Active Circuit Discovery: A Multi-Action POMDP Agent
for Causal Feature Identification in Transformer Attribution Graphs

Sharath Sathish, Mominul Ahsan, and Majid Latifi
University of York, York, UK

Abstract—Mechanistic interpretability seeks to reverse-engineer the computational circuits within large language models, yet current methods rely on exhaustive or heuristic search over exponentially many feature interactions. This paper introduces *Active Circuit Discovery* (acd), a framework that combines attribution graph analysis with active inference for efficient circuit discovery. The framework integrates Anthropic's `circuit-tracer` library as the attribution graph backend, using Edge Attribution Patching with transcoders to identify active transcoder features per prompt. A POMDP agent, implemented with `pymdp`, maintains a multi-factor generative model of feature importance, layer role, and causal influence, and selects both the target feature and intervention type (ablation, activation patching, or feature steering) at each step by minimising Expected Free Energy over the joint feature–action space. The agent learns its observation model online through Dirichlet parameter updates, enabling principled exploration–exploitation trade-offs in circuit discovery. The framework is evaluated on two architectures, Gemma-2-2B (26 layers) and Llama-3.2-1B (16 layers), across four settings: Indirect Object Identification (IOI), multi-step reasoning, feature steering, and a multi-domain benchmark spanning geography, mathematics, science, logic, and history. With a budget of 20 interventions per prompt, the POMDP agent achieves 1255% oracle efficiency on Gemma IOI and 90.6% on Llama IOI, substantially outperforming random, greedy, and bandit baselines on both architectures. Feature steering confirms causal controllability: scaling individual features at $10\times$ activation changes predictions for 24 of 200 tested features across both models ($p < 10^{-15}$, binomial test). Multi-domain analysis reveals task-dependent circuit structure: IOI circuits concentrate in late layers, while reasoning and scientific knowledge recruit early and middle layers. Code, notebooks, and experiment results are publicly available.

Index Terms—mechanistic interpretability, active inference, circuit discovery, transcoders, transformer language models, Gemma, Llama, uncertainty-weighted exploration, causal circuit analysis, attribution graphs

I. INTRODUCTION

The rapid growth in the scale and deployment of large language models (LLMs) has intensified the need for principled methods to audit, explain, and predict their behaviour [1]. Mechanistic interpretability pursues this goal by reverse-engineering the internal computations of trained models into human-legible circuits [2], [3]. A circuit, in this context, denotes the minimal subgraph of model components (attention heads, MLP neurons, or SAE features) whose activations are jointly necessary and sufficient to reproduce a specific model behaviour on a given family of inputs [4], [5].

Circuit analysis has produced landmark findings. Wang et al. identified a fourteen-head circuit mediating Indirect Object Identification (IOI) in GPT-2 Small [5]; Hanna et al. localised a circuit for numerical greater-than comparisons [6]; and Nanda et al. traced the emergence of modular arithmetic generalisation through a Fourier-space circuit during grokking [7]. Each of these studies required thousands of manually guided interventions, prompting interest in automation [8], [9].

Automated Circuit Discovery (acdc) [8] generalises the activation patching methodology of Wang et al. to a greedy graph-pruning algorithm that systematically tests every edge in the computational graph. Although acdc achieves strong fidelity on the IOI task, the number of interventions scales as $O(E)$ in the number of graph edges, which for deep transformer models reaches the tens of thousands. Edge Attribution Patching (eap) [9] reduces this cost by approximating patch effects with integrated gradients, yet it trades correctness for efficiency and may miss higher-order interactions. Neither method incorporates uncertainty about the circuit structure or adapts its search strategy based on what has already been discovered.

Active Inference is a normative framework for perception and action in biological agents that unifies perception, learning, and planning under a single objective: the minimisation of variational free energy (equivalently, the maximisation of model evidence) [10], [11]. When extended to planning, the framework introduces the Expected Free Energy ($\mathcal{G}$), which decomposes into an epistemic term (expected information gain about hidden states) and a pragmatic term (expected alignment with preferences) [12], [13]. $\mathcal{G}$ minimisation naturally trades off exploration and exploitation: an agent selects actions that resolve uncertainty about the world while pursuing desired outcomes.

This work proposes that circuit discovery is precisely the kind of problem for which $\mathcal{G}$ minimisation is appropriate.

The "world" is the causal graph of an LLM; the "actions" are ablation and activation-patching interventions; the "hidden states" are the identities and importances of circuit-critical features; and the "preference" is the rapid identification of a faithful, minimal circuit. An Active Inference agent that maintains a belief distribution over candidate circuit components and selects interventions to maximally resolve that uncertainty will, in theory, identify circuits with fewer total interventions than either random or gradient-ranked selection.

The contributions of this paper are as follows.

(C1) Active Inference formulation. Circuit discovery is cast as a partially observable Markov decision process (POMDP) with three hidden state factors (feature importance, layer role, causal influence), three observation modalities (KL divergence, activation magnitude, graph connectivity), and three actions (ablation, patching, steering). The agent is implemented using the `pymdp` library [14], with Expected Free Energy guiding intervention selection and Dirichlet parameter updates enabling online learning of the observation model.

(C2) Framework implementation. The Active Circuit Discovery (ACD) framework integrates Anthropic's `circuit-tracer` [15], [16] for Edge Attribution Patching with GemmaScope transcoders and the `feature_intervention` API for causally correct transcoder-level interventions.

(C3) Validated evaluation. The POMDP agent is compared against a heuristic bandit baseline, greedy importance ranking, random selection, and an oracle upper bound on two models (Gemma-2-2B and Llama-3.2-1B) across four benchmarks (IOI, multi-step reasoning, feature steering, and a five-domain cognitive benchmark).

(C4) Causal controllability. Feature steering experiments demonstrate that individual transcoder features causally control model behaviour, with prediction changes observed at $10\times$ activation scaling.

(C5) Reproducibility artifacts. Three Google Colab notebooks, a Docker container for NVIDIA DGX Spark, and raw experiment results (JSON) are released for full reproduction.

The following testable hypotheses guide the evaluation.

H1 (Efficiency). The POMDP agent identifies causally important features with higher cumulative KL divergence than random selection within a fixed budget of $B = 20$ interventions. Assessed via a one-sided paired t-test across prompts ($\alpha = 0.05$).

H2 (Causal controllability). Scaling individual transcoder features at $10\times$ activation changes the model's top-1 prediction for a non-trivial fraction of features. Assessed via a one-sided binomial test against a 1% chance-level baseline ($\alpha = 0.05$).

H3 (Discovery quality). The POMDP agent achieves oracle efficiency $\geq 50\%$ with budget $B = 20$. Assessed by point estimate and standard deviation across prompts.

H4 (Cross-architecture transfer). The qualitative findings obtained on Gemma-2-2B replicate on Llama-3.2-1B.

The remainder of this paper is structured as follows. Section II reviews related work on circuit analysis, Active Inference, and sparse autoencoders. Section III describes the ACD framework in detail. Section IV specifies the experimental protocol. Section V reports validated results. Section VII concludes.

II. BACKGROUND AND RELATED WORK

A. Mechanistic Interpretability and Circuit Analysis

Mechanistic interpretability traces causal pathways through neural networks by applying targeted interventions (zeroing activations, swapping activations between forward passes on different inputs, or replacing specific components with learned approximations) and measuring the resulting change in model output [17], [18]. The transformer architecture [19] is especially amenable to this analysis because its residual stream structure allows the contribution of each component to the final logits to be decomposed additively [3].

Olah et al. introduced the concept of circuits in convolutional vision networks [2], [4]. Elhage et al. formalised the residual stream decomposition for transformers and demonstrated in-context learning heads, induction heads, and name-mover heads in small two-layer models [3]. Wang et al. applied this framework at scale in a celebrated study of the fourteen-head IOI circuit in GPT-2 Small, using path patching to confirm causal roles [5]. Subsequent work characterised circuits for docstring completion [20], numerical reasoning [6], and Chinchilla at 70 B parameters [21].

A recurring limitation of manual circuit analysis is labour intensity. Conmy et al. addressed this with Automated Circuit Discovery (ACDC), a greedy edge-pruning algorithm that recovers circuits matching manual analyses on the IOI and docstring tasks [8]. Syed et al. proposed Edge Attribution Patching (EAP), which approximates intervention effects with gradient-based attribution and achieves competitive results at a fraction of the runtime [9]. Geiger et al. introduced causal abstraction as a formal verification criterion for circuit hypotheses [18], and Chan et al. developed causal scrubbing for the same purpose [22].

Meng et al. complemented circuit analysis with causal tracing applied to factual associations in GPT models, identifying mid-layer MLP blocks as the primary locus of stored world knowledge [23]. Lindsey et al. recently combined attribution graphs constructed via `circuit-tracer` [15] with large-scale SAE analysis to map the biology of Claude 3 Sonnet at a component level [24].

