DISCOVERING SPARSE CONCEPT GRAPHS AS INTERPRETABLE SURROGATES FOR VISION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent methods for mechanistic interpretability of vision models uncover interpretable concepts and trace the circuits underlying predictions for individual samples or classes. However, predictions for different classes often depend on shared concepts, and this shared structure is central to understanding how the network operates as a whole. We address this gap by learning sparse rules that describe how combinations of concepts in one layer give rise to concepts in the next, and compose these rules across layers into a CONCEPTOME, a single directed concept graph that serves as a global, symbolic surrogate of the network. The CONCEPTOME can be examined as a whole or queried for individual samples, single classes, or groups of classes to reveal the computational structure they share. We evaluate our approach on CLIP ViT and CLIP ResNet and show that the recovered CONCEPTOMEs remain sparse while faithfully preserving model computation. Inspection reveals how concepts are combined, reused, transformed, and specialized throughout the network. Interventions show that the recovered structure isolates a compact set of concepts on which the computations and predictions of the model depend. To make this structure accessible, we develop an interactive interface for exploring the CONCEPTOME, visualizing concepts and rules, and intervening on predictions.

1 INTRODUCTION

Neural networks make predictions by progressively transforming an input through a sequence of layers, and the resulting internal representations trace the computation from input to decision. Early interpretability methods attribute predictions to input regions (Simonyan et al., 2013; Selvaraju et al., 2017), while concept-based approaches identify human-interpretable concepts within internal representations (Kim et al., 2018; Ghorbani et al., 2019; Fel et al., 2023; Bricken et al., 2023; Gao et al., 2025). Neither, however, captures the computational path through which the decision is formed. Circuit methods address this by identifying a sparse set of network components that produce a behavior of interest, such as the prediction of a specific class (Olah et al., 2020; Conmy et al., 2023; Marks et al., 2025; Kim et al., 2025). Understanding the general computation of a network instead requires tracing how concepts activate and interact throughout the forward pass and how this structure is organized and shared across predictions.

A global view is essential because concepts rarely act in isolation. As information propagates through the network, concepts are combined, reused across different predictions, transformed into new representations, or specialized toward more specific roles. Analyzing one class or sample at a time therefore cannot reveal how this shared computation is organized. Such a description can act as an interpretable surrogate of the model that uncovers recurring and previously unknown concept interactions and can be queried globally or for particular classes and samples.

We formulate this missing organizational view as *population-level cross-layer concept dependency discovery*. Rather than explaining a single sample or class, we seek a sparse, reusable population level description of how combinations of concepts in one layer systematically relate to concepts in the next across a data distribution. We represent these dependencies as rules with multi-concept heads and tails, capturing how groups of concepts jointly relate across consecutive layers.

We propose CONCORD (CONcept COmpositional Rule Discovery) to recover this structure at scale. We first train TopK SAEs (Gao et al., 2025) to obtain sparse concept representations that provide an interpretable interface to the backbone. CONCORD then learns a sparse rule bank between each

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

pair of consecutive concept representations, capturing recurring dependencies between combinations of concepts across the data. From the resulting rule banks, we construct a network-wide graph over concepts, which we call a CONCEPT CONNECTOME, a layered directed graph containing concept and rule nodes. Each rule node connects its head concepts at one layer to its tail concepts at the next, preserving multi-concept dependencies without decomposing them into pairwise connections. We accompany the recovered rules and CONCEPTOMEs with an interactive website that lets users inspect concepts, trace their cross-layer dependencies, and explore the structure throughout the model.[1]

We evaluate the recovered structure on transformer- and convolution-based vision architectures, specifically CLIP ViT-B/32 and CLIP RN50. At the rule level, we find compact multi-concept dependencies that recur across samples and are often shared across classes. At the network level, the composed CONCEPTOME preserves a substantial portion of the original model computation, while interventions show that its pathways identify concepts with substantial influence on the model's output. Qualitative inspection further illustrates how shared concepts are repeatedly combined across layers into increasingly class-relevant structure.

