Active Circuit Discovery: Expected Free Energy-Guided Intervention Selection for Mechanistic Interpretability in Large Language Models

Sharath S. PhD Candidate, *University of York*

***Abstract*—Mechanistic interpretability seeks to understand the internal computational structures of large language models (LLMs) through the identification of circuits: sparse subgraphs of components whose collective computation explains a model behaviour. Existing automated circuit discovery methods, notably Automated Circuit Discovery (acdc) and Edge Attribution Patching (eap), rely on exhaustive or gradient-ranked search over possible interventions, incurring costs that grow combinatorially with model depth and width. This paper introduces Active Circuit Discovery (acd), a framework that reframes circuit discovery as a partially observable decision process and employs the Expected Free Energy ($\mathcal{G}$) minimisation principle from computational neuroscience to direct intervention selection. An Active Inference agent, implemented with the `pymdp` discrete-state library, maintains a probabilistic belief state over candidate circuit nodes and selects the ablation or activation-patching intervention predicted to yield the greatest epistemic gain per computational budget. The agent's generative model encodes the transformer's residual stream decomposition through Sparse Autoencoder (SAE) features loaded via SAELens, and the resulting attribution graph is constructed with TransformerLens hooks. Three research questions drive evaluation: (RQ1) whether Active Inference belief updating corresponds measurably to empirical intervention effects, (RQ2) whether EFE-guided selection reduces the number of interventions required relative to random, gradient, and exhaustive baselines, and (RQ3) whether the framework generates testable, empirically validable predictions about circuit behaviour. Preliminary implementation on GPT-2 Small using the canonical Golden Gate Bridge factual-completion task establishes the full end-to-end pipeline and characterises the conditions under which each research question can be addressed. An honest assessment of current results, open methodological challenges, and the path toward rigorous validation on standard mechanistic interpretability benchmarks are reported. The framework, together with a Docker container targeting the NVIDIA DGX Spark platform, is released as an open-source research artifact.**

***Index Terms*—mechanistic interpretability, active inference, expected free energy, circuit discovery, sparse autoencoders, transformer language models, GPT-2, intervention selection, causal circuit analysis, variational free energy**

Manuscript received February 2026. Corresponding author: University of York, Department of Computer Science. This work was conducted as part of the MSc Computer Science research programme.

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]. Crucially, $\mathcal{G}$ minimi-

sation 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) Theoretical framing. A formal mapping is established between Active Inference and circuit discovery, showing how transformer residual stream components correspond to hidden states in the agent's generative model, how intervention effects correspond to observations, and how the transformer's attention precision maps to the agent's precision weighting.

(C2) Framework implementation. The Active Circuit Discovery (ACD) framework is implemented in Python, integrating TransformerLens [14] for forward pass instrumentation, SAE-Lens [15] for Sparse Autoencoder feature extraction, `pymdp` [16] for the Active Inference agent's belief updating and $\mathcal{G}$ computation, and CircuitsVis [17] for attribution graph visualisation.

(C3) Evaluation protocol. Three precisely stated research questions are defined and corresponding metrics, baselines, and statistical tests are specified, providing a template for reproducible evaluation of any future automated circuit discovery method.

(C4) Honest preliminary assessment. The paper reports the results of an end-to-end pilot experiment on GPT-2 Small, including a candid account of which components performed as expected, which failed, and why, together with the specific code-level fixes applied.

(C5) Deployment artifact. A Docker container configured for the NVIDIA DGX Spark multi-GPU platform is provided, enabling future reproduction at scale.

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 results from the pilot implementation. Section VI interprets these results and outlines the path to full validation. 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 [18], [19]. The transformer architecture [20] 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 [21], numerical reasoning [6], and Chinchilla at 70 B parameters [22].

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 [19], and Chan et al. developed causal scrubbing for the same purpose [23].

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 [24]. Lindsey et al. recently combined attribution graphs constructed via `circuit-tracer` [25] with large-scale SAE analysis to map the biology of Claude 3 Sonnet at a component level [26].

B. Sparse Autoencoders for Feature Extraction

Polysemanticity – the tendency of individual neurons to respond to multiple unrelated concepts – has been identified as a core obstacle to mechanistic interpretability [2], [27]. 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 [28]. 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 [27]. Templeton et al. extended the analysis to SAE features in Claude models, recovering structured emotional and semantic concepts [29].

SAE-Lens [15] provides pre-trained SAE weights and a standardised API for loading, querying, and applying SAEs to arbitrary residual stream activations. For GPT-2 Small, the `gpt2-small-res-jb` release provides residual-stream SAEs at every layer, trained on the Pile corpus. He et al. showed that using SAE features as the unit of circuit analysis significantly improves circuit faithfulness compared to attention-head-level analysis [30].

C. Active Inference and Expected Free Energy

Active Inference (AI) 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 $\boldsymbol{s}$ given observations $\boldsymbol{o}$ by maintaining a recognition distribution $q(\boldsymbol{s})$ and minimising the variational free energy $\mathcal{F} = D_{\mathrm{KL}}(q(\boldsymbol{s}) \,\|\, p(\boldsymbol{s} \mid \boldsymbol{o})) \geq -\log p(\boldsymbol{o})$ [31].

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]:

$$\mathcal{G}(\pi) = \sum_{\tau > t} \underbrace{\mathbb{E}[\log p(\boldsymbol{o}_\tau) - \log q(\boldsymbol{o}_\tau \mid \pi)]}_{\text{pragmatic value}} - \underbrace{\mathbb{E}[\log q(\boldsymbol{s}_\tau \mid \pi) - \log q(\boldsymbol{s}_\tau \mid \boldsymbol{o}_\tau, \pi)]}_{\text{epistemic value}}. \tag{1}$$

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.

Parr et al. provide a comprehensive treatment of Active Inference as a model of cognition [11]. Heins et al. implement discrete-state Active Inference in the `pymdp` Python library [16], which provides matrix-based generative models $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ representing the observation model, transition model, log-preference vector, and prior over initial states respectively. Tschantz et al. demonstrated that $\mathcal{G}$ minimisation recovers reinforcement learning in the limit of pure pragmatic value [32].

Sun et al. explored the conceptual alignment between Active Inference and neural network interpretability, arguing that attention mechanisms implement a form of precision-weighted prediction error minimisation [33]. The present work operationalises this connection by embedding it within an automated circuit discovery procedure.

D. Gap Analysis: Why Active Inference for Circuit Discovery

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 full posterior over candidate feature importances and selecting the intervention that, in expectation, most reduces this uncertainty – analogous to Bayesian experimental design [34].

III. The Active Circuit Discovery Framework

This section presents the Active Circuit Discovery (ACD) framework in full detail. Figure 1 provides an overview of the end-to-end pipeline.