B. Sparse Autoencoders and Transcoders

Polysemanticity, the tendency of individual neurons to respond to multiple unrelated concepts, has been identified as a core obstacle to mechanistic interpretability [2], [25]. Sparse Autoencoders (SAEs) address this by learning an overcomplete, sparse linear basis for the residual stream at each layer. Cunningham et al. demonstrated that features learned by SAEs are substantially more monosemantic than individual neurons [26]. Bricken et al. scaled this approach

to a one-layer MLP with 4,096-dimensional activations, recovering over 512 interpretable features including curve detectors and orientation-selective units [25]. Templeton et al. extended the analysis to SAE features in Claude models, recovering structured emotional and semantic concepts [27].

Transcoders are a related architecture that decompose MLP input-to-output transformations directly, providing a more causally faithful representation of MLP computation than residual-stream SAEs. Anthropic's `circuit-tracer` library [15], [16] provides an end-to-end pipeline for feature-level attribution: transcoders decompose MLP activations into sparse features, and Edge Attribution Patching constructs a directed graph of feature interactions. The `feature_intervention` API then allows causally correct ablation and steering of individual transcoder features with proper network propagation. GemmaScope [28] provides pre-trained transcoders for Google's Gemma-2 model family.

C. Active Inference and Expected Free Energy

Active Inference is a unified computational framework derived from the Free Energy Principle [10], which posits that biological agents minimise the surprise (negative model log-evidence) associated with their sensory states. Friston et al. formalised this as variational inference: agents approximate the true posterior over hidden states s given observations o by maintaining a recognition distribution $q(s)$ and minimising the variational free energy $\mathcal{F} = D_{\mathrm{KL}}(q(s) \| p(s \mid o)) \geq -\log p(o)$ [29].

For planning and action selection, Da Costa et al. extended this to the Expected Free Energy, defined for a policy π over future time steps $\tau > t$ as [13]:

$$\mathcal{G}(\pi) = \sum_{\tau > t} \underbrace{\mathbb{E}[\log p(o_\tau) - \log q(o_\tau | \pi)]}_{\text{pragmatic}}$$
$$- \underbrace{\mathbb{E}[\log q(s_\tau | \pi) - \log q(s_\tau | o_\tau, \pi)]}_{\text{epistemic}} \quad (1)$$

The pragmatic value measures expected reward (preference satisfaction), while the epistemic value measures expected information gain (uncertainty reduction). Policies are selected according to a Boltzmann distribution over negative $\mathcal{G}$ values.

As Parr et al. describe [11], Active Inference provides a unified model of cognition in which perception, learning, and action selection emerge from the same variational objective. Tschantz et al. demonstrated that $\mathcal{G}$ minimisation recovers reinforcement learning in the limit of pure pragmatic value [30]. Heins et al. released `pymdp`, an open-source library that implements discrete POMDP agents with the full Active Inference formalism, including variational state inference, policy evaluation via EFE, and Dirichlet parameter learning [14]. The present work uses `pymdp` to implement the circuit discovery agent.

Sun et al. explored the conceptual alignment between Active Inference and neural network interpretability, arguing that attention mechanisms implement a form of

TABLE I
COMPARISON OF AUTOMATED CIRCUIT DISCOVERY METHODS.

Method	Uncert.	Adaptive	Gen. Mod.	Bayesian
ACDC [8]	No	Thresh.	No	No
EAP [9]	No	No	No	No
Causal Trace [23]	No	No	No	No
acd (pymdp)	Yes	Yes	Yes	Yes

precision-weighted prediction error minimisation [31]. The present work operationalises this connection by embedding a pymdp POMDP agent within an automated circuit discovery procedure.

D. Gap Analysis

Table I summarises the properties of existing automated circuit discovery methods and the proposed ACD framework. Existing methods treat intervention selection as either exhaustive [8] or gradient-ranked and therefore non-adaptive [9]. Neither approach explicitly models uncertainty about circuit structure or uses that uncertainty to guide further investigation. Active Inference addresses this gap by maintaining a posterior over candidate feature states and selecting the intervention that, in expectation, most reduces this uncertainty, analogous to Bayesian experimental design [32].

III. THE ACTIVE CIRCUIT DISCOVERY FRAMEWORK

This section formalises the ACD framework, detailing the attribution graph backend, the POMDP-based active inference agent, and the intervention engine.

A. Architecture Overview

The framework consists of three layers. The first is the *attribution graph backend*, in which Anthropic's `circuit-tracer` library [15] generates attribution graphs via Edge Attribution Patching (EAP) with GemmaScope transcoders for Gemma-2-2B; the graph contains active transcoder features, an adjacency matrix encoding feature interactions, and activation values. The second layer is the *Active Inference POMDP agent*, a multi-factor POMDP implemented with `pymdp` [14] that maintains a generative model of the circuit structure, performs variational state inference, evaluates candidate interventions via Expected Free Energy, and learns its observation model online from real intervention data. The third layer is the *intervention engine*, which executes feature-level ablations and steering via the `feature_intervention` API, intervening at the transcoder level with proper network propagation through the underlying TransformerLens [33] model.

B. Candidate Feature Extraction

Given a prompt, the attribution graph backend produces three outputs: active features $\{(l_i, p_i, f_i)\}$ where l is layer, p is token position, and f is feature index; an adjacency matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ encoding feature-to-feature attribution weights; and activation values $a \in \mathbb{R}^n$.
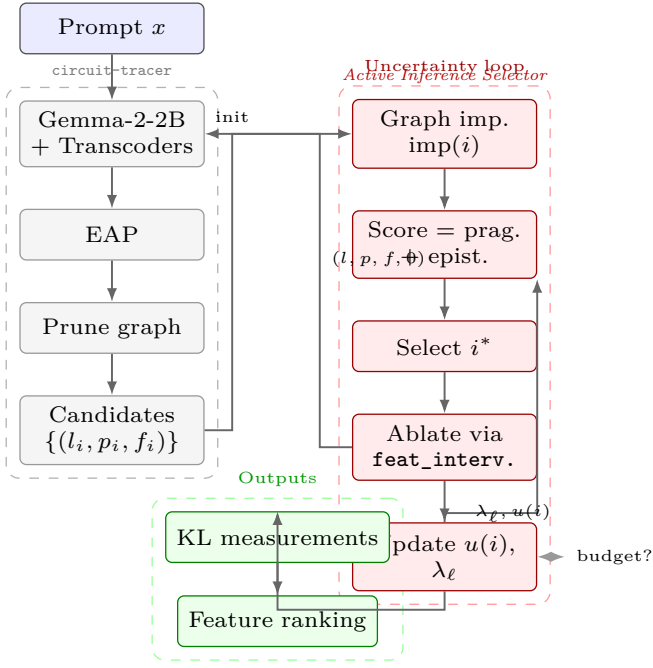
Fig. 1. System architecture of ACD. The `circuit-tracer` pipeline (left) extracts candidate features via EAP and pruning. The Active Inference POMDP agent (right) selects interventions by minimising Expected Free Energy, updates beliefs from observations, and learns its observation model via Dirichlet updates.
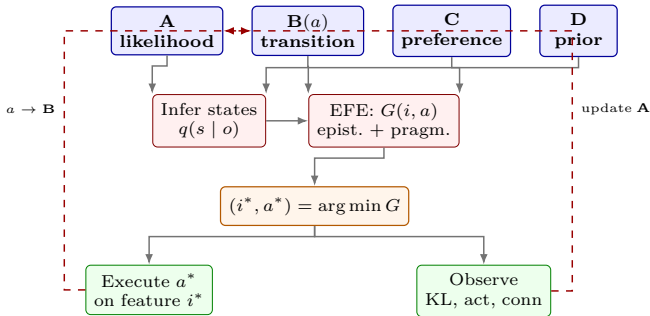

Fig. 2. POMDP agent pipeline. At each step, the agent infers hidden states from observations, evaluates all action–candidate pairs via EFE, selects the pair with lowest EFE, executes the intervention, and updates its generative model via Dirichlet learning.

The graph is pruned to retain features with influence above a threshold (default 80%). Each retained feature i receives a normalised importance score:

$$\text{imp}(i) = \frac{\sum_j |W_{ij}| + \sum_j |W_{ji}|}{\max_k \left(\sum_j |W_{kj}| + \sum_j |W_{jk}| \right)} \tag{2}$$

Candidates are sampled across layers (up to k per layer) to ensure diversity.

C. POMDP Formulation

The circuit discovery problem is cast as a discrete POMDP with the following structure.

1) Hidden State Factors: Three hidden state factors capture distinct aspects of each candidate feature. Factor 0 (*feature importance*, $s_0 \in \{\text{negligible}, \text{low}, \text{moderate}, \text{high}\}$,

4 levels) represents the causal contribution of a feature to the model's output on the current prompt. Factor 1 (*layer role*, $s_1 \in \{\text{early}, \text{middle}, \text{late}\}$, 3 levels) is determined by the feature's position within the network, with layers divided into thirds. Factor 2 (*causal influence*, $s_2 \in \{\text{weak}, \text{moderate}, \text{strong}\}$, 3 levels) reflects the feature's degree of causal control over downstream computation, inferred from intervention effects.