2 RELATED WORK

From activations to concepts A major line of work seeks to uncover interpretable concepts within the internal representations of vision models. Early approaches relied on activation visualization and network dissection to associate individual neurons with human-interpretable concepts (Zeiler & Fergus, 2014; Bau et al., 2017). Later work showed that concepts can be distributed across multiple neurons (Fong & Vedaldi, 2018) and developed methods to identify or construct them, including concept activation vectors, concept bases and automatic concept discovery (Kim et al., 2018; Zhou et al., 2018; Ghorbani et al., 2019; Fel et al., 2023; Bykov et al., 2023). More recently, sparse autoencoders (SAEs) have emerged as a scalable approach for decomposing neural representations into sparse concepts (Bricken et al., 2023; Cunningham et al., 2023; Gao et al., 2025), and have also been applied to vision and vision-language models (Gorton, 2024; Pach et al., 2025). These methods yield rich concept spaces but leave open how concepts combine across layers.

From concepts to computation Early work in mechanistic interpretability introduced circuits as groups of interpretable features and their connections that jointly implement meaningful computations in neural networks (Olah et al., 2020). In convolutional models, such circuits revealed how low-level visual concepts combine into more complex ones (Cammarata et al., 2020). Subsequent work characterized processing stages for relational reasoning in Vision Transformers (Lepori et al., 2024), traced influential neuron paths across layers (Wang et al., 2025), and recovered circuits for particular visual concepts (Kwon et al., 2025). More recent approaches construct sparse or edge-level circuits in modern vision models (Kim et al., 2025; Li et al., 2026; Żukowska et al., 2026) or identify subnetworks that implement particular functions (Csordás et al., 2021).

From computation to structure A less explored direction studies the organization of internal computation. Fischer et al. (2021) mine noise-robust logical rules between groups of neurons in different layers and chain them to reveal compositional structure, and Hu & Tian (2022) abstract neural computation through directed causal dependencies between neurons. Kowal et al. (2024) discover concepts and their inter-layer connections in vision models, while Chen et al. (2025) study computational structure through compositions of AND, OR, and ADDER gates. We build on these structural perspectives but operate in a sparse, interpretable concept space rather than on neurons, and recover multi-concept rules across consecutive layers instead of pairwise links. Our rules capture OR, AND, and noisy-AND relations, expressing how multiple concepts jointly contribute to a cross-layer dependency. We design this rule-based representation to scale to modern vision models and learn a single global population-level structure that can be easily queried for individual samples or classes.

[1]To comply with the double-blind review requirements, we cannot host the website publicly during the review period. Instead we provide screenshots in Appendix C.

3 PRELIMINARIES

We first introduce the notation and preliminaries that our method builds upon, then formalize the problem. Let F denote the trained neural network we seek to interpret. We select a sequence of intermediate representation states

$$a^{(0)}(x), a^{(1)}(x), \ldots, a^{(L)}(x),$$

and denote the complete computation between consecutive selected states by f_ℓ, such that

$$a^{(\ell+1)}(x) = f_\ell\Big(a^{(\ell)}(x)\Big), \qquad \ell = 0, \ldots, L-1.$$

The computation before $a^{(0)}$ and after $a^{(L)}$ is denoted by f_{in} and f_{out}, respectively, so that

$$a^{(0)}(x) = f_{\text{in}}(x), \qquad F(x) = f_{\text{out}} \circ f_{L-1} \circ \cdots \circ f_0 \circ f_{\text{in}}(x).$$

Each f_ℓ therefore denotes the complete computation between two *selected* states, and the decomposition reflects the chosen granularity. For brevity, we refer to each selected intermediate state as a *layer* throughout the paper. We select consistent post-block states as layers and study how information changes between consecutive layers.

Sparse concept representations Intermediate neural representations can be polysemantic, meaning multiple concepts can be superposed across the same activation dimensions, making individual neurons or channels difficult to interpret directly (Elhage et al., 2022). We therefore use sparse dictionary learning to decompose each layer representation into a larger set of sparse concepts. At each layer ℓ, we denote by g_ℓ the mapping from the model activation $a^{(\ell)}(x)$ to a sparse, non-negative concept representation $z^{(\ell)}(x)$, and by h_ℓ the corresponding reconstruction map:

$$z^{(\ell)}(x) = g_\ell\Big(a^{(\ell)}(x)\Big) \in \mathbb{R}^{d_\ell}_{\geq 0}, \qquad \hat{a}^{(\ell)}(x) = h_\ell\Big(z^{(\ell)}(x)\Big).$$