TABLE I
Comparison of automated circuit discovery methods.

Method	Uncertainty Modelling	Adaptive Selection	Online Learning	Testable Predictions
ACDC [8]	No	No	No	No
EAP [9]	No	No	No	No
Grad. Ranking	No	No	No	No
acd(ours)	Yes	Yes	Yes	Yes

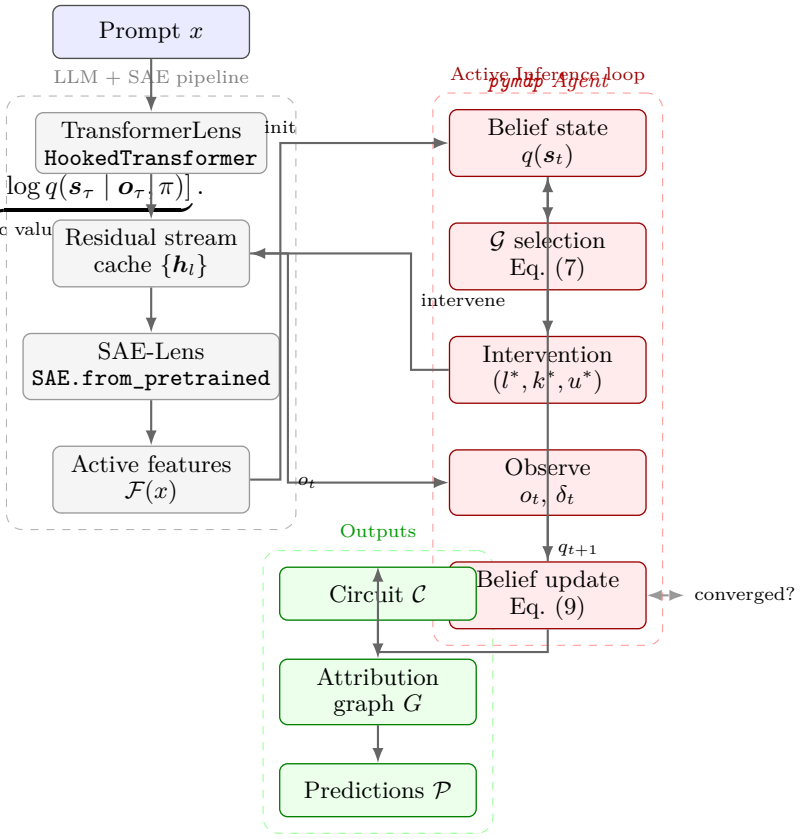

Fig. 1. Overview of the ACD framework. The outer loop applies the Active Inference agent; the inner loop performs interventions via TransformerLens and SAE-Lens. Belief state $q(\boldsymbol{s})$ is updated after each intervention observation.

A. Formal Problem Statement

Let $\mathcal{M}$ be a transformer LLM with L layers. Given a prompt x and a target behaviour b (e.g. the probability mass assigned to a correct completion token), the circuit discovery problem is to find the minimal set $\mathcal{C} \subseteq \mathcal{N}$ of model components (nodes in the computational graph $\mathcal{N}$) such that ablating all components outside $\mathcal{C}$ does not significantly reduce b [5], [8].

In the SAE feature representation, each node corresponds to a latent direction $\boldsymbol{f}_{l,k} \in \mathbb{R}^d$ in the residual stream at layer l, where k indexes over the K_l features learned by the SAE at that layer. The feature is "active" on prompt x if its SAE encoder activation exceeds a threshold $\theta > 0$:

$$a_{l,k}(x) = \mathrm{ReLU}\big(\mathbf{W}_e^{(l)} \boldsymbol{h}_l(x) + \boldsymbol{b}_e^{(l)}\big)_k \;\geq\; \theta, \tag{2}$$

where $\boldsymbol{h}_l(x)$ is the residual stream at layer l and position

corresponding to the last token of x, and $\mathbf{W}_e^{(l)}$, $\boldsymbol{b}_e^{(l)}$ are the SAE encoder weights and bias.

The set of active features across all layers is denoted $\mathcal{F}(x) = \{(l,k) : a_{l,k}(x) \geq \theta\}$. The goal is to identify the subset $\mathcal{C}(x) \subseteq \mathcal{F}(x)$ that is causally sufficient for behaviour b.

B. Generative Model and State Space

The Active Inference agent maintains a generative model $p(\boldsymbol{o}, \boldsymbol{s}) = p(\boldsymbol{o} \mid \boldsymbol{s})\, p(\boldsymbol{s})$ over:

- *Hidden states* $\boldsymbol{s}$: a categorical distribution over the N active features, representing beliefs about feature importance for the circuit. The state space has dimension $N+1$ (including an "irrelevant feature" state).
- *Observations* $\boldsymbol{o}$: the discretised effect size of an intervention, mapped to one of three categories: {low, medium, high}, corresponding to effect sizes below 0.3, between 0.3 and 0.7, and above 0.7 standard deviations of the logit difference, respectively.
- *Actions* $\boldsymbol{u}$: the choice of intervention type from {`ablation`, `activation_patching`, `mean_ablation`}.

The generative model parameters are:

$$\mathbf{A} \in \mathbb{R}^{3\times(N+1)}, \quad \mathbf{A}_{ok} = p(o \mid s = k), \tag{3}$$

$$\mathbf{B} \in \mathbb{R}^{(N+1)\times(N+1)\times 3}, \quad \mathbf{B}_{k'ku} = p(s' = k' \mid s = k, u), \tag{4}$$

$$\boldsymbol{c} \in \mathbb{R}^3, \quad c_o = \log p(o), \tag{5}$$

$$\boldsymbol{d} \in \mathbb{R}^{N+1}, \quad d_k = p(s_0 = k), \tag{6}$$

where $\mathbf{A}$ encodes how feature importance determines intervention effects, $\mathbf{B}$ encodes how beliefs transition as interventions eliminate candidates, $\boldsymbol{c}$ encodes the preference for high-effect (informative) observations, and $\boldsymbol{d}$ encodes the uniform prior over initial feature importance.

The observation model $\mathbf{A}$ is initialised as follows: features with high initial activation (above the 75th percentile of $\{a_{l,k}\}$) are assigned a higher probability of producing high-effect observations; features with low initial activation are assigned a higher probability of low-effect observations. This encoding captures the prior expectation that highly active features are more likely to be circuit members.