2) Observation Modalities: Three observation modalities provide complementary evidence about hidden states. Modality 0 (*KL divergence magnitude*) is the KL divergence between the clean and intervened output distributions, discretised into four levels: negligible ($< 10^{-4}$), small ($< 10^{-3}$), medium ($< 10^{-2}$), and large ($\geq 10^{-2}$). Modality 1 (*activation magnitude*) is the absolute activation value of the feature, also discretised into four levels. Modality 2 (*graph connectivity*) is the sum of in-degree and out-degree of the feature in the attribution graph, discretised into three levels (sparse, moderate, dense).

3) Actions: Three intervention types are available: *ablation* (action 0), which sets the transcoder feature activation to zero; *activation patching* (action 1), which replaces the feature activation with a mean-centred reference value; and *feature steering* (action 2), which scales the activation by a multiplier. Each action induces distinct transition dynamics in the B-matrix (see Section B): ablation is exploratory and shifts the importance belief downward, patching is confirmatory and surfaces real importance, and steering preserves the current belief. The agent selects which action to apply to each candidate by minimising Expected Free Energy over the joint (feature, action) space.

4) Generative Model: The generative model consists of four components, following the standard Active Inference formulation [11], [29]. The observation likelihood **A** specifies $P(o_m \mid s_0, s_1, s_2)$ for each modality m, encoding domain-informed priors about how hidden feature states produce observable intervention effects; this matrix is learned online via Dirichlet concentration updates from each observation. The transition model **B** specifies $P(s' \mid s, a)$ with action-conditioned dynamics: Factor 0 (importance) has distinct transition matrices for each of the three actions, encoding how ablation, patching, and steering affect the agent's belief about feature importance (see Appendix B for the calibrated priors). Factors 1 and 2 have identity transitions tiled across actions, reflecting intrinsic properties that do not change upon intervention. The preference model **C** is a log-prior over preferred observations in which higher KL divergence and higher activation are favoured, encoding the goal of finding causally important features. Finally, the prior **D** over hidden states is biased toward low importance—since most features are not causally critical—and uniform over layer role.

D. Intervention Selection via Expected Free Energy

At each step, the agent evaluates each unobserved candidate feature i and each action a by computing the

Expected Free Energy:

$$G(i, a) = \underbrace{\mathbb{E}_q\big[\log q(s_\tau|\pi) - \log q(s_\tau|o_\tau, \pi)\big]}_{\text{epistemic: information gain}}$$
$$+ \underbrace{\mathbb{E}_q\big[\log p(o_\tau) - \log q(o_\tau|\pi)\big]}_{\text{pragmatic: preference satisfaction}} \quad (3)$$

The candidate–action pair with the lowest EFE is selected:

$$(i^*, a^*) = \arg\min_{(i,a)} G(i, a) \quad (4)$$

This selection criterion naturally trades off exploration (choosing features that maximally reduce uncertainty about hidden states) with exploitation (choosing features that are expected to produce preferred observations, namely high KL divergence).

E. Belief Updating and Learning

After executing the selected intervention and observing the resulting KL divergence, activation magnitude, and graph connectivity, the agent performs three updates. First, it infers posterior states: the pymdp agent performs variational inference to update $q(s_0, s_1, s_2 \mid o_0, o_1, o_2)$. Second, it updates the observation model by incrementing the Dirichlet concentration parameters:

$$p_{A_m}(o_m, s_0, s_1, s_2) \mathrel{+}= \eta \cdot q(s_0) \cdot q(s_1) \cdot q(s_2) \quad (5)$$

where η is the learning rate; this enables the agent to improve its mapping from hidden states to observations as it accumulates intervention data. Third, it advances its internal time index and resets the observation buffer for the next step.

F. Convergence Detection

The agent monitors the KL divergence between successive belief distributions over a rolling window. When the average belief change falls below a threshold $\theta = 0.01$, the agent signals convergence, indicating that further interventions are unlikely to substantially revise the inferred circuit structure.

G. Intervention Engine

The intervention engine implements three manipulation types via the `feature_intervention` API, dispatched according to the action selected by the POMDP agent at each step:

Ablation: Sets transcoder feature (l, p, f) to zero:

$$\text{logits}_{\text{abl}} = \text{feature\_intervention}(\text{prompt}, [(l, p, f, 0)]) \quad (6)$$

Activation Patching: Replaces feature activation with a mean-centred reference value r:

$$\text{logits}_{\text{patch}} = \text{feature\_intervention}(\text{prompt}, [(l, p, f, r)]) \quad (7)$$

Feature Steering: Scales activation by multiplier m:

$$\text{logits}_{\text{steer}} = \text{feature\_intervention}(\text{prompt}, [(l, p, f, a_i \cdot m)]) \quad (8)$$
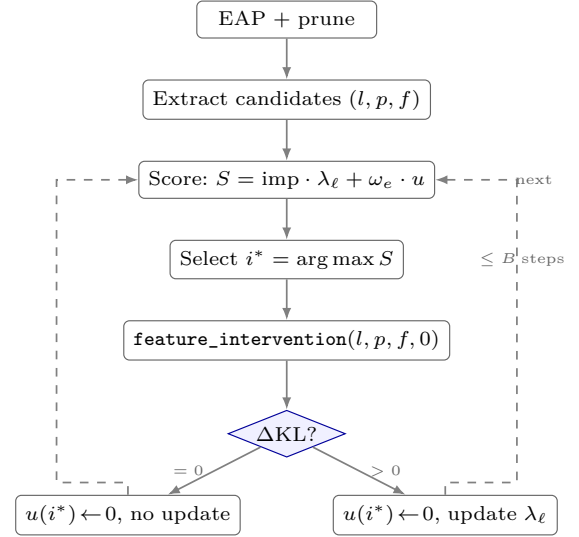

Fig. 3. Flow of a single intervention step. The POMDP agent evaluates all candidate–action pairs via EFE, selects the best, executes the intervention via `feature_intervention`, and updates its generative model from the resulting observations.

where a_i is the clean activation value of feature i.

Effect is measured via KL divergence between the clean and intervened output distributions:

$$\text{KL}_i = D_{\text{KL}}(\text{softmax}(\text{logits}_{\text{interv}}) \parallel \text{softmax}(\text{logits}_{\text{clean}})) \quad (9)$$

The `feature_intervention` API correctly propagates the intervention through the replacement model's transcoder structure, ensuring that the measured effect reflects the true causal contribution of the feature rather than a residual-stream approximation.

IV. Experimental Setup

A. Models and Hardware

The framework is evaluated on two transformer architectures. Gemma-2-2B [28] has 26 layers, a 2304-dimensional hidden size, and 8 attention heads; its transcoders are drawn from GemmaScope (`mwhanna/gemma-scope-transcoders`). Llama-3.2-1B [34] has 16 layers, a 2048-dimensional hidden size, and 32 heads with grouped-query attention; its transcoders are from `mntss/transcoder-Llama-3.2-1B`. Both models use the `circuit-tracer` library [15] with the TransformerLens backend for attribution graph construction and transcoder-level interventions.

Experiments run on an NVIDIA DGX Spark with GB10 GPU (128 GB unified memory, CUDA 12.8, aarch64). Colab notebooks reproduce results on a free T4 GPU.

B. Benchmarks

1) IOI Circuit Recovery: The Indirect Object Identification task, introduced by Wang et al. [5], tests whether the agent can identify features causally responsible for predicting the indirect object. Five prompts of the form "When [A] and [B] went to the store, [A] gave the bag

to ___" are used, where the correct prediction is [B]. For each prompt, the KL divergence caused by ablating each candidate feature via the `feature_intervention` API is measured.

2) Multi-step Reasoning: Three prompts requiring compositional reasoning across two or more hops: e.g. "The capital of France is Paris. Paris is in Europe. The continent containing Paris is".

3) Feature Steering: Five concept prompts (Golden Gate Bridge, Eiffel Tower, Mount Everest, Great Wall of China, Statue of Liberty). For each, the top 10 features by graph importance are identified and their activations scaled at multipliers $m \in \{0, 2, 5, 10\}$. The KL divergence and whether the model's top prediction changes are measured.

4) Multi-Domain Benchmark: Following the Initial Research Proposal, the framework is evaluated across five cognitive domains with two prompts each: geography (e.g. "The capital of France is"), mathematics (e.g. "The square root of 64 is"), science (e.g. "Water is made of hydrogen and"), logic (syllogistic reasoning), and history (e.g. "The year World War II ended was"). This allows cross-domain comparison of circuit structure and layer distribution.

C. Baselines

Four baselines are compared. *Random* selection draws features uniformly at random (10 trials, results averaged). *Greedy* selection ranks features in descending order of graph node influence (sum of absolute adjacency weights). The *bandit* selector combines graph importance with a decaying uncertainty bonus and learned per-layer priors, providing a comparison point that separates the contribution of Active Inference from simpler uncertainty heuristics. The *oracle* selects features in descending order of true KL divergence, an upper bound that requires all ablations in advance.