Our approach is agnostic to the particular dictionary-learning method. In this work, we instantiate g_ℓ and h_ℓ using TopK Sparse Autoencoders (SAEs) (Makhzani & Frey, 2014; Gao et al., 2025), which learn sparse concept representations by reconstructing the original activations. Given an activation vector $a^{(\ell)}(x)$, they compute

$$z^{(\ell)}(x) = \text{TopK}(\text{ReLU}(W_{\text{enc}} a^{(\ell)}(x) + b_{\text{enc}}); k), \qquad \hat{a}^{(\ell)}(x) = W_{\text{dec}} z^{(\ell)}(x) + b_{\text{dec}},$$

and are trained to minimize the reconstruction error between $a^{(\ell)}(x)$ and $\hat{a}^{(\ell)}(x)$. The TopK operator enforces sparsity directly by retaining only the k largest latent activations. We treat each latent dimension $z^{(\ell)}_j(x)$ as a concept and its value as the corresponding concept activation. For convolutional vision architectures, a single dictionary is learned and shared across all spatial positions p within a layer; analogously, for vision transformers, a single dictionary is shared across all token positions p. This gives each layer a common concept vocabulary while retaining position-wise activations.

4 METHOD

Given sparse concept representations at each selected layer, we seek to recover recurring dependencies between concepts across layers. We model these dependencies using sparse rules defined over concept presence rather than exact activation magnitudes. We therefore first derive an image-level concept-presence representation, then learn rule banks between consecutive layers and compose them across the network into a CONCEPTOME. In Figure 1 we illustrate this construction.

Image-level concept presence Let $z^{(\ell)}_{j,p}(x) \geq 0$ denote the activation of concept j in an image x at position p, which indexes spatial locations in convolutional maps or tokens in ViTs. We consider a concept present if it is active at any position,

$$c^{(\ell)}_j(x) = \mathbb{I}\left[\max_p z^{(\ell)}_{j,p}(x) > 0\right],$$

yielding a concept-presence vector $c^{(\ell)}(x) \in \{0,1\}^{d_\ell}$. The continuous activations are retained for reconstruction and intervention, while the binary concept-presence vectors form the input to our cross-layer rule discovery.

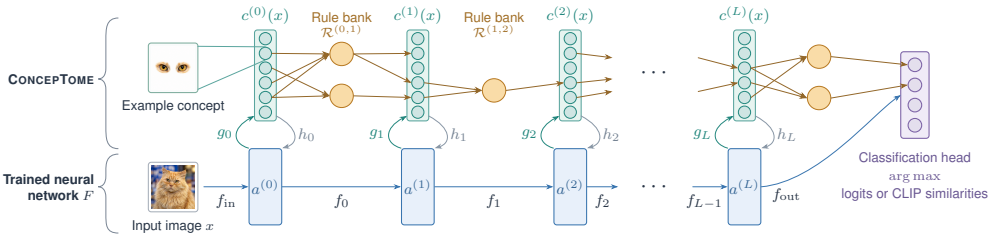


Figure 1: **Layered view of the ConceptTome.** The trained neural network F produces selected intermediate representations $a^{(\ell)}(x)$ connected by computations f_ℓ (bottom). At each layer, a TopK SAE encoder g_ℓ maps $a^{(\ell)}(x)$ to a sparse concept representation $z^{(\ell)}(x)$, which is binarized to $c^{(\ell)}(x)$. Concord learns rule banks between consecutive binary concept representations and composes them across layers into the network-wide ConceptTome (top), yielding an interpretable symbolic surrogate of the network's internal computation.

4.1 Concord: Cross-Layer Rule Discovery

For each pair of consecutive layers $(\ell, \ell+1)$, we seek a concise rule set describing how concept combinations at layer ℓ relate to concepts at layer $\ell+1$ across the data. Each rule is written as

$$H_r \xrightarrow{\tau_r} T_r, \qquad H_r \subseteq [d_\ell], \quad T_r \subseteq [d_{\ell+1}],$$

where H_r and T_r contain concepts from layers ℓ and $\ell+1$, respectively, and τ_r is the minimum number of head concepts required for activation. $\tau_r = 1$ gives OR semantics, $\tau_r = |H_r|$ gives AND semantics, and intermediate values give noisy-AND relations that tolerate missing head concepts.

Rule network Concord uses a two-layer binary neural network with R hidden rule units. It maintains continuous edge-probability matrices $W_H \in [0,1]^{R \times d_\ell}$ and $W_T \in [0,1]^{d_{\ell+1} \times R}$, with corresponding binary connection matrices W_H^b and W_T^b, together with a learned bias $\mathbf{b} \in \mathbb{R}^R$. We denote the continuous parameters by $\theta = \{W_H, W_T, \mathbf{b}\}$. The entries of W_H and W_T parameterize the probabilities of head-to-rule and rule-to-tail connections, respectively. Figure 2 shows this architecture.