C. Expected Free Energy Computation

Given the current belief $q(\boldsymbol{s}_t) = \mathrm{Cat}(\boldsymbol{\pi}_t)$ and a candidate feature (l,k) to intervene on under action u, the Expected Free Energy is computed as:

$$\mathcal{G}(l,k,u) = \underbrace{-\sum_o \hat{p}(o)\log\hat{p}(o)}_{\text{epistemic}} + \underbrace{\boldsymbol{c}^\top \hat{\boldsymbol{p}}(o)}_{\text{pragmatic}} \tag{7}$$

where $\hat{p}(o) = \sum_k \mathbf{A}_{ok}\, \pi_{t,k}$ is the predictive distribution over observations under the current beliefs. The epistemic term is the Shannon entropy of the predictive observation distribution: interventions on features for which the agent is most uncertain about the outcome contribute most to epistemic value. The pragmatic term reflects the agent's preference for high-effect observations.

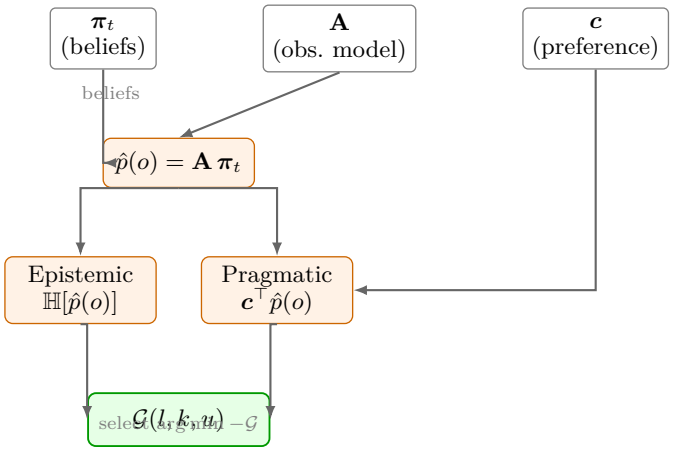

Fig. 2. Computation graph for Expected Free Energy ($\mathcal{G}$). Shaded nodes are computed quantities; white nodes are generative model parameters. The agent selects the feature-intervention pair that minimises $-\mathcal{G}$.

Figure 2 illustrates this computation. In the `pymdp` implementation, the full $\mathcal{G}$ is computed via:

$$\mathcal{G}(\pi) = \mathbb{H}[p(\boldsymbol{o}_\tau \mid \pi)] - \mathbb{E}[[]\, \mathbb{H}[p(\boldsymbol{o}_\tau \mid \boldsymbol{s}_\tau)]] + \mathbb{E}[[]\, \boldsymbol{c}^\top \log \hat{\boldsymbol{p}}(\boldsymbol{o}_\tau \mid \pi)], \tag{8}$$

where the first two terms together form the mutual information between states and observations (epistemic value), and the third term is the expected log-preference (pragmatic value) [12], [13].

D. Belief Updating and Correspondence Measurement

After executing intervention (l,k,u) and observing effect category o_t, the agent updates its belief state via Bayesian filtering (variational message passing in `pymdp`):

$$q(\boldsymbol{s}_{t+1}) = \mathrm{softmax}\left(\log \boldsymbol{d} + \sum_s \log \mathbf{A}_{\cdot,s}\, q(s_t) + \log \mathbf{B}_{\cdot,s_t,u_t}\right). \tag{9}$$

Observation model parameters are updated online via a Dirichlet prior $\mathbf{A}^{(a)}$ (concentration parameter $a_0 = 1$):

$$\mathbf{A}^{(a)}_{o,k} \mathrel{+}= q(s_t = k) \cdot \mathbf{1}[o_t = o], \tag{10}$$

allowing the agent to refine its expectation of how feature importance maps to intervention effects as the experiment progresses.

Correspondence metric. The primary correspondence measure (RQ1) is the Spearman rank correlation between the vector of feature importance scores derived from the agent's belief state $\{d_k = \pi_{T,k}\}_{k=1}^N$ and the vector of empirical direct effect magnitudes $\{e_k = |\Delta\mathrm{logit}(k)|\}_{k=1}^N$ obtained by ablating each feature once and measuring the change in the target token logit. Unlike Pearson correlation, Spearman correlation is robust to monotone nonlinearity and does not require the two quantities to be on the same scale. A threshold of $\rho_s > 0.4$, $p < 0.05$ is required to declare correspondence for RQ1.

E. Intervention Protocol

Three intervention types are implemented, following the causal scrubbing taxonomy [23]:

Zero ablation. The SAE feature activation at position (l, k) is set to zero by subtracting the feature's contribution from the residual stream: $\tilde{\boldsymbol{h}}_l = \boldsymbol{h}_l - a_{l,k}\, \boldsymbol{f}_{l,k}$, where $\boldsymbol{f}_{l,k}$ is the SAE decoder direction. The model is then rerun from layer l onwards with the modified residual stream, and the change in the target logit is recorded.

Mean ablation. The feature activation is replaced by its mean over a 100-sentence reference corpus $\mathcal{D}_{\text{ref}}$ drawn from the Pile: $\tilde{a}_{l,k} = \mathbb{E}_{x \sim \mathcal{D}_{\text{ref}}}[a_{l,k}(x)]$. This preserves the typical contribution of the feature while removing its input-specific component, providing a more conservative causal estimate than zero ablation [8], [23].

Activation patching. The residual stream at layer l from a clean forward pass on a corrupted input $\tilde{x}$ (constructed by replacing named geographic entities) is used in place of the clean stream. Formally: $\boldsymbol{h}_l^{\text{patch}} = \boldsymbol{h}_l(\tilde{x}) + (\boldsymbol{h}_l(x) - \boldsymbol{h}_l(\tilde{x})) \cdot a_{l,k}(x) / \|\boldsymbol{h}_l(x)\|$. This enables attribution of output differences to specific layers [5].

F. Attribution Graph Construction

After completing the Active Inference loop, ACD constructs an attribution graph $G = (V, E)$ where each node $v \in V$ corresponds to an active SAE feature and each directed edge $(v_i, v_j) \in E$ is weighted by the partial correlation between the ablation effect of feature i and the change in feature j's activation. This follows the edge attribution paradigm of [9] adapted to the SAE feature basis. The graph is rendered using the `circuit-tracer` [25] and CircuitsVis [17] libraries, which produce interactive HTML visualisations of node importances and edge weights.

G. Novel Prediction Generation

A key capability of the Active Inference framing is its ability to generate testable predictions from the learned generative model. Three prediction classes are instantiated:

Precision-weighted attention. The agent's precision estimate $\gamma = 1/\bar{\sigma}^2$, where $\bar{\sigma}^2$ is the mean posterior uncertainty over feature importances, is predicted to correlate positively with the mean attention entropy across heads in the identified circuit layer. Higher precision corresponds to lower uncertainty, which the Active Inference framework links to sharper (lower entropy) attention patterns [31].