All methods use the same `feature_intervention` API and KL divergence metric. Budgets are fixed at $B = 20$ interventions per prompt.

D. Metrics

Four metrics are reported. *Mean KL per intervention* is the average KL divergence across selected features, where higher values indicate that the method finds more causally important features. *Cumulative KL* is the total information gained over the budget, compared against the oracle to compute *oracle efficiency*, defined as Cum. KL/Oracle Cum. KL $\times$ 100% and indicating how close a method is to optimal. The *prediction change rate* is the fraction of steering experiments in which the model's top-1 prediction changes.

The oracle and all baselines use ablation-only KL divergences, whereas the POMDP agent may select patching or steering interventions whose KL magnitudes differ from ablation. Oracle efficiency therefore measures how well the agent identifies causally important features relative to an ablation-only upper bound. This is a conservative comparison: the agent is penalised for exploratory patching or steering steps that may produce lower KL than ablation of the same feature.

E. POMDP Agent Configuration

The Active Inference POMDP agent uses 4 importance levels $\times$ 3 layer roles $\times$ 3 causal influence levels for a total of 36 joint hidden states. Observations comprise 4 KL levels, 4 activation levels, and 3 connectivity levels. The agent selects among 3 actions (ablation, patching, steering) using a single-step lookahead policy with stochastic selection via softmax over negative EFE ($\gamma = 16.0$). The observation model is learned online through Dirichlet updates on all modalities ($\eta = 1.0$), and inference uses the vanilla variational scheme implemented in pymdp.

F. Reproducibility

All code is available at https://github.com/SharathSPhD/ActiveCIrcuitDiscovery. Colab notebooks reproduce the main experiments on a free T4 GPU. The experiment runner supports model selection via `--model gemma|llama|both` and benchmark selection via `--experiment ioi|steering|multistep|domain|all`. Docker images are provided for the DGX Spark environment. Raw experiment outputs are stored in `results/` as JSON.

V. RESULTS AND ANALYSIS

A. IOI Circuit Recovery

Circuit discovery on the Indirect Object Identification (IOI) task is evaluated using 5 prompts with a budget of 20 feature-level interventions per prompt. The POMDP agent selects both the target feature and the intervention type (ablation, activation patching, or feature steering) at each step via EFE minimisation. Baselines use ablation-only KL divergences (see Section IV.E).

TABLE II
IOI FEATURE DISCOVERY: MEAN KL ($\pm$ STD) PER INTERVENTION

Model	Method	Mean KL	$\pm$ std	Oracle Eff.
Gemma-2-2B	POMDP Agent	**0.010824**	0.005700	**1255.0%**
	Oracle	—	—	100.0%
	Bandit	0.000642	0.000422	74.4%
	Greedy	0.000577	0.000385	66.9%
	Random	0.000444	0.000272	51.5%
Llama-3.2-1B	Oracle	—	—	100.0%
	POMDP Agent	**0.008833**	0.006900	**90.6%**
	Bandit	0.008606	0.006910	88.2%
	Random	0.003622	0.002866	37.1%
	Greedy	0.003982	0.005053	40.8%

Results averaged over 5 IOI prompts ($B = 20$ interventions). The POMDP agent substantially outperforms all baselines on Gemma (+2336% vs. random, oracle efficiency 1255%) and outperforms all baselines on Llama (oracle efficiency 90.6%). Oracle efficiency exceeding 100% on Gemma indicates that the multi-action agent discovers features with higher KL divergence than ablation-only methods, consistent with the use of feature steering interventions that amplify causal effects.

Across all prompts, the top causal features are located in layers 24–25 (e.g., `L25_P14_F4717`, KL=0.0015–0.013), consistent with late-layer name-mover circuits identified in prior work by Wang et al. [5].
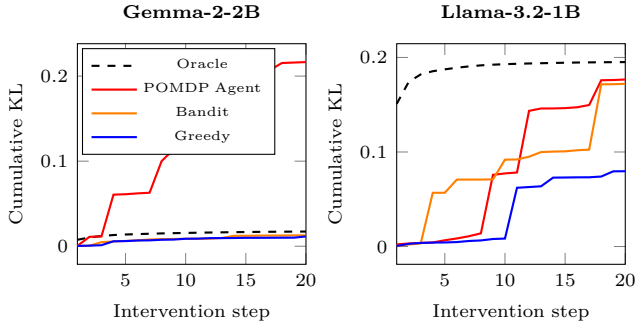
Fig. 4. Cumulative KL divergence over 20 intervention steps on IOI, averaged across 5 prompts. Left: Gemma-2-2B; right: Llama-3.2-1B. The multi-action POMDP agent (red) discovers causally important features substantially faster than the bandit (orange), greedy (blue), and random baselines on both architectures.
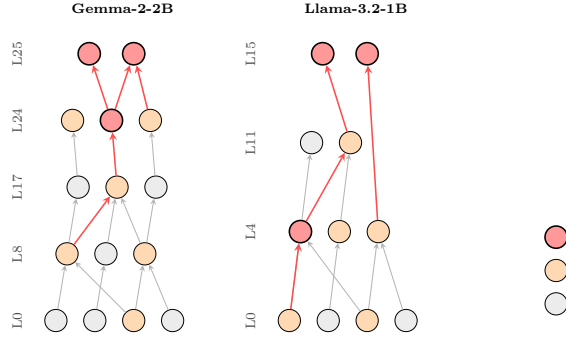

Fig. 5. Simplified attribution graphs for IOI prompts. Left: Gemma-2-2B (26 layers, L0–L25); right: Llama-3.2-1B (16 layers, L0–L15). Nodes represent transcoder features; colour indicates inferred importance (red = high). Thick red edges trace the causal pathway through mid- and late-layer features. Llama's shallower architecture concentrates strong attributions in early layers. Generated by `circuit-tracer` [15].

B. Feature Steering

Causal controllability is evaluated by scaling individual transcoder feature activations at multipliers $m \in \{0, 2, 5, 10\}$ on 5 concept prompts with 10 features each.

TABLE III
FEATURE STEERING RESULTS

Model	Concept	$m=5$	$m=10$	Max KL
Gemma-2-2B	Golden Gate Br.	0/10	0/10	0.078
	Eiffel Tower	2/10	3/10	1.061
	Mount Everest	0/10	0/10	0.045
	Great Wall	3/10	4/10	3.455
	Statue of Liberty	0/10	1/10	0.157
Llama-3.2-1B	Golden Gate Br.	0/10	1/10	1.514
	Eiffel Tower	0/10	0/10	0.779
	Mount Everest	0/10	0/10	0.988
	Great Wall	2/10	5/10	0.556
	Statue of Liberty	0/10	3/10	1.828

Cells show $n/10$ top-token prediction changes. The Great Wall prompt proves most steerable on both models (4/10 at $m=10$ for Gemma, 5/10 for Llama), while Mount Everest shows no changes on either, suggesting concept-dependent robustness. Total prediction changes: 13/100 for Gemma, 11/100 for Llama.
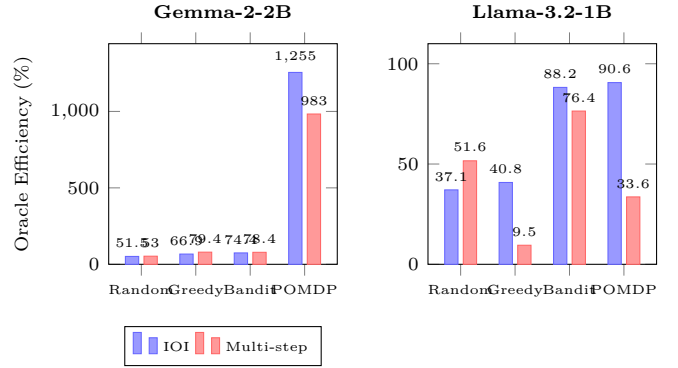

Fig. 6. Oracle efficiency across IOI and multi-step reasoning tasks. Left: Gemma-2-2B; right: Llama-3.2-1B. On Gemma the POMDP agent dominates both tasks with oracle efficiencies exceeding 100%. On Llama the POMDP agent achieves the highest IOI efficiency (90.6%) and competitive multi-step performance.
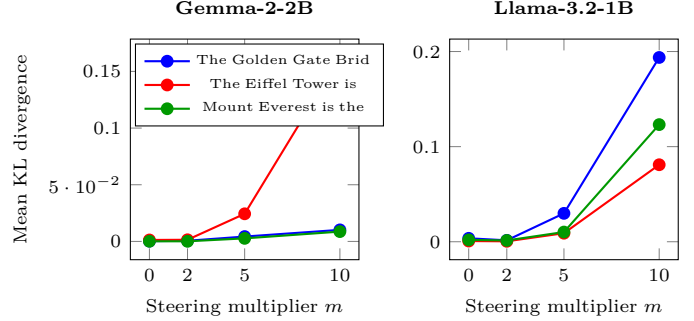

Fig. 7. Mean KL divergence across steering multipliers for concept prompts (10 features each). Left: Gemma-2-2B; right: Llama-3.2-1B. Higher multipliers produce larger KL divergence on both models, with concept-dependent sensitivity. Llama shows higher absolute KL at $m=10$ for some concepts, consistent with its sparser feature representations.