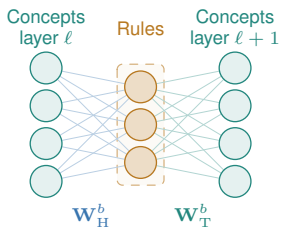


Figure 2: Concord architecture. W_H^b maps source concepts to rule units and W_T^b maps active rules to target concepts.

Forward pass During each forward pass, binary connections are sampled independently element-wise, $W_H^b \sim \text{Bernoulli}(W_H)$ and $W_T^b \sim \text{Bernoulli}(W_T)$.[2] The bias is discretized to $\mathbf{b}^d$, yielding a binary rule graph

$$G = \left(W_H^b, W_T^b, \mathbf{b}^d\right) \sim p_\theta.$$

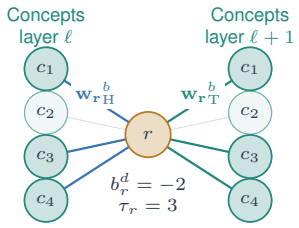


Figure 3: AND-rule with head $\{c_1^{(\ell)}, c_3^{(\ell)}, c_4^{(\ell)}\}$ and tail $\{c_1^{(\ell+1)}, c_3^{(\ell+1)}, c_4^{(\ell+1)}\}$. The bias $b_r^d = -2$ sets the threshold $\tau_r = 3$.

Given this sampled graph, the first layer determines which rules are active, with $\boldsymbol{\rho}(x) \in \{0,1\}^R$,

$$\boldsymbol{\rho}(x) = \mathbb{I}\left[W_H^b c^{(\ell)}(x) + \mathbf{b}^d > 0\right].$$

The second layer maps the active rules to concepts at layer $\ell+1$,

$$\hat{c}^{(\ell+1)}(x) = \mathbb{I}\left[W_T^b \boldsymbol{\rho}(x) > 0\right].$$

The output follows OR semantics, so a concept is active whenever at least one active rule connects to it.

For each rule unit r, the sampled connections define its head and tail, while the discretized bias determines its threshold: $H_r = \{j \mid W_H^b[r,j] = 1\}$, $T_r = \{k \mid W_T^b[k,r] = 1\}$, $\tau_r = 1 - \mathbf{b}_r^d$. The discretized bias is constrained so that $\tau_r \in \{1, \ldots, |H_r|\}$. Figure 3 illustrates this interpretation for an AND-rule.

[2] W_H and W_T are clipped to $[0,1]$ after each update.

Training objective We train the continuous parameters θ through the binary graphs they induce. We minimize the expected target concept-presence loss together with regularization,

$$\mathcal{L}_{\text{CONCORD}}(x) = \mathbb{E}_{G\sim p_\theta}\left[\mathcal{L}_{\text{target}}(x, G)\right] + \mathcal{R}(x, \theta). \tag{1}$$

We approximate the expectation using M Monte Carlo graph samples per update. For a sampled graph G, we train the rule network to reproduce the observed concept-presence vector at layer $\ell + 1$. Because active target concepts are sparse, we balance errors according to their frequency in the training data. Let α denote the fraction of active target concepts in the training set, and weight false positives by α and false negatives by $1 - \alpha$, giving

$$\mathcal{L}_{\text{target}}(x, G) = \sum_{j=1}^{d_{\ell+1}} \left[\alpha \left(1 - c_j^{(\ell+1)}(x)\right) \hat{c}_j^{(\ell+1,G)}(x) + (1-\alpha) c_j^{(\ell+1)}(x) \left(1 - \hat{c}_j^{(\ell+1,G)}(x)\right)\right]. \tag{2}$$

The first term penalizes false positives and the second false negatives. Because α is small under sparse concept presence, the weighting prevents inactive concepts from dominating the objective.