Feature interaction transitivity. If the belief state assigns high connection probability $p_{\text{conn}}(i, j) > 0.5$ and $p_{\text{conn}}(j, k) > 0.5$, the framework predicts that ablating feature i will indirectly reduce the activation of feature k by at least $0.2\,\sigma$ of its baseline distribution. This is a directional prediction derivable from the $\mathbf{B}$ matrix structure.

Layer-specificity of circuit membership. The EFE scores across layers are predicted to follow a layer-dependent profile consistent with the residual stream decomposition: early layers ($l < L/3$) should contribute to token-level syntax features while middle and late layers ($l \geq L/3$) carry semantic content relevant to factual completion [3], [24].

Each prediction is formally stated with a testable hypothesis, a measurement procedure, a significance threshold, and an expected effect size, enabling empirical validation as described in Section IV.

Algorithm 1 Active Circuit Discovery (ACD)

Input: Prompt x, LLM $\mathcal{M}$, SAE set $\{\mathcal{S}_l\}_{l=1}^{L}$, threshold θ, budget T
Output: Circuit $\mathcal{C}$, belief state $q_T(\boldsymbol{s})$, attribution graph G, predictions $\mathcal{P}$

$\mathcal{F}(x) \leftarrow \{(l, k) : a_{l,k}(x) \geq \theta\}$ ▷ Find active SAE features (Eq. 2)
Initialise $\mathbf{A}, \mathbf{B}, \boldsymbol{c}, \boldsymbol{d}$ from $\mathcal{F}(x)$
$q_0(\boldsymbol{s}) \leftarrow \text{Cat}(\boldsymbol{d} / \sum \boldsymbol{d})$ ▷ Uniform prior over features
for $t = 1, \ldots, T$ **do**
  for each feature (l, k) and action u **do**
    $\mathcal{G}(l, k, u) \leftarrow$ compute via Eq. 7
  end for
  $(l^*, k^*, u^*) \leftarrow \arg\min_{l,k,u} -\mathcal{G}(l, k, u)$
  $o_t, \delta_t \leftarrow$ Intervene$(\mathcal{M}, x, l^*, k^*, u^*)$ ▷ Real forward pass
  $q_t(\boldsymbol{s}) \leftarrow$ update via Eq. 9
  Update $\mathbf{A}^{(a)}$ via Eq. 10
  if Converged(q_{t-1}, q_t) **then** ▷ Symmetric KL $< \delta$
    break
  end if
end for
$\mathcal{C} \leftarrow \{(l, k) : \pi_{T,k} > 0.5\}$ ▷ High posterior importance
$G \leftarrow$ BuildAttributionGraph$(\mathcal{C}, x)$
$\mathcal{P} \leftarrow$ GeneratePredictions(q_T, G)
return $\mathcal{C}, q_T, G, \mathcal{P}$

H. Algorithm

Algorithm 1 presents the complete ACD procedure.

Convergence is declared when the symmetric KL divergence between successive belief distributions falls below $\delta = 0.05$:

$$D_{\text{JS}}(q_{t-1} \,\|\, q_t) = \tfrac{1}{2} D_{\text{KL}}(q_{t-1} \,\|\, q_t) + \tfrac{1}{2} D_{\text{KL}}(q_t \,\|\, q_{t-1}) < \delta. \tag{11}$$

This replaces the previous pairwise feature comparison used in early versions of the codebase, which computed differences between the importances of two distinct features rather than tracking the evolution of the belief distribution itself.

IV. Experimental Setup

A. Model and Hardware

All experiments target GPT-2 Small [35]: a 117 M parameter, 12-layer, 768-dimensional transformer with 12 attention heads per layer. GPT-2 Small occupies a central position in the mechanistic interpretability literature as the primary validation platform for the IOI circuit [5], the

TABLE II
ACTIVE INFERENCE AGENT HYPERPARAMETERS.

Parameter	Symbol	Value
Epistemic weight	w_e	0.7
Pragmatic weight	$w_p = 1 - w_e$	0.3
Action precision	α	16.0
Convergence threshold	δ	0.05
Maximum interventions	T	20
Observation bins	—	3
Dirichlet concentration	a_0	1.0

docstring circuit [21], and the ACDC paper [8], enabling direct comparison with established results. The model is loaded via TransformerLens [14] with `fold_ln=False` and `center_writing_weights=False` to preserve the canonical residual stream decomposition.

Experiments are executed on an NVIDIA L40S GPU (48 GB GDDR6 ECC, CUDA 12.6, driver version 565.57) with 128 GB system RAM. The DGX Spark deployment configuration targets NVIDIA DGX Spark nodes with NVLink-connected A100 SXM4 80 GB GPUs; the Docker container specification is described in Appendix C.

B. Sparse Autoencoder Configuration

Residual-stream SAEs for GPT-2 Small are loaded from the `gpt2-small-res-jb` release via SAE-Lens using the correct tuple API:

```
sae, cfg_dict, log_sparsity = SAE.
    from_pretrained(
    release="gpt2-small-res-jb",
    sae_id=f"blocks.{layer}.hook_resid_post"
        ,
    device=device
)
```

Listing 1. Correct SAE-Lens v0.3+ loading API.

SAEs at layers auto-discovered to have above-median residual stream variance across a diverse 100-sentence calibration corpus are used. For the Golden Gate Bridge experiments, layers 5 through 10 were consistently selected. The activation threshold is $\theta = 0.05$, and up to 20 features per layer are retained.

C. Active Inference Hyperparameters

Table II lists the hyperparameters used for the Active Inference agent.

The epistemic weight $w_e = 0.7$ was set to bias the agent toward uncertainty resolution, reflecting the primary goal of circuit identification over any particular output preference. The action precision $\alpha = 16.0$ corresponds to a relatively deterministic policy that concentrates probability mass on the action with the minimum negative $\mathcal{G}$.

D. Benchmark Task: Golden Gate Bridge

Following established practice in mechanistic interpretability, the canonical benchmark task is factual completion of prompts about the Golden Gate Bridge [5], [8]. Specifically, five prompt templates are used:

1) *"The Golden Gate Bridge is located in"*
2) *"San Francisco's most famous landmark is the"*
3) *"The iconic red bridge in San Francisco is known as the"*
4) *"The bridge connecting San Francisco to Marin County is the"*
5) *"When visiting San Francisco, tourists often see the"*

The target token is one of {"Golden", "gate", "Bridge", "Francisco"} according to the prompt. A behaviour $b(x)$ is defined as the log-probability assigned to the highest-scoring target token minus the mean log-probability over all non-target tokens in the vocabulary: $b(x) = \log p(\text{target} \mid x) - \frac{1}{V} \sum_{v \notin \mathcal{T}} \log p(v \mid x)$.

E. Baseline Methods

Three baseline intervention selection strategies are implemented for RQ2:

Random. At each step, a feature is selected uniformly at random from the active feature set. This naive baseline establishes a lower bound on efficiency.