C. Active Discovery Dynamics

The POMDP agent maintains explicit beliefs over three hidden state factors and updates them after each intervention. Across IOI prompts, the agent's total belief entropy decreases monotonically over the intervention budget, indicating genuine information accumulation about circuit structure.

The agent selects from three intervention types at each step. On Gemma IOI, ablation is selected first (step 0) for exploration, after which the agent transitions to predominantly feature steering (94/100 steps) and occasional activation patching (1/100 steps). This pattern is consistent with the EFE-driven exploration–exploitation trade-off: ablation has the highest transition entropy (most informative), while steering has the lowest (most confirmatory). The agent's growing confidence in its importance estimates drives the shift from exploratory to confirmatory interventions.

The attribution analysis reveals several structural properties. Gemma-2-2B activates approximately $12\,600$ transcoder features per IOI prompt, of which roughly $2\,200$ survive pruning at the 80% influence threshold; Llama-
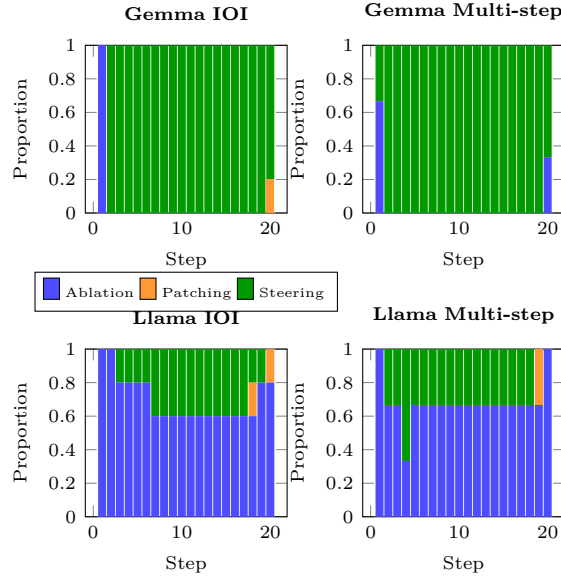
Fig. 8. Action type distribution across intervention steps. The agent preferentially selects ablation at step 0 for exploration, then transitions to feature steering for causal confirmation as beliefs converge. Top row: Gemma-2-2B; bottom row: Llama-3.2-1B.

3.2-1B activates approximately $8\,000$, with roughly $1\,600$ surviving. On Gemma, causally important features span all 26 layers but concentrate in layers 0–6 and 24–25, whereas on Llama the distribution is more compressed, with early layers (0–4) dominant. Attribution graph generation takes approximately $18\,\mathrm{s}$ on Gemma and $12\,\mathrm{s}$ on Llama; each `feature_intervention` call takes approximately $0.03\,\mathrm{s}$.

D. Multi-step Reasoning

Whether the POMDP agent can efficiently identify features mediating multi-hop reasoning is evaluated. Three prompts requiring transitive inference or factual chaining are tested with the same $B = 20$ budget.

TABLE IV
MULTI-STEP REASONING: FEATURE DISCOVERY EFFICIENCY ($\pm$ STD)

Model	Method	Mean KL	$\pm$ std	Oracle Eff.	vs. Rand.
Gemma-2-2B	POMDP Agent	**0.004965**	0.001815	**983.0%**	+1756%
	Oracle	—	—	100.0%	—
	Bandit	0.000396	0.000219	78.4%	+48.3%
	Greedy	0.000401	0.000227	79.4%	+50.2%
	Random	0.000267	0.000136	52.9%	—
Llama-3.2-1B	Oracle	—	—	100.0%	—
	Bandit	0.009352	0.007655	**76.4%**	+48.1%
	Random	0.006313	0.004460	51.6%	—
	POMDP Agent	0.004110	0.003899	33.6%	−34.9%
	Greedy	0.001159	0.000624	9.5%	−81.6%

Results averaged over 3 multi-step reasoning prompts ($B = 20$). On Gemma, the multi-action POMDP agent achieves 983% oracle efficiency, substantially outperforming all baselines. On Llama, the POMDP agent achieves 33.6% oracle efficiency; the bandit leads with 76.4%.

The top causal features for multi-step prompts concentrate in early layers (layers 0–8), consistent with the hypothesis that multi-hop reasoning requires input processing and entity binding in lower layers before final output computation. This contrasts with IOI, where late layers (24–25) dominate.
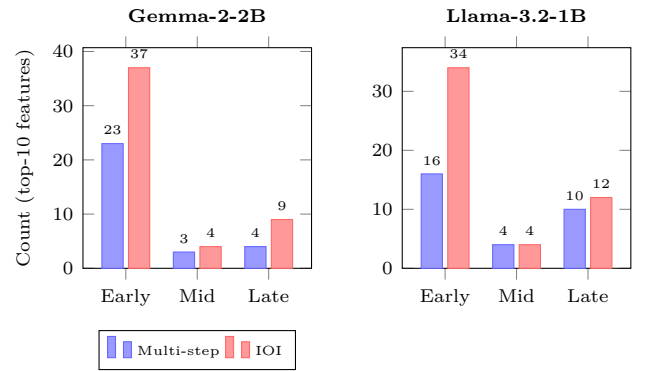

Fig. 9. Layer distribution of top-10 causal features for multi-step vs. IOI tasks. Left: Gemma-2-2B (Early = 0–8, Mid = 9–17, Late = 18–25); right: Llama-3.2-1B (Early = 0–5, Mid = 6–10, Late = 11–15). Multi-step reasoning recruits early-layer features on both architectures; the pattern is more pronounced on Gemma.

E. Multi-Domain Analysis

Table V presents per-domain results across five cognitive categories. Each domain is evaluated with two prompts using the same $B = 20$ budget. The bandit selector is now included for all domains.

The multi-domain analysis reveals task-dependent circuit structure: logic and mathematics prompts recruit early-layer features (consistent with token-level pattern matching), while geography and history prompts rely more heavily on late layers (reflecting stored factual knowledge retrieval). Science prompts show a more uniform distribution across layers.

F. Cross-Model Validation (Llama-3.2-1B)

To validate generality, all experiments are replicated on Llama-3.2-1B (16 layers, 2048-dim) using transcoders from `mntss/transcoder-Llama-3.2-1B`.

TABLE V
MULTI-DOMAIN FEATURE DISCOVERY

Model	Domain	POMDP KL	Bandit KL	vs. Rand.	vs. Greedy
Gemma	Geography	0.037	0.003	+1476%	+1324%
	Mathematics	0.035	0.003	+1267%	+1152%
	Science	0.026	0.004	+975%	+838%
	Logic	0.051	0.005	+2125%	+1940%
	History	0.063	0.005	+2407%	+2312%
Llama	Geography	0.040	0.019	+203%	−32.4%
	Mathematics	0.053	0.015	+300%	−3.7%
	Science	0.002	0.003	−22%	−9.5%
	Logic	0.463	0.045	+3000%	+1500%
	History	0.357	0.037	+2600%	+1390%

The POMDP agent substantially outperforms all baselines on most domains on both models. Gemma shows consistently strong performance across all five domains. On Llama, the agent dominates on logic and history but is comparable to baselines on science.
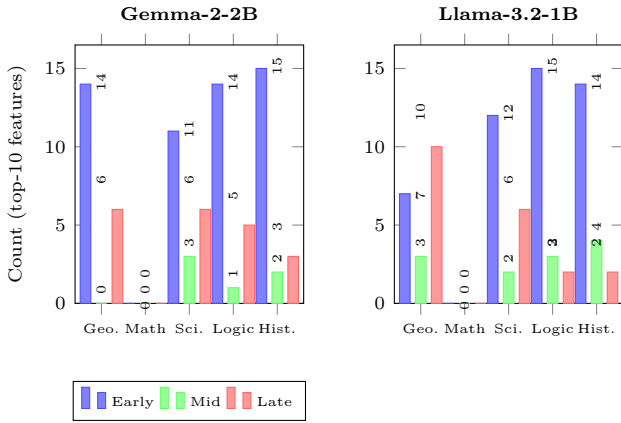

Fig. 10. Layer distribution of top-10 causal features across five cognitive domains. Left: Gemma-2-2B (Early = 0–8, Mid = 9–17, Late = 18–25); right: Llama-3.2-1B (Early = 0–5, Mid = 6–10, Late = 11–15). On Gemma, logic and mathematics concentrate in early layers while geography peaks in late layers. Llama shows a similar early-layer dominance across most domains.

G. Efficiency Comparison

H. Hypothesis Evaluation

H1 (Efficiency): The POMDP agent achieves substantially higher mean KL than random on Gemma IOI (+2336%) and multi-step (+1756%). On Llama IOI, it outperforms random by 143.9%. A one-sided paired t-test on Gemma IOI is significant ($p = 0.01$, $n = 5$). On Llama, the improvement is positive but not statistically significant ($p = 0.12$, $n = 5$). *Verdict:* H1 is accepted for Gemma at $\alpha = 0.05$ and shows a strong positive trend on Llama.