The remaining objectives are grouped into

$$\mathcal{R}(x, \theta) = \lambda_{\text{var}} \operatorname{Var}_{G\sim p_\theta}\left[\mathcal{L}_{\text{target}}(x, G)\right] + \lambda_{\text{bin}} \mathcal{R}_{\text{bin}}(W_H, W_T) + \lambda_{\text{rule}} \mathcal{R}_{\text{rule}}(W_H, W_T). \tag{3}$$

The variance term encourages sampled binary graphs to achieve consistent target concept-presence loss and is estimated from the same Monte Carlo samples. $\mathcal{R}_{\text{bin}}$ is a binarization regularizer that pushes the continuous edge probabilities in W_H and W_T toward 0 or 1, while $\mathcal{R}_{\text{rule}}$ penalizes rule complexity. The exact regularization terms and Monte Carlo estimator are provided in Appendices A.1 and A.2.

Backward pass Because the sampled connections, discretized bias, and binary activations are non-differentiable, we optimize the underlying continuous parameters using straight-through estimators. We use custom gradient routing for the hidden rule layer and a standard straight-through estimator for the output layer. The exact backward rules are provided in Appendix A.3. We select the final rule bank from the learned distribution using held-out selection data, as detailed in Appendix A.4.

Endpoint rules The final transition differs from the intermediate rule banks by mapping concepts to output classes. We therefore use the binary rule architecture introduced for class-specific pattern discovery in prior work (Walter et al., 2024), whose output layer enforces a single dominant class and is trained with a classification objective. The remaining training procedure follows CONCORD.

4.2 SOURCE-MASKED TRAINING

When rule banks are composed, downstream layers receive concept representations filtered by the preceding learned structure rather than the full original representation. We therefore train each rule bank directly on these source-masked concept activations. For a layer pair $(\ell, \ell+1)$, the filtered source concepts are decoded and propagated through the original model to obtain the corresponding target representation, yielding $\left(\widetilde{c}^{(\ell)}(x), \widetilde{c}^{(\ell+1)}(x)\right)$. This aligns each rule bank with the representations it encounters when composed into the CONCEPTOME. Further details are provided in Appendix A.5.

4.3 CONSTRUCTING THE CONCEPTOME

After learning a deterministic rule bank for each consecutive layer pair, we compose them into a CONCEPTOME. Concepts form the layer-wise nodes of the graph, while rules remain intermediate nodes connecting their heads to their tails. Hence, a multi-concept dependency is preserved as a single compositional relation rather than flattened into independent pairwise edges.

The resulting layered directed graph connects the local concept rules across the selected layers of the network into a single network-wide graph.

5 EXPERIMENTS

In this section, we evaluate the recovered structure at two levels: individual CONCORD rule banks and the composed CONCEPTOME.

Backbones and representations We use OpenCLIP CLIP RN50 and CLIP ViT-B/32 (QuickGELU, OpenAI weights) (He et al., 2016; Dosovitskiy et al., 2021; Radford et al., 2021; Ilharco et al., 2021). For ViT-B/32, we extract activations at the output of each of its 12 transformer blocks, after both the attention and MLP residual updates. For RN50, we extract activations after each of its 16 residual bottleneck blocks, following the residual addition and final ReLU. Further details on the corresponding SAE configurations are provided in Appendix B.1.

Data split and model selection The SAEs are trained on the original ImageNet-1K (Russakovsky et al., 2015) training split. For CONCORD, we consider IMAGENETTE, IMAGEMEOW, and IMAGEWOOF, which are 10-class, 13-class, and 10-class subsets of ImageNet-1K, respectively. For each subset, we split its training split into 90% *train* and 10% *selection*. The train portion is used to fit CONCORD, while the selection portion is used to choose training checkpoints and the final rule banks. The corresponding validation splits are reserved exclusively for final evaluation.

5.1 EVALUATING THE LOCAL CONCORD

We first characterize the rule banks discovered by CONCORD along four dimensions. We measure structural complexity by the number of recovered rules and their head and tail sizes, and measure recurrence by support, the fraction of validation samples to which a rule applies. Consistency is captured by $P(\text{tail} \mid \text{head})$, which measures how reliably a rule's tail is observed when its head is satisfied. Finally, the normalized class entropy H_{class} quantifies cross-class reuse, with values near 0 indicating class-specific rules and values near 1 indicating broad reuse across classes. Exact metric definitions are provided in Appendix B.2. In Table 1 we summarize these quantities across all layer pairs. To illustrate how these properties evolve across depth, Figure 4 shows their layer-wise distributions for CLIP ViT-B/32 on IMAGEMEOW as an example.

