finding in Model Cartography's demo MoE — that cold experts can be removed with near-zero accuracy impact while hot domain-specific experts cannot — replicates the theoretical prediction from conditional computation theory [13].

4.3 Activation Steering Without Retraining

Activation steering [14] adds a scaled feature direction to the model's residual stream at inference time, biasing the model's predictions without updating any weights. This technique was demonstrated in the interpretability literature for large language models; Model Cartography implements it for the Dense Text adapter's SAE-recovered features. The Intervene tab exposes a gain slider; the user can observe a weather-classified sentence shift toward finance classification as the finance feature direction is added to the residual at a controlled magnitude.

This makes the causal role of learned features observable and manipulable in real time — a compelling demonstration of why features-as-directions is a useful frame for model understanding.

05 Simplify Mode: Accessibility for Non-Researchers

A distinctive design decision in Model Cartography is the Simplify toggle. Research interpretability vocabulary ("logit lens," "ablation," "collateral damage," "residual stream") is precise but inaccessible to practitioners who lack a research background. Simplify mode relabels every technical term in plain language while leaving the underlying data and analysis unchanged:

Technical	Simplified
Logit lens	How the guess takes shape
Ablate	Remove and recheck
Collateral damage	This broke something that was working
Residual stream	Information accumulator
Routing path	Path taken
Attribution	What drove this answer

This design enables Model Cartography to serve two distinct audiences: interpretability researchers who want precise vocabulary and domain practitioners who want actionable insight. The same application, the same data, two vocabularies.

06 Economic and Policy Impact

6.1 Explainability Requirements in Regulated Domains

EU Regulation 2024/1689 (the AI Act) [15] classifies AI systems used in healthcare, critical infrastructure, education, employment, essential services, law enforcement, migration control, and administration of justice as high-risk, requiring technical documentation, logging, transparency, and human oversight. For image classifiers used in medical device software, explainability is an expected property under both the AI Act and FDA guidance on Software as a Medical Device (SaMD) [16].

Model Cartography provides exactly the kind of explainability evidence that these regulatory frameworks seek: saliency maps showing which input regions drove the prediction, attribution graphs showing which model components contributed, and ablation experiments demonstrating that the model's correct behavior is robust to the removal of specific components. This is not merely a conceptual claim; the Verify tab produces quantitative accuracy reports that can be included in technical documentation.

6.2 Reducing the Cost of Model Auditing

Model auditing — the process of evaluating a deployed model's behavior systematically before or after deployment — is currently expensive, requiring data science expertise, custom tooling, and compute infrastructure. Model Cartography democratizes model auditing by providing these capabilities in a native, self-contained application that runs on hardware the operator already owns. A compliance officer can run a saliency analysis and an ablation experiment without writing a line of Python.

6.3 Trust Calibration

A recurring finding in human-factors research on AI-assisted decision-making is that users tend toward two failure modes: over-trust (accepting AI outputs without appropriate skepticism) and under-trust (ignoring AI outputs even when they are accurate) [17]. Both failure modes have costs. Interpretability tools that make model confidence and model reasoning legible to decision-makers help calibrate trust appropriately — users learn which regions of the input space the model is reliable in, and which it is not.

---

07 Limitations and Future Work

Scale. Model Cartography's built-in adapters target small to medium architectures (hundreds to millions of parameters). Applying these techniques to production-scale LLMs (7B–70B parameters) would require adapter implementations that offload to GPU compute and manage activation tensors that do not fit in CPU RAM. The adapter interface is designed to support this without UI changes; it is an infrastructure problem, not a design problem.

Causal vs. correlational attribution. The attribution methods implemented (gradient-based, activation magnitude) are correlational — they measure which components covary with the output, not which components causally produce it. Truly causal attribution requires intervention experiments (counterfactual inputs, activation patching) that are computationally expensive and model-specific. The Intervene tab's ablation experiments provide causal evidence, but the Trace tab's attribution is correlational.

Distribution shift. All attribution and verification results are conditioned on the evaluation corpus. Attribution labels learned from one distribution may not generalize to inputs from a different distribution.

---

08 Conclusion

Model Cartography demonstrates that the full mechanistic interpretability pipeline — saliency, feature discovery, logit-lens probing, routing visualization, attribution, ablation, collateral damage detection, and activation steering — can be delivered as a unified, native, cross-platform application requiring no Python

runtime, no cloud dependency, and no research computing background. The adapter pattern that enables honest capability degradation across five architectures makes the application extensible to any model that can expose its internals through the `ModelAdapter` protocol. As regulatory requirements for AI explainability intensify and as on-device AI deployment expands, tools like Model Cartography become essential infrastructure rather than academic curiosities.

// References

[1] Olah, C., "Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases," Transformer Circuits Thread, 2022. https://transformer-circuits.pub/2022/mech-interp-essay/index.html

[2] Olsson, C., et al., "In-context Learning and Induction Heads," Transformer Circuits Thread, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html

[3] Elhage, N., et al., "Toy Models of Superposition," Transformer Circuits Thread, 2022. https://transformer-circuits.pub/2022/toy_model/index.html

[4] Mikolov, T., et al., "Linguistic Regularities in Continuous Space Word Representations," Proceedings of NAACL HLT 2013.

[5] Li, Y., et al., "Convergent Learning: Do Different Neural Networks Learn the Same Representations?" ICLR Workshop, 2016.

[6] Simonyan, K., Vedaldi, A., and Zisserman, A., "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," arXiv:1312.6034, 2013.

[7] Sundararajan, M., Taly, A., and Yan, Q., "Axiomatic Attribution for Deep Networks," Proceedings of the 34th ICML, 2017.

[8] Selvaraju, R.R., et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," ICCV 2017.

[9] Zeiler, M.D., and Fergus, R., "Visualizing and Understanding Convolutional Networks," ECCV 2014.

[10] nostalgebraist, "interpreting GPT: the logit lens," LessWrong, 2020. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/

[11] Bricken, T., et al., "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning," Transformer Circuits Thread, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html

[12] Shazeer, N., et al., "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," ICLR 2017.

[13] Bengio, Y., et al., "Conditional Computation in Neural Networks for Faster Models," arXiv:1511.06297, 2015.

[14] Templeton, A., et al., "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet," Anthropic, 2024. https://transformer-circuits.pub/2024/scaling-monosemanticity/

[15] European Parliament and Council, "Regulation (EU) 2024/1689 (AI Act)," Official Journal of the European Union, July 2024.

[16] U.S. Food and Drug Administration, "Artificial Intelligence and Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) Action Plan," January 2021.

[17] Lee, J.D., and See, K.A., "Trust in Automation: Designing for Appropriate Reliance," Human Factors, vol. 46, no. 1, pp. 50–80, 2004.