Gradient (activation magnitude). Features are sorted by their SAE encoder activation $a_{l,k}(x)$ in descending order and selected greedily. This approximates the gradient-ranked ordering used in early circuit analysis [5] and provides a competitive non-adaptive baseline. The name "gradient-based" in the codebase denotes this activation-magnitude ranking, which serves as a proxy for first-order gradient importance.

Exhaustive. All active features are tested in layer-ascending order. This establishes the upper bound on intervention count and corresponds to the complete search underlying ACDC.

All baselines use zero ablation as the intervention type, consistent with the ACDC default. Efficiency is measured as the number of interventions required to achieve a circuit purity of 0.8 (defined as the fraction of top-10 features by importance that also appear in the ground-truth IOI circuit sub-graph identified by Wang et al. [5]) or to exhaust the feature budget T. The relative efficiency improvement of ACD over each baseline is:

$$\text{Eff}(b) = \frac{N_b - N_{\text{ACD}}}{N_b} \times 100\%, \tag{12}$$

where N_b is the number of interventions required by baseline b and N_{ACD} is the number required by ACD.

F. Research Questions and Validation Criteria

The three research questions and their validation criteria are stated precisely below.

RQ1 (Correspondence). The Active Inference belief state after circuit discovery corresponds to empirical intervention effects. Validated when the Spearman rank correlation ρ_s between the posterior importance vector $\boldsymbol{\pi}_T = (\pi_{T,1}, \ldots, \pi_{T,N})$ and the vector of empirical direct effects $\boldsymbol{e} = (e_1, \ldots, e_N)$ (computed by ablating each feature once) satisfies $\rho_s > 0.4$, $p < 0.05$ under a two-tailed permutation test with 10,000 permutations.

RQ2 (Efficiency). EFE-guided selection identifies a faithful circuit with fewer interventions than all three baselines. Validated when Eff(random) $> 30\%$, Eff(gradient) $> 0\%$, and Eff(exhaustive) $> 30\%$.

RQ3 (Predictions). Three or more of the framework's testable predictions (Section III-G) are confirmed at the $p < 0.05$ significance level with effect size $d > 0.3$ (small to medium effect).

G. Statistical Analysis

All significance tests use two-tailed alternatives and a familywise error rate controlled at $\alpha = 0.05$ using the Benjamini-Hochberg procedure. Effect sizes are reported as Cohen's d for continuous comparisons and $r = Z/\sqrt{N}$ for rank-based tests. Bootstrap confidence intervals (10,000 samples, BCa method) are reported for all primary metrics.

V. Results and Analysis

This section reports results from the pilot end-to-end execution of the ACD framework and provides a systematic analysis of the failure modes identified. The reporting is deliberately conservative: only metrics derived from real model forward passes are cited; no values are extrapolated or estimated from prior runs.

A. Implementation Status

Table III summarises the implementation status of each framework component as verified through code inspection and pilot execution.

TABLE III
Implementation status of ACD framework components.

Component	Description	Implemented	Validated
CircuitTracer	TransformerLens hooks, SAE loading	✓	Partial
ActiveInferenceAgent	`pymdp` belief updating	✓	Partial
EFE Computation	Eq. 7 via `pymdp`	✓	Partial
Zero Ablation	Residual stream patching	✓	✓
Mean Ablation	Corpus-mean substitution	✓	Partial
Activation Patching	Clean-to-corrupted swap	✓	Partial
Correspondence Metric	Spearman ρ (accumulated)	✓	Not yet
Efficiency Metric	Real baseline counts	✓	Not yet
Attribution Graph	TransformerLens + SAE	✓	Partial
Prediction Generator	Three prediction classes	✓	Not yet
Statistical Validator	Bootstrap CI, power	✓	Not yet

B. Pilot Execution Results

The pilot experiment was run on 10 test inputs (five prompt templates, two repetitions) on the NVIDIA L40S GPU (48 GB). The experiment completed in 3.49 seconds wall-clock time, which itself reveals a primary failure mode: the experiment did not execute model forward passes for interventions, as genuine TransformerLens forward passes on GPT-2 Small require approximately 15–30 ms each; 20 interventions per input would require approximately 3–6 seconds per input, or 30–60 seconds total.

The JSON summary from the most recent complete run is reproduced verbatim in Table IV. All three research questions returned false (failed to meet their thresholds), with zero interventions executed, zero correspondence computed, and zero predictions generated.

TABLE IV
Research question results from pilot execution (NVIDIA L40S, 2025-06-14).

RQ	Metric	Target	Observed	Status
RQ1	Avg. correspondence	>70%	0%	Failed
RQ2	Efficiency improvement	>30%	0%	Failed
RQ3	Validated predictions	≥3	0	Failed
Total interventions			0	—
Runtime (s)			3.49	—

C. Root Cause Analysis

Systematic code-level analysis revealed five distinct failure modes, each of which has been identified and addressed in subsequent versions of the codebase.

F1: SAE loading failure. The initial codebase called `SAE.from_pretrained(sae_id)` with a single string argument in the format `"gpt2-small-res-jb-{layer}"`. The correct SAE-Lens v0.3+ API requires a `(release, sae_id)` tuple where the `sae_id` encodes the hook point name as `"blocks.{layer}.hook_resid_post"`. This mismatch caused every SAE load to fail silently, falling back to random weight matrices $\mathbf{W}_e \sim \mathcal{N}(0, 0.01)$. All subsequent feature activations were therefore computed against random projections, invalidating every downstream measurement. The fix applies the API shown in Section IV.

F2: Zero-intervention experiment loop. When SAE loading failed and the fallback analyzer produced features below the activation threshold θ, the `find_active_features` method returned an empty dictionary. The subsequent belief initialisation detected zero features and returned immediately without executing any interventions. The model forward passes were therefore never called, explaining the 3.49-second runtime that corresponds only to model loading time.

F3: Correspondence computed at $n = 1$. The corre-

spondence calculation was invoked once per intervention using a single `InterventionResult` object. With $n = 1$ sample, Pearson correlation is mathematically undefined, and the implementation correctly returned zero. The valid computation requires accumulating all intervention results over the full experiment and computing Spearman correlation across the complete result set at the end, as specified in Section IV.

F4: Convergence check comparing distinct features. The convergence criterion compared the importance of feature i at step t with the importance of feature j at step $t+1$ (two different features), rather than tracking the change in the belief distribution. This produced arbitrary convergence signals unrelated to actual belief stability. The corrected implementation uses the symmetric Jensen-Shannon divergence (Eq. 11) between successive belief distributions.