Overall, the recovered rules remain compact despite the large number of dependencies discovered, with heads and tails typically containing only around ten concepts. This keeps individual rules easy to inspect while still capturing multi-concept relations. Support, conditional tail probability, and class entropy further show that many of these rules recur across samples and are reused across classes, while Figure 4 illustrates how their prevalence and complexity evolve across depth.

Table 1: Structural and statistical properties of the rule banks recovered by CONCORD on the validation set, averaged across all consecutive layer pairs for the different models and datasets. We report the number of rules, mean head and tail sizes, support, $P(\text{tail} \mid \text{head})$, and H_{class}.

Model	Dataset	# Rules	$\overline{\lvert H\rvert}$	$\overline{\lvert T\rvert}$	$\overline{\text{Supp}}$ (%)	$\overline{P(\text{tail} \mid \text{head})}$ (%)	$\overline{H_{\text{class}}}$
ViT-B/32	IMAGENETTE	55,663	13	9	40.08	48.48	0.93
	IMAGEMEOW	56,119	13	10	36.05	50.46	0.96
	IMAGEWOOF	49,083	8	9	41.40	57.06	0.96
RN50	IMAGENETTE	95,440	12	9	10.15	52.86	0.66
	IMAGEMEOW	70,794	14	9	12.12	53.85	0.67
	IMAGEWOOF	92,108	12	10	11.34	59.04	0.66

5.2 EVALUATING THE COMPOSED CONCEPTOME

We next evaluate the CONCEPTOME obtained by composing the rule banks of CONCORD. Although this structure is learned independently of the predictions of the model, we test whether it preserves the original computation, whether its concepts are sufficient and necessary for that computation, and whether the recovered dependencies have functional influence. We complement these evaluations by inspecting an example pathway through the CONCEPTOME that is shared across multiple samples, illustrating how recurring computation can be traced and interpreted.

Computational fidelity We first assess how much of the active concept representation is retained and whether this suffices to preserve the original prediction. We report active-concept retention, the fraction of originally active concepts retained by the activated CONCEPTOME, and agreement, the fraction of samples for which the original prediction is preserved. Following Marks et al. (2025), we

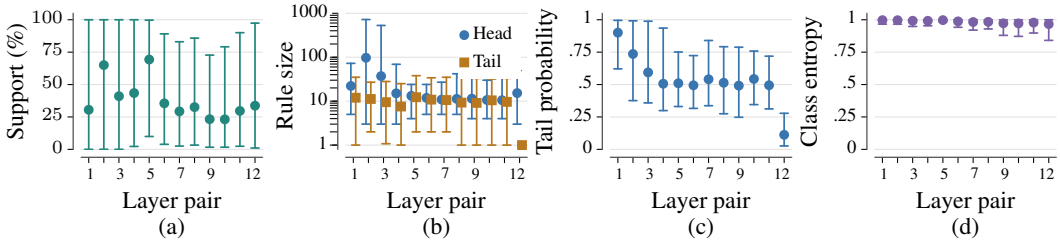


Figure 4: **Layer-wise characteristics of the rules recovered by CONCORD for CLIP ViT-B/32 on IMAGEMEOW**. Each x-axis position denotes a consecutive layer pair $\ell \to \ell + 1$, with intervals spanning the 2.5th to 97.5th percentiles. We show (a) support, (b) head and tail sizes on a logarithmic scale, (c) $P(\text{tail} \mid \text{head})$, and (d) support-weighted normalized class entropy. The rules remain compact and broadly reused across classes, while their prevalence and size vary across depth.

assess whether the retained structure is sufficient and necessary for preserving model computation using faithfulness and completeness, respectively. Faithfulness measures the computation preserved by the retained structure relative to the full model and an empty graph, while completeness measures how much computation is lost when this structure is removed. Finally, to test whether the recovered cross-layer dependencies exhibit functional influence, we report tail disappearance, which measures how often suppressing upstream concepts causes their connected downstream concepts to become inactive. Exact metric definitions and intervention details are provided in Appendix B.3.

To contextualize these measurements, we additionally report RRM (Kim et al., 2025) for CLIP ViT-B/32. RRM is a sample-specific circuit discovery method that identifies a sparse sub-network tied to the prediction of an individual sample. This differs fundamentally from our setting, where a single network-wide CONCEPTOME is learned at the population level and shared across samples and classes rather than optimized separately for each prediction. We therefore use RRM as a reference rather than a directly comparable baseline. To make this reference informative, we evaluate RRM at the operating point whose whole-graph active-concept retention is closest to that of the CONCEPTOME.