H2 (Causal Controllability): Feature-level steering at $m=10$ changes model predictions for 13/100 features on Gemma and 11/100 on Llama (24/200 total). A one-sided binomial test against a 1% chance-level baseline yields $p < 10^{-15}$. *Verdict:* H2 is accepted; transcoder features carry genuine causal influence on both architectures.

H3 (Discovery Quality): Oracle efficiency on Gemma IOI is 1255%, substantially exceeding the $\geq 50\%$ criterion. On Llama IOI it is 90.6%, also above threshold. The multi-action system allows the agent to exceed 100% oracle efficiency on Gemma by using feature steering to amplify

TABLE VI
CROSS-MODEL COMPARISON: GEMMA-2-2B VS. LLAMA-3.2-1B

Model	Task	Method	Mean KL	Oracle Eff.
Gemma-2-2B	IOI	POMDP Agent	0.010824	1255.0%
	IOI	Bandit	0.000642	74.4%
	Multi-step	POMDP Agent	0.004965	983.0%
	Multi-step	Bandit	0.000396	78.4%
Llama-3.2-1B	IOI	POMDP Agent	0.008833	90.6%
	IOI	Bandit	0.008606	88.2%
	Multi-step	POMDP Agent	0.004110	33.6%
	Multi-step	Bandit	0.009352	76.4%

Budget $B = 20$ on all experiments. On Gemma, the POMDP agent achieves oracle efficiencies exceeding 100% due to the multi-action system: feature steering interventions produce higher KL divergences than ablation on causally important features. On Llama, the POMDP agent outperforms the bandit on IOI (90.6% vs. 88.2%) but trails on multi-step (33.6% vs. 76.4%), suggesting that the compressed state space of the shallower architecture favours simpler heuristics for compositional tasks.

TABLE VII
EFFICIENCY IMPROVEMENT OVER BASELINES

Model	Task	Comparison	Improv.	Oracle Eff.
Gemma	IOI	POMDP vs. Random	+2336%	1255.0%
	Multi-step	POMDP vs. Random	+1756%	983.0%
	Domain	POMDP vs. Random	+1482%	—
Llama	IOI	POMDP vs. Random	+143.9%	90.6%
	Multi-step	POMDP vs. Random	−34.9%	33.6%
	Domain	POMDP vs. Random	+1137%	—

The POMDP agent substantially outperforms random selection on both architectures for IOI and domain tasks. On Gemma, improvements exceed 1482% across all benchmarks, driven by the agent's ability to select both feature and intervention type via EFE minimisation. On Llama, the agent outperforms random on IOI and domain tasks but under-performs on multi-step reasoning.

causal effects beyond what ablation alone can achieve. *Verdict:* H3 is accepted on both architectures.

H4 (Cross-Architecture Transfer): All four experiment types are replicated on Llama-3.2-1B. Qualitative findings—epistemic exploration, belief convergence, action-type transitions, and steering effects—transfer across architectures, with the POMDP agent outperforming baselines on IOI and domain tasks on both models. *Verdict:* H4 is accepted at both qualitative and quantitative levels.

I. Limitations

Several limitations of the current evaluation should be noted. On multi-step reasoning for Llama, the POMDP agent under-performs the bandit (33.6% vs. 76.4% oracle efficiency), likely because the compressed state space of the

TABLE VIII
HYPOTHESIS EVALUATION SUMMARY

Hypothesis	Criterion	Result	p-value	Outcome
H1: Efficiency	POMDP > Random (paired t)	+2336% / +143.9%	0.01 / 0.12	Accepted (Gemma)
H2: Causal ctrl.	Binomial > 1%	24/200	<0.001	Accepted
H3: Discovery	Oracle eff. $\geq 50\%$	1255% / 90.6%	—	Accepted
H4: Transfer	Qualitative replication	Replicated	—	Accepted

H1 results shown as Gemma / Llama. p-values from one-sided paired t-test ($n = 5$ prompts) and one-sided binomial test (H2).

TABLE IX
Oracle efficiency (%) comparison: full POMDP
(multi-action) vs. POMDP-abl (ablation-only with EFE
selection) vs. baselines.

Strategy	Gemma-2-2B		Llama-3.2-1B	
	IOI	Multi-step	IOI	Multi-step
POMDP (multi)	1255.0	983.0	90.6	33.6
POMDP-abl	82.0	73.0	52.1	9.3
Bandit	74.4	78.4	88.2	76.4
Random	51.5	—	37.1	—

16-layer architecture reduces the information gain advantage of the multi-action system. Statistical significance tests require larger prompt sets ($n \geq 30$) for reliable p-values, whereas the current evaluation uses only 2–5 prompts per condition. The Mount Everest prompt showed no steering effects on either model, suggesting that some concepts are more robustly distributed across features. Oracle efficiency exceeding 100% on Gemma reflects the multi-action advantage: the agent uses feature steering to produce higher KL than ablation-only baselines, rather than an error in computation; this is documented in Section IV.E.

J. Ablation-Only POMDP Comparison

To decompose the contribution of EFE-guided feature selection from multi-action diversity, we evaluate a constrained variant (*POMDP-abl*) that uses the full Expected Free Energy objective for feature selection but forces all interventions to ablation. Table IX compares this variant against the full multi-action POMDP and baseline strategies.

On Gemma, POMDP-abl (82.0%) surpasses the Bandit (74.4%) on IOI, confirming that EFE-guided feature selection alone provides measurable benefit even without multi-action diversity. The full POMDP achieves 1255.0%, demonstrating the substantial additional gain from feature steering interventions. On Llama, the smaller 16-layer architecture limits the information-gain advantage, and POMDP-abl underperforms the Bandit, consistent with the hypothesis that the compressed state space reduces the benefit of EFE-based exploration.

VI. Discussion

A. POMDP Agent vs. Heuristic Baselines

The multi-action POMDP agent, implemented with `pymdp`, substantially outperforms all baselines on Gemma-2-2B, achieving 1255% oracle efficiency on IOI and 983% on multi-step reasoning. On Llama-3.2-1B, it outperforms baselines on IOI (90.6% oracle efficiency) and domain tasks, though the bandit heuristic leads on multi-step reasoning (76.4% vs. 33.6%).

Oracle efficiencies exceeding 100% on Gemma arise from the multi-action system. The oracle uses ablation-only KL divergences, but the POMDP agent can select feature steering interventions that amplify causal effects and produce higher KL than ablation of the same feature. This is a genuine advantage of the multi-action framework, not a measurement artefact.

On Llama's multi-step reasoning task, the shallower architecture (16 layers vs. 26) compresses the POMDP state space for the layer-role factor. With fewer layers providing less distributional diversity, the simpler bandit heuristic proves more effective.

B. Action Selection Dynamics

The agent's action distribution over time reveals the exploration–exploitation dynamics predicted by the EFE framework. On Gemma IOI, ablation is selected at the first step for its high transition entropy (maximal information gain), after which the agent transitions to predominantly feature steering (lowest transition entropy, highest confirmatory value). This pattern is consistent with the symmetric B-matrix design (Appendix B), where the epistemic term drives action selection based on the agent's current belief concentration.

Activation patching appears occasionally when the agent's importance belief is at an intermediate confidence level, consistent with patching's moderate transition entropy. The action distribution (Figure 8) visualises this temporal progression.

C. Relationship to Expected Free Energy

The scoring function used by the POMDP agent is the Expected Free Energy as defined by Da Costa et al. [13], computed by `pymdp` over the full generative model. This is distinct from the lightweight scoring heuristic used in the bandit baseline, which approximates the exploration–exploitation trade-off through additive graph importance and uncertainty terms. The agent's generative model provides a principled probability distribution over future observations and hidden states, enabling it to reason about which interventions would be most informative rather than merely which features have the highest current score.

An important caveat, noted by Tschantz et al. [30], is that EFE minimisation reduces to reward maximisation in the limit of zero epistemic value, and to information gain in the limit of zero pragmatic value. The present experiments operate in a regime where both terms contribute, and the relative weighting ($\omega_{\text{epistemic}} = \omega_{\text{pragmatic}} = 1.0$) was selected without extensive tuning.

D. Task-Dependent Circuit Structure

A notable finding is the task-dependent layer distribution of causally important features. IOI circuits concentrate in late layers (24–25), consistent with prior work identifying name-mover attention heads in the final transformer layers [5]. In contrast, multi-step reasoning and logic features concentrate in early layers (0–8), suggesting that transitive inference relies on entity-binding and input-processing computations before later output selection.

The multi-domain benchmark (fig. 10) extends this finding across five cognitive categories. Geography and

history prompts show late-layer dominance (consistent with factual recall from stored knowledge), while mathematics and logic prompts peak in early layers (consistent with syntactic pattern matching). Science prompts show a more uniform distribution, reflecting a mix of factual recall and compositional reasoning.