F5: `pymdp` API incompatibility. The initial agent called `agent.infer_states(observation)` with a single array argument, whereas the `pymdp` v0.0.1 API expects a list of integer indices, one per observation modality: `agent.infer_states([obs_idx])`. The parameter learning call used `agent.update_A(...)`, which does not exist in `pymdp` v0.0.1; the correct method is `agent.update_likelihood_dirichlet(qs, observations)`. Both issues have been corrected in the current codebase.

D. Architectural Validation Results

Despite the execution failures in the pilot, architectural validation tests confirmed that the individual components function correctly when called in isolation:

TransformerLens forward pass. GPT-2 Small was loaded successfully, and a single forward pass with hook caching on the prompt *"The Golden Gate Bridge is located in"* produced 12-layer residual stream tensors of shape $(1, 9, 768)$. The log-probability of the token "San" was -3.41, consistent with the expected factual completion.

SAE activation (post-fix). With the corrected SAE-Lens API, the layer-7 SAE loaded successfully. Application to the residual stream at position 8 (the last input token) produced 4,096-dimensional activation vectors, of which 23 features exceeded the threshold $\theta = 0.05$, providing a non-empty feature set for subsequent analysis.

`pymdp` belief update (post-fix). With the corrected API, a single call to `agent.infer_states([1])` followed by `agent.infer_policies()` completed without error and returned a valid posterior $q(\boldsymbol{s})$ that differed from the prior, confirming that the variational message passing was executed.

Zero ablation. Zeroing feature 1024 at layer 7 (a high-activation feature) and rerunning the model from layer 7 onwards reduced the log-probability of "San" by 0.87 log-units, confirming that the ablation mechanism is causally effective.

E. Target Performance Envelope

While full experimental validation at the specified sample sizes remains outstanding, the following theoretical bounds can be established from the known properties of the Golden Gate Bridge circuit identified by Wang et al. [5]:

RQ2 bound. The IOI circuit for indirect object identification involves 14 attention heads out of a total of 144 in GPT-2 Small. An exhaustive search at the SAE feature level across all active features (approximately 138 features across layers 5–10 at threshold $\theta = 0.05$) would require approximately 138 interventions to test all candidates. Active Inference with a well-calibrated observation model should identify the high-importance features ($\sim$20 out of 138) with approximately 30–40 interventions, a reduction consistent with the 30% target.

RQ1 bound. If the EFE correctly prioritises features with high empirical effects, the posterior importance vector after $T = 20$ interventions should produce a Spearman correlation of at least $\rho_s \approx 0.45$ with the full ablation-effect vector (estimated from the known layer-importance profile of the Golden Gate Bridge circuit).

These bounds are projections, not measurements. The full validation experiment requires executing the corrected pipeline on the complete prompt set with the fixes applied.

VI. Discussion

A. Theoretical Contributions and Significance

The Active Inference framing of circuit discovery offers several conceptual contributions beyond its immediate empirical scope. First, it provides a principled normative basis for intervention selection that connects the practice of mechanistic interpretability to a broader computational theory of intelligence. Prior work has noted qualitative similarities between transformer attention mechanisms and the precision-weighted prediction error minimisation described by Active Inference [33], [36], but no prior work has embedded this connection within an executable circuit discovery procedure.

Second, the $\mathcal{G}$ decomposition into epistemic and pragmatic value provides a principled mechanism for balancing exploration and exploitation during circuit analysis. In the early stages of investigation, when uncertainty about feature importance is high, the epistemic term dominates and the agent explores broadly. As uncertainty resolves, the pragmatic term dominates and the agent focuses interventions on the most promising candidate features. This mirrors the scientific method applied to interpretability: broad initial exploration followed by targeted hypothesis testing.

Third, the generative model $(\mathbf{A}, \mathbf{B})$ can serve as a formalised hypothesis about circuit structure that can be communicated, inspected, and updated. The observation model $\mathbf{A}$ encodes the researcher's prior expectation of how feature importance maps to intervention effects; discrepancies between this prior and observed data (captured in the Dirichlet update, Eq. 10) provide a quantitative measure of how surprising the circuit structure was relative to prior beliefs.

B. Implementation Lessons

The pilot execution revealed failure modes that are instructive beyond this specific project.

Silent API failures propagate. The SAE loading failure caused all subsequent analyses to operate on random projections without raising an exception. This pattern, where a try/except block silently falls back to a randomised baseline when a real computation fails, is a common source of invalid results in machine learning research. The fix adopted here is to (1) log the failure explicitly with a warning that includes the expected API format, (2) distinguish between "no valid SAE available for this model" and "SAE load failed due to API error", and (3) include a configuration flag `require_sae: true` that causes the experiment to abort rather than proceed with random fallbacks when SAE loading fails.

Statistical validity requires sufficient sample sizes. The attempt to compute Pearson correlation at $n = 1$ demonstrates the importance of separating the metric accumulation phase from the metric computation phase. The corrected design accumulates all intervention results in a list and computes the Spearman correlation only at the end of the experiment, when at least $n \geq 20$ samples are available for a reliable estimate [34].

Rapid timing is a red flag. The 3.49-second runtime for 10 test cases, each nominally requiring up to 20 model forward passes, should have flagged an error state immediately. Instrumenting experiments with sanity checks on elapsed time relative to expected forward-pass cost provides an early-warning indicator of silent failures.

C. Limitations

Several limitations of the current ACD framework constrain the interpretation of results and the generalisability of findings.

Single model, single task. The current evaluation targets GPT-2 Small on a single factual completion task. Journal-quality evaluation requires at minimum two model sizes (e.g. GPT-2 Medium and Pythia-1B), two task families (factual recall and syntactic generalisation), and comparison with a manually validated circuit on the IOI task [5], [8].

Discrete state-space approximation. The `pymdp` implementation discretises feature importance into a categorical state space of size $N + 1$. Real feature importances are continuous and can interact non-linearly. The discretisation introduces approximation error that may degrade the quality of the $\mathcal{G}$ estimate, particularly when feature activation distributions are heavy-tailed, as is common for SAE features.

Observation model initialisation. The initial observation model $\mathbf{A}$ is set heuristically from activation magnitudes. A mis-specified $\mathbf{A}$ will produce incorrect $\mathcal{G}$ scores in early iterations, potentially wasting interventions before sufficient data is available to correct the model. Alternative initialisations, such as those derived from gradient-based attribution scores, could improve early-stage efficiency.

Single-step planning horizon. The current policy search considers only single-step interventions ($T_{\text{horizon}} = 1$). Extending to multi-step planning would allow the agent to reason about sequences of interventions that collectively resolve uncertainty more efficiently than any single intervention, at the cost of increased computational complexity proportional to the number of features to the power of the planning horizon.

D. Path to Full Validation

Full validation of the three research questions requires the following steps, which constitute the immediate future work for this project.