Table 2: Computational fidelity of the composed CONCEPTOME. For ViT-B/32, RRM is included as a sample-specific, prediction-tied reference at the operating point with the closest whole-graph active-concept retention. All values are percentages. Across datasets and architectures, the CONCEPTOME preserves much of the original model computation despite being learned as a single population-level structure rather than separately for each prediction.

Architecture	Dataset	Method	Active ret.	Agreement	Faithfulness	Completeness	Disappearance
ViT-B/32	IMAGENETTE	CONCEPTOME	72.6	79.1	80.2	100.0	50.5
		RRM	71.4	86.3	98.7	100.0	45.1
	IMAGEMEOW	CONCEPTOME	74.7	79.6	92.9	100.0	50.5
		RRM	70.7	80.5	89.9	100.0	45.0
	IMAGEWOOF	CONCEPTOME	75.2	64.8	86.7	100.0	41.3
		RRM	70.8	60.2	89.6	100.0	45.0
RN50	IMAGENETTE	CONCEPTOME	89.6	89.1	99.5	99.8	28.0
	IMAGEMEOW	CONCEPTOME	89.8	93.0	100.0	97.1	28.6
	IMAGEWOOF	CONCEPTOME	91.3	88.5	99.2	100.0	29.5

In Table 2 we report active-concept retention, agreement, faithfulness, completeness, and tail disappearance. RRM is included only for CLIP ViT-B/32, since it is specific to Vision Transformers. The CLIP RN50 results provide a complementary evaluation of the CONCEPTOME on a convolutional architecture.

Although recovered as a single population-level structure rather than per prediction, the CONCEPTOME preserves much of the original computation. On ViT-B/32, retaining roughly three quarters of the active concepts yields 80–93% faithfulness and 100% completeness across all three datasets, and RN50 similarly achieves high agreement and near-complete faithfulness and completeness. Tail disappearance further shows that suppressing concepts selected by the recovered structure propagates to their connected downstream concepts, providing evidence that the learned dependencies reflect functional relationships in the model.

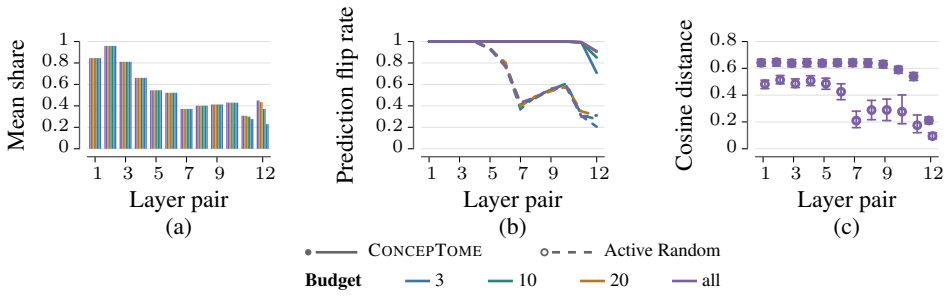


Figure 5: **Layer-wise effects of interventions along prediction-linked ConceptTome pathways for CLIP ViT-B/32 on ImageMeow.** Budgets indicate the number of top-ranked final rules used for pathway tracing. (a) Fraction of active SAE concepts suppressed, (b) prediction flip rate against the matched Active Random control, and (c) final-representation cosine distance for the *all* budget, shown as medians with interquartile intervals. ConceptTome pathways cause more prediction changes in later layers and consistently larger representation movements.

Interventional effect Having established that the ConceptTome preserves a good portion of the model computation at the global level, we next examine whether the pathways it associates with a prediction identify concepts with particular influence on that prediction. We continue the analysis on CLIP ViT-B/32 with ImageMeow as an example. For each validation sample, we rank the final rules linked to the predicted label by their importance in the ConceptTome and select the top 3, 10, 20, or *all* such rules. We trace their pathways through the ConceptTome backwards and suppress the active concepts linked to the selected prediction rules at each layer. To separate the effect of the recovered structure from that of suppressing active concepts alone, we compare against an Active Random control that randomly suppresses the same number of active concepts at each layer.

We measure the resulting effect at both the model output and final representation level. Prediction flip rate measures how often the original prediction changes, while representation movement is measured by the cosine distance $1 - \cos(e(x), e^{(\ell)}_{\text{int}}(x))$ between the original and intervened final image embeddings. We additionally report the fraction of originally active concepts suppressed at each budget to contextualize the scale of the interventions.