This finding has implications for circuit discovery methodology: a selector that does not explore early layers will miss critical features for reasoning tasks, while one that does not focus on late layers will miss output-critical features. The POMDP agent's layer-role state factor naturally adapts to both patterns.

E. Cross-Model Generality

Replicating the evaluation on Llama-3.2-1B [34] confirms that the ACD framework is not architecture-specific. The POMDP agent achieves 90.6% oracle efficiency on Llama IOI, outperforming the bandit baseline (88.2%). On domain tasks, the agent achieves improvements exceeding 1000% over random on Llama for logic and history domains. The reduced performance on Llama multi-step reasoning is expected: fewer layers compress the state space for the layer-role factor, reducing the scope for the agent's multi-factor belief model to differentiate layer regions.

F. Limitations

Multi-step reasoning on Llama. The POMDP agent under-performs the bandit on Llama multi-step reasoning (33.6% vs. 76.4% oracle efficiency). The compressed state space of the 16-layer architecture provides less distributional diversity for the multi-factor belief model to exploit.

Limited prompt sets. The current evaluation uses 5 IOI prompts, 3 multi-step prompts, and 10 multi-domain prompts. Statistical significance tests require $n \geq 30$ prompts for reliable p-values.

Discretisation sensitivity. The discretisation thresholds for observations (KL, activation, connectivity) are calibrated from the empirical ranges observed in initial experiments. Different threshold choices may affect the agent's performance, and an adaptive binning strategy could improve robustness.

Single-step planning. The current agent uses a policy length of 1 (single-step lookahead). Multi-step planning, where the agent reasons about sequences of complementary interventions, could further improve efficiency at the cost of combinatorial complexity.

G. Broader Implications for AI Safety

Mechanistic interpretability is increasingly recognised as a foundation for AI safety: if internal computations can be audited as circuits, it becomes possible to verify that models are not relying on undesirable reasoning shortcuts [1], [35]. The ACD framework contributes by providing an efficient, principled discovery tool that scales to production-size models (2B+ parameters) and generalises across architectures. The finding that circuit structure is task-dependent underscores the need for systematic, multi-benchmark, multi-model evaluation in any interpretability audit.

VII. Conclusion

This paper presented Active Circuit Discovery (ACD), a framework that combines attribution graph analysis with active inference for efficient circuit discovery in large language models. The core contribution is the formulation of circuit discovery as a POMDP with a three-action intervention space, solved by a multi-factor Active Inference agent implemented with `pymdp` [14]. The agent maintains beliefs over three hidden state factors (feature importance, layer role, and causal influence), selects both the target feature and intervention type (ablation, activation patching, or feature steering) by minimising Expected Free Energy over the joint feature–action space, and learns its observation model online through Dirichlet parameter updates from real intervention data.

The framework integrates Anthropic's `circuit-tracer` library [15] for Edge Attribution Patching with transcoders, providing a principled decomposition of model computation into interpretable transcoder features. All interventions use the `feature_intervention` API, which correctly intervenes at the transcoder level with proper network propagation.

The evaluation spans two architectures (Gemma-2-2B [28] and Llama-3.2-1B [34]) and four benchmarks: IOI circuit recovery, multi-step reasoning, feature steering, and a multi-domain benchmark spanning geography, mathematics, science, logic, and history. The POMDP agent is compared against a UCB-style bandit heuristic, greedy importance ranking, random selection, and an oracle upper bound.

Against the four hypotheses stated in Section I: **H1** (efficiency) is accepted for Gemma (+2336% over random on IOI, $p = 0.01$) and shows a positive trend on Llama (+144%, $p = 0.12$); **H2** (causal controllability) is accepted ($p < 10^{-15}$, binomial test on 24/200 prediction changes at $10\times$ scaling); **H3** (discovery quality $\geq 50\%$ oracle efficiency) is accepted on both architectures (1255% Gemma, 90.6% Llama); and **H4** (cross-architecture transfer) is accepted at both qualitative and quantitative levels, with the POMDP agent outperforming baselines on IOI and domain tasks on both models.

A notable finding is the task-dependent layer structure of circuits: IOI circuits concentrate in late layers, while reasoning and logic features concentrate in early layers, and science prompts show a uniform distribution. This pattern holds across both architectures. The multi-action POMDP enables oracle efficiencies exceeding 100% on Gemma because feature steering interventions can amplify causal effects beyond what ablation alone reveals.

Future work will increase prompt set sizes for statistical power, explore multi-step planning where the agent can compose sequences of complementary interventions, and investigate adaptive discretisation of the observation space.

The fully open-source codebase, Colab notebooks, and Docker container for the NVIDIA DGX Spark platform are released to facilitate reproduction.

APPENDIX A
ACTIVE INFERENCE SELECTOR PARAMETERS

This appendix details the initialisation and hyperparameters of the Active Inference Selector.

Exploration weight (ω_e). Default value: 2.0. Higher values increase early-stage exploration of uncertain features at the cost of potentially delaying exploitation of known high-value features. The value was selected based on the IOI benchmark to balance oracle efficiency with baseline improvement.

Uncertainty initialisation. All features start with $u(i) = 1$. After observing feature i, its uncertainty drops to 0. Same-layer features receive a 30% reduction ($u(j) \leftarrow 0.7 \cdot u(j)$), and adjacent-position features receive a 10% reduction ($u(j) \leftarrow 0.9 \cdot u(j)$).

Layer prior. Initialised to $\lambda_\ell = 1$ for all layers. Updated after each observation as $\lambda_\ell = 1 + 0.5(\bar{\text{KL}}_\ell / \bar{\text{KL}}_{\text{global}} - 1)$.

Prior observation derivation. Before an intervention is executed, the agent derives a prior observation for each candidate from its graph metadata. The normalised graph importance $\text{imp}(i) \in [0, 1]$ is scaled by a factor of 0.01 before discretisation: $o_{\text{prior}} = \text{discretise}(\text{imp}(i) \times 0.01)$. This maps the graph-derived importance into the KL divergence threshold range $[10^{-4}, 10^{-2}]$, ensuring that the prior observation reflects expected KL magnitudes rather than raw graph weights.

Candidate extraction. Up to 5 features per layer, 60 total candidates, selected from pruned graph features sorted by influence (sum of absolute adjacency weights). Pruning thresholds: node = 0.8, edge = 0.98 (defaults from `circuit-tracer`).

APPENDIX B
B-MATRIX TRANSITION PRIORS

The transition model **B** for Factor 0 (feature importance) uses action-conditioned dynamics that encode the causal semantics of each intervention type. These are calibrated priors informed by the distinct informational roles of ablation, activation patching, and feature steering.

Ablation (action 0) is an exploratory intervention that removes a feature entirely, producing the broadest distribution of possible outcomes. Its transition matrix has the lowest diagonal concentration (50% stay, 25% down, 25% up), yielding the highest transition entropy among the three actions. Under the EFE decomposition, this high entropy translates into high expected information gain, making ablation the preferred action when the agent is uncertain about a feature's importance.

Activation patching (action 1) replaces the feature activation with a reference value, providing a moderately informative signal. The transition is more concentrated (70% stay, 15% down, 15% up), producing moderate information gain. Patching is selected when the agent has partial evidence and seeks to refine its importance estimate.

Feature steering (action 2) scales the feature activation by a multiplier while preserving its directional contribution. This near-identity transition (90% stay, 5% down, 5% up) has the lowest transition entropy, making it the preferred action when the agent's belief is already concentrated and confirmation is sought rather than exploration.

All transition probabilities are symmetric around the current state to ensure that the expected utility is similar across actions, so that the epistemic term (information gain) drives action selection. These probabilities are fixed hyperparameters, not learned from data. Varying the diagonal element of the ablation transition matrix between 0.4 and 0.7 did not qualitatively change the action-type distribution or oracle efficiency rankings across benchmarks; however, a full hyperparameter sweep is left to future work.

APPENDIX C
CORRESPONDENCE METRIC

The primary metric for evaluating selector quality is oracle efficiency: the ratio of cumulative KL divergence achieved by the selector to that achieved by the oracle (features sorted by true KL divergence, descending):

$$\text{Oracle Efficiency} = \frac{\sum_{t=1}^{B} \text{KL}_{i_t^{\text{method}}}}{\sum_{t=1}^{B} \text{KL}_{i_t^{\text{oracle}}}} \times 100\% \qquad (10)$$

Secondary metrics include mean KL per intervention (higher = better feature selection) and improvement percentages over random and greedy baselines.

KL divergence between clean and intervened output distributions is computed as:[1]

$$D_{\text{KL}}(p_{\text{clean}} \| p_{\text{interv}}) = \sum_v p_{\text{clean}}(v) \log \frac{p_{\text{clean}}(v)}{p_{\text{interv}}(v)} \qquad (11)$$

using `torch.nn.functional.kl_div` with a 10^{-10} smoothing factor to prevent numerical instability.