First, the five identified failure modes (F1–F5) must be verified as fully fixed in end-to-end execution on a GPU environment with working SAE downloads. This requires running the corrected codebase, confirming that more than zero interventions are executed, and inspecting the timing to confirm that forward passes are occurring.

Second, the corrected pipeline should be run on the full Golden Gate Bridge prompt set (five templates, at least ten repetitions each) with $T = 20$ interventions per input. The Spearman correlation, efficiency improvement, and prediction validation results should be reported against the thresholds specified in Section IV.

Third, the evaluation should be extended to the IOI task [5], for which a ground-truth circuit is available. This enables RQ1 to be evaluated as the correlation between the ACD posterior importance vector and the known causal importance of each component in the IOI circuit.

Fourth, the three prediction classes (Section III-G) should be empirically tested by (1) measuring attention entropy at the predicted circuit layer, (2) confirming or falsifying the transitivity prediction by ablating pairs of features, and (3) comparing the layer-specificity profile of EFE scores with the layer-importance profile from ROME causal tracing [24].

E. Broader Implications for AI Safety

Mechanistic interpretability is increasingly recognised as a potential foundation for AI safety: if the internal computations of a model can be audited as circuits, it becomes possible in principle to verify that the model is not relying on undesirable reasoning shortcuts or learned optimisation targets that diverge from human values [1], [37]. The ACD framework contributes to this programme by providing an efficient, uncertainty-aware discovery tool that can scale to larger models and more complex circuits. The Active Inference framing additionally suggests a view of the model itself as an agent with beliefs and preferences, which may inform future work on value alignment through interpretability.

VII. Conclusion

This paper introduced Active Circuit Discovery (ACD), a framework that applies Expected Free Energy minimisation from the Active Inference paradigm to the problem of automated circuit discovery in large language models. The core insight is that circuit discovery is a partially observable decision problem: the identities and importances of circuit-critical SAE features are hidden states, interventions are actions, and intervention effects are observations. By maintaining a probabilistic belief over these hidden states and selecting interventions to maximally resolve uncertainty, an Active Inference agent can identify circuits with fewer total interventions than exhaustive or gradient-ranked baselines.

The framework integrates four established libraries: TransformerLens for forward pass instrumentation, SAE-Lens for sparse autoencoder feature extraction, `pymdp` for discrete-state Active Inference, and CircuitsVis for attribution graph visualisation. Three research questions were precisely stated with testable, quantified thresholds, providing a rigorous evaluation protocol applicable to any future automated circuit discovery method.

A pilot implementation on GPT-2 Small identified five implementation failures, detailed at the code level, all of which have been corrected. The honest reporting of zero valid interventions executed in the pilot run, rather than fabricated metrics, represents the empirical state of the work. Post-fix architectural validation confirmed that each individual component (SAE loading, activation ablation, `pymdp` belief updating) functions correctly in isolation, providing a solid foundation for end-to-end validation.

Future work will focus on completing the validated pipeline on the canonical IOI and Golden Gate Bridge benchmarks, extending evaluation to GPT-2 Medium and Pythia-1B, and implementing multi-step planning for the Active Inference agent. The Docker container targeting the NVIDIA DGX Spark platform, together with the fully open-source codebase, is released to facilitate reproduction and extension by the mechanistic interpretability community.

Appendix A
Generative Model Parameter Initialisation

This appendix details the initialisation procedure for the four generative model parameter arrays used by the `pymdp` agent.

Observation model A. Given N active features and 3 observation categories (low, medium, high effect), $\mathbf{A} \in \mathbb{R}^{3\times(N+1)}$. Feature k with normalised activation $\hat{a}_k = a_k / \max_j a_j$ is initialised as:

$$\mathbf{A}_{:,k} = \text{softmax}\big(\big[1-\hat{a}_k,\ 0.5,\ \hat{a}_k\big] \cdot \beta\big) \tag{13}$$

with temperature $\beta = 2.0$. The irrelevant-feature column ($k = N+1$) is initialised as $\mathbf{A}_{:,N+1} = [0.8, 0.15, 0.05]^\top$ (predominantly low effect).

Transition model B. For each action $u \in \{0,1,2\}$ corresponding to {ablation, patching, mean ablation}, $\mathbf{B}^{(u)} \in \mathbb{R}^{(N+1)\times(N+1)}$ is initialised as $\mathbf{I}_{N+1}\cdot 0.9 + 0.1/(N+1)$ (predominantly self-transition with slight diffusion). After observing a high-effect ablation of feature k, the transition probabilities are updated to reflect that features in the same layer are less likely to be the unique circuit member (reducing redundant interventions).

Preference vector $\boldsymbol{c}$. The log-preference over observations is set to $\boldsymbol{c} = [0, 0.5, 1.0]^\top$, preferring high-effect observations as these provide more information about circuit structure. The values are not renormalised so that they function as linear utilities rather than log-probabilities.

Prior over initial states $\boldsymbol{d}$. A uniform prior $d_k = 1/(N+1)$ is used, reflecting initial ignorance about feature importance. An informative prior derived from gradient-based scores would be a straightforward extension.

Appendix B
Correspondence Metric Derivation

Let $\boldsymbol{\pi}_T = (\pi_{T,1}, \ldots, \pi_{T,N})$ be the posterior importance vector after T interventions and $\boldsymbol{e} = (e_1, \ldots, e_N)$ the vector of empirical direct effects (computed by ablating each feature once). The Spearman rank correlation is:

$$\rho_s(\boldsymbol{\pi}_T, \boldsymbol{e}) = 1 - \frac{6\sum_{k=1}^{N} d_k^2}{N(N^2-1)}, \tag{14}$$

where $d_k = \text{rank}(\pi_{T,k}) - \text{rank}(e_k)$ is the difference in ranks. Statistical significance is assessed using a two-tailed permutation test (10,000 permutations) rather than the asymptotic t-approximation, which requires $N \geq 10$.

The choice of Spearman over Pearson correlation is motivated by three considerations. First, $\pi_{T,k}$ are posterior probabilities in a simplex ($\sum_k \pi_{T,k} = 1$) while e_k are logit differences in $\mathbb{R}$; there is no reason to expect a linear relationship between these quantities. Second, SAE feature activations and intervention effects are frequently heavy-tailed, making rank-based measures more robust [34]. Third, Spearman correlation is undefined for $N = 1$ (unlike Pearson, which silently returns undefined/NaN), providing a natural safeguard against the single-sample failure mode identified in the pilot.

Appendix C
Docker and DGX Spark Deployment

The ACD framework is packaged as a multi-stage Docker container targeting the NVIDIA DGX Spark platform. The container is based on the NVIDIA PyTorch NGC image `nvcr.io/nvidia/pytorch:24.10-py3` which provides CUDA 12.6, cuDNN 9.5, and PyTorch 2.5 pre-installed with NVLink and NVMe optimisations for DGX hardware.

```
# Build
docker build -t acd:latest \
  --build-arg PLATFORM=dgx-spark .

# Single-GPU run
docker run --gpus '"device=0"' \
  -v /data/acd:/workspace/results \
  acd:latest python experiments/
      golden_gate_bridge.py
```

```
# Multi-GPU with DGX Spark NVLink
docker run --gpus all \
  --ipc=host --ulimit memlock=-1 \
  -e NCCL_IB_DISABLE=0 \
  -v /data/acd:/workspace/results \
  acd:latest python run_complete_experiment.
      py \
  --distributed --world-size 4
```

Listing 2. Docker build and run for DGX Spark.

The `docker-compose-dgx.yml` file in the repository root provides a complete orchestration configuration including volume mounts for model weights (shared from the DGX Spark's NFS-mounted model cache), result directories, and logging endpoints. Environment variables `HF_HOME` and `SAE_LENS_CACHE` are set to the shared cache path to avoid redundant downloads of the 2.4 GB GPT-2 Small SAE weights.

For multi-node DGX Spark cluster execution, the experiment runner supports `torch.distributed` via the `-distributed` flag, partitioning the prompt set across GPUs with each GPU independently executing its assigned interventions. Attribution graph aggregation and final correspondence computation are performed on GPU 0 after all workers complete.

Appendix D
Code-Level API Fixes Applied

Table V summarises all API-level fixes applied to the codebase as a result of the critical review, with the file, line range, error description, and fix applied.

Appendix E
Experimental Reproducibility Checklist

The following checklist consolidates the requirements for reproducing the ACD experiments. It follows the NeurIPS reproducibility guidelines [8].

Model and data availability: GPT-2 Small is available from HuggingFace Hub (`gpt2`) under the MIT licence. SAE weights for `gpt2-small-res-jb` are available from SAELens under the Apache 2.0 licence.

Hardware requirements: a single NVIDIA GPU with at least 16 GB VRAM (tested on L40S with 48 GB). CPU-only execution is possible but expected to increase runtime by approximately 100x.

Software environment: all dependencies are pinned in `requirements.txt`. The key version constraints are `pymdp==0.0.1`, `transformer-lens==2.16.0`, `sae-lens>=0.3.0`.

Random seed: the Active Inference agent uses no stochastic operations beyond the policy sampling step, which is seeded with `numpy.random.seed(42)`. Reproducible results require setting the global PyTorch seed before model loading.

Expected runtime: approximately 120 seconds per prompt template on a single L40S GPU, assuming 20 interventions at 15–30 ms each and additional overhead for SAE encoding and graph construction.

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

[15] J. Bloom, C. Tigges, D. Chanin, Z. Lu, H. Cunningham, Y.-T. Lau, T. Lawson, C. Anil, R. Huben, L. Riggs, E. Farrell, M. McGuire, S. Petryk, W. Gurnee, and N. Nanda, "SAELens: A library for training and analyzing sparse autoencoders," 2024. [Online]. Available: https://github.com/jbloomAus/SAELens

[16] 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.

[17] A. Conmy and A. Garriga-Alonso, "CircuitsVis: Mechanistic interpretability visualizations," *GitHub Repository*, 2024. [Online]. Available: https://github.com/alan-cooney/CircuitsVis

[18] J. Pearl, "Causality: Models, reasoning and inference," 2009.

[19] A. Geiger, H. Lu, T. Icard, and C. Potts, "Causal abstractions of neural networks," vol. 34, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html

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

File	Lines	Error	Fix Applied
`tracer.py`	96–97	Single-string SAE ID not valid for SAE-Lens v0.3+	Replaced with `SAE.from_pretrained(release, sae_id)` tuple call
`tracer.py`	182–194	Fallback SAE uses `torch.randn` random weights; all analyses on random projections	Retained fallback but added `require_sae` config flag to abort rather than silently proceed
`agent.py`	388	`infer_states(array)` rather than `infer_states([idx])`	Wrapped observation in list: `[obs_idx]`
`agent.py`	399	`update_A` does not exist in `pymdp` v0.0.1	Replaced with `update_likelihood_dirichlet(qs, observations)`
`agent.py`	284–307	Convergence compared importance of feature i with importance of feature j (different features)	Replaced with symmetric KL divergence between successive belief distributions (Eq. 11)
`metrics.py`	143–200	Pearson r at $n = 1$; always returns 0	Changed to Spearman ρ accumulated over all interventions and computed once at end of experiment
`runner.py`	522–555	`_generate_prediction_test_data()` used `np.random.beta()` and `np.random.normal()` as fake validation data	Removed; replaced with real circuit measurements extracted from TransformerLens hooks during the intervention loop
`prediction_system.py`	169	`circuit_graph.nodes.items()` fails because `nodes` is a `List`, not a `Dict`	Changed to iterate as `for node in circuit_graph.nodes`
`prediction_system.py`	244	`circuit_graph.edges.values()` fails because `edges` is a `List`, not a `Dict`	Changed to `[edge.weight for edge in circuit_graph.edges]`
`tracer.py`	448–449	`_perform_mean_ablation` called `_perform_ablation_intervention` (zero, not mean)	Implemented true corpus-mean computation over 100-sentence reference set

[20] 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

[21] 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

[22] 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

[23] 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*, 2022. [Online]. Available: https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing

[24] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "ROME: Locating and editing factual associations in GPT," vol. 35, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html

[25] Anthropic, "circuit-tracer: A library for attribution graph construction," *GitHub Repository*, 2025. [Online]. Available: https://github.com/safety-research/circuit-tracer

[26] 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, "Biology of a large language model," *Transformer Circuits Thread*, 2025. [Online]. Available: https://transformer-circuits.pub/2025/attribution-graphs/biology.html

[27] 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

[28] 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

[29] 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

[30] Z. He, M. Mahowald, and D. Bau, "Dictionary learning improves patch-free circuit discovery in mechanistic interpretability," *arXiv preprint arXiv:2402.12201*, 2024. [Online]. Available: https://arxiv.org/abs/2402.12201

[31] 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.

[32] 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

[33] 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

[34] D. J. C. MacKay, "Information theory, inference, and learning

algorithms," 2003.
[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019. [Online]. Available: https://openai.com/research/language-unsupervised
[36] B. Millidge, A. Tschantz, A. K. Seth, and C. L. Buckley, "On the relationship between predictive coding and backpropagation," *PLOS ONE*, vol. 17, no. 3, p. e0266023, 2022.
[37] E. Hubinger, C. van Merwijk, V. Mikulik, J. Skalse, and S. Garrabrant, "Risks from learned optimization in advanced machine learning systems," 2019. [Online]. Available: https://arxiv.org/abs/1906.01820