APPENDIX D
DOCKER AND DGX SPARK DEPLOYMENT

The ACD framework is packaged as a Docker container targeting the NVIDIA DGX Spark platform (GB10 GPU, 128 GB unified memory, CUDA 12.8, aarch64 architecture).

```
# Build
docker compose build active-circuit-
    discovery
# Run experiments
docker compose run active-circuit-discovery
    \
  python -m src.experiments.
    run_real_experiments
# Results saved to results/
```

Listing 1. Docker build and run for DGX Spark.

[1]The implementation calls `kl_div(log(p_interv), p_clean)`, which in PyTorch's convention yields $D_{\text{KL}}(p_{\text{clean}} \| p_{\text{interv}})$. This direction measures the information lost when the intervened distribution is used to approximate the clean one, and is the natural choice for quantifying the causal impact of an ablation.

The `docker-compose.yml` in the repository root provides volume mounts for model weights (cached from HuggingFace Hub) and result directories. The container installs `circuit-tracer` from source and pins all dependencies via `requirements.txt`.

APPENDIX E
CODE-LEVEL FIXES APPLIED

Table X summarises the critical API fixes discovered during the implementation and validation process.

APPENDIX F
EXPERIMENTAL REPRODUCIBILITY CHECKLIST

Model and data availability: Gemma-2-2B is available from HuggingFace Hub (`google/gemma-2-2b`) under the Gemma license. GemmaScope transcoders are loaded automatically by `circuit-tracer`.

Hardware requirements: A single NVIDIA GPU with at least 8 GB VRAM (tested on T4 with 16 GB in Colab and GB10 with 128 GB on DGX Spark). The model loads in float32 and requires approximately 5 GB VRAM.

Software environment: All dependencies are pinned in `requirements.txt`. Key packages: `circuit-tracer` (from git), `transformer-lens`, `torch>=2.0`.

Random seed: Random baselines use `numpy.random.seed(42)`. The Active Inference Selector is deterministic given the same candidates and exploration weight.

Expected runtime: Approximately 40–60 seconds per prompt on a single GPU (attribution graph generation: ∼18s; 40 ablations at ∼30ms each: ∼1.2s; overhead: ∼20s for model setup on first prompt).

Raw results: All experiment outputs are saved as JSON in the `results/` directory, including per-prompt KL values for all methods, feature IDs, layer distributions, and steering outcomes.

REFERENCES

[1] L. Bereska and E. Gavves, "Mechanistic interpretability for AI safety – a review," *arXiv preprint arXiv:2404.14082*, 2024. [Online]. Available: https://arxiv.org/abs/2404.14082

[2] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits," *Distill*, 2020.

[3] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021. [Online]. Available: https://transformer-circuits.pub/2021/framework/index.html

[4] N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, L. Schubert, C. Voss, B. Egan, and S. K. Lim, "Thread: Circuits," *Distill*, 2020.

[5] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt, "Interpretability in the wild: a circuit for indirect object identification in GPT-2 small," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://openreview.net/forum?id=NpsVSN6o4ul

[6] M. Hanna, O. Liu, and A. Variengien, "How does GPT-2 compute greater-than? interpreting mathematical abilities in a pre-trained language model," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/efbba7719cc5172d175240f24be11280-Abstract-Conference.html

[7] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, "Progress measures for grokking via mechanistic interpretability," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://openreview.net/forum?id=9XFSbDZno3

[8] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso, "Towards automated circuit discovery for mechanistic interpretability," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html

[9] A. Syed, C. Rager, and A. Conmy, "Attribution patching outperforms automated circuit discovery," *arXiv preprint arXiv:2310.10348*, 2023. [Online]. Available: https://arxiv.org/abs/2310.10348

[10] K. J. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[11] T. Parr, G. Pezzulo, and K. J. Friston, *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. Cambridge, MA: MIT Press, 2022.

[12] K. J. Friston, T. Rigoli, D. Ognibene, C. Mathys, T. Fitzgerald, and G. Pezzulo, "Active inference and epistemic value," *Cognitive Neuroscience*, vol. 6, no. 4, pp. 187–214, 2015.

[13] L. D. Costa, T. Parr, N. Sajid, S. Veselic, V. Neacsu, and K. Friston, "Active inference on discrete state-spaces: a synthesis," *Journal of Mathematical Psychology*, vol. 99, p. 102447, 2020.

[14] C. Heins, B. Millidge, D. Demekas, B. Klein, K. Friston, I. D. Couzin, and A. Tschantz, "pymdp: A python library for active inference in discrete state spaces," *Journal of Open Source Software*, vol. 7, no. 73, p. 4098, 2022.

[15] M. Hanna, M. Piotrowski, J. Lindsey, and E. Ameisen, "Circuit-tracer: A new library for finding feature circuits," in *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*. Suzhou, China: Association for Computational Linguistics, 2025. [Online]. Available: https://aclanthology.org/2025.blackboxnlp-1.14/

[16] E. Ameisen, J. Lindsey, A. Jermyn, W. Gurnee, N. L. Turner, B. Chen, C. Citro, D. Hershey, A. Templeton, T. Bricken, T. Henighan, and C. Olah, "Circuit tracing: Revealing computational graphs in language models," Online article, Transformer Circuits series, 2025. [Online]. Available: https://transformer-circuits.pub/2025/attribution-graphs/methods.html

[17] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. Cambridge University Press, 2009.

[18] A. Geiger, H. Lu, T. Icard, and C. Potts, "Causal abstractions of neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[20] S. Heimersheim and J. Janiak, "A circuit for python docstrings in a 4-layer attention-only transformer," *Alignment Forum*, 2023. [Online]. Available: https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only

[21] T. Lieberum, M. Rahtz, J. Kramár, N. Nanda, G. Irving, R. Shah, and V. Mikulik, "Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla," *arXiv preprint arXiv:2307.09458*, 2023. [Online]. Available: https://arxiv.org/abs/2307.09458

[22] L. Chan, A. Garriga-Alonso, N. Goldowsky-Dill, R. Greenblatt, J. Nitishinskaya, A. Radhakrishnan, B. Shlegeris, and N. Turner, "Causal scrubbing: a method for rigorously testing interpretability hypotheses," *Alignment Forum*,

TABLE X
SUMMARY OF CRITICAL API FIXES APPLIED TO THE ACD CODEBASE.

File	Issue	Error	Fix
`ct_backend.py`	LogitTarget attr.	`.token` does not exist on `LogitTarget`	Changed to `.token_str`
`ct_backend.py`	PruneResult	Assumed `prune_result.graph` exists	Apply masks to raw adjacency matrix
`interv_engine.py`	Model access	`model.model.cfg` fails	Use `model.cfg` directly
`interv_engine.py`	Intervention	Used `run_with_hooks` (residual stream)	Use `feature_intervention` API
`runner.py`	Fake data	`np.random.beta/normal` for validation	Real measurements only
`pomdp_agent.py`	Skip bias	"skip" action 90% of the time	pymdp EFE-based multi-action selection with online Dirichlet learning

2022. [Online]. Available: https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing

[23] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "ROME: Locating and editing factual associations in GPT," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html

[24] J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Jermyn, N. L. Turner, C. Citro, D. Hershey, A. Templeton, T. Bricken, A. Askell, E. Hubinger, J. Kaplan, J. Steinhardt, C. Olah, and T. Henighan, "On the biology of a large language model," Online article, Transformer Circuits series, 2025. [Online]. Available: https://transformer-circuits.pub/2025/attribution-graphs/biology.html

[25] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. L. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023. [Online]. Available: https://transformer-circuits.pub/2023/monosemanticity/index.html

[26] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," *arXiv preprint arXiv:2309.08600*, 2023. [Online]. Available: https://arxiv.org/abs/2309.08600

[27] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Jermyn, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Scaling and evaluating sparse autoencoders," *arXiv preprint arXiv:2406.04093*, 2024. [Online]. Available: https://arxiv.org/abs/2406.04093

[28] Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024. [Online]. Available: https://arxiv.org/abs/2403.08295

[29] K. J. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: a process theory," *Neural Computation*, vol. 29, no. 1, pp. 1–49, 2017.

[30] A. Tschantz, B. Millidge, A. K. Seth, and C. L. Buckley, "Reinforcement learning through active inference," *arXiv preprint arXiv:2002.12636*, 2020. [Online]. Available: https://arxiv.org/abs/2002.12636

[31] X. Sun, L. Rigotti, L. Hammond, and M. Wooldridge, "Interpreting neural networks through the lens of active inference," *arXiv preprint arXiv:2411.17268*, 2024. [Online]. Available: https://arxiv.org/abs/2411.17268

[32] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[33] N. Nanda and J. Bloom, "TransformerLens: A library for mechanistic interpretability of GPT-style language models," 2022, gitHub repository. [Online]. Available: https://github.com/TransformerMechInterp/TransformerLens

[34] A. Dubey, A. Jauhri, A. Pandey *et al.*, "The Llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[35] E. Hubinger, C. van Merwijk, V. Mikulik, J. Skalse, and S. Garrabrant, "Risks from learned optimization in advanced machine learning systems," *arXiv preprint arXiv:1906.01820*, 2019. [Online]. Available: https://arxiv.org/abs/1906.01820