Results in Figure 5 show that interventions along the recovered ConceptTome pathways have a clear effect on model computation, and that this effect cannot be explained by intervention size alone. In later layers, where these pathways still involve relatively few active concepts, ConceptTome interventions change the prediction more often than the matched control and produce larger movements in the final representation. Toward earlier layers, the traced pathways fan out and the intervened share becomes large, causing prediction flip rates to approach each other and often saturate. Even there, ConceptTome interventions generally move the final representation farther than the matched control.

Qualitative pathway analysis To illustrate how the recovered structure can be inspected in practice, we examine one pathway shared across multiple samples in the CLIP Vit-B/32 ConceptTome learned on ImageMeow, leading to the cheetah class. The example is presented in Figure 6 and it showcases how broad and increasingly class-specific cues appear together.

Figure 6 shows how these concepts are progressively combined. At $L07$, circular shapes, generic spotting, spotted animal bodies, and round tail regions are combined into generic and animal-specific spotting at $L08$. These are joined by concepts such as the region below an animal's face, an animal's back, and checked texture, yielding a more localized spotted-neck concept at $L09$. The next rule combines this with furred animal bodies, yellow animal appearance, spotted bodies, and a radiating strand-like pattern, producing both generic spotting and spotted animal head and body concepts at $L10$. These are then combined with an animal tail, a cat concept, and black-and-white spotting to form a spotted-animal-body concept at $L11$. At this late stage, the pathway still draws on diverse reusable concepts, including cat ears and other feline regions, a text-sensitive concept responding to the word "image", and two striped-pattern concepts. The striped concepts may initially seem unexpected for a spotted animal, but they align naturally with the ringed pattern of a cheetah's tail, which already appears explicitly among the concepts entering the preceding rule. Together,

these concepts give rise at $L12$ to both an animal-in-the-wild concept and a distinctly cheetah-like face. The final rule combines these with broader concepts for yellow furry animals, big-cat bodies, and small-cat faces before reaching the cheetah class. Rather than following a simple progression from generic to class-specific concepts, the pathway preserves useful evidence while incorporating additional cues. Several concepts remain semantically consistent across adjacent layers, such as recurring spotted-animal and feline concepts, suggesting that related information is carried forward while being recombined with new evidence. This is consistent with residual-stream computation, where existing information can remain available while later blocks add new concepts, and illustrates how class-specific evidence can emerge from successive combinations of reusable concepts and rules.

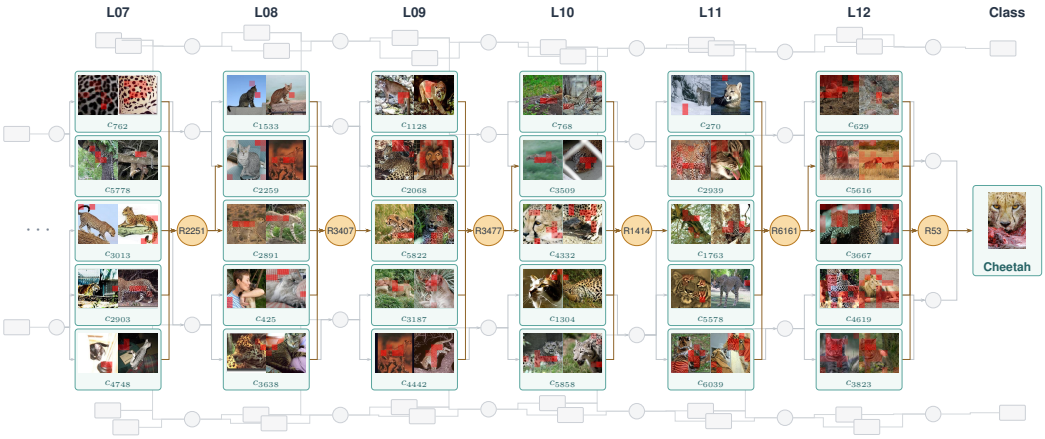


Figure 6: A pathway in our CONCEPTOME leading to the cheetah class. Green cards show concepts and amber circles show rules; faint gray links are illustrative.

6 CONCLUSION

We presented CONCORD, an approach for discovering recurring cross-layer dependencies between sparse concepts and composing them into a network-wide CONCEPTOME. The resulting graph provides a symbolic view of how concepts are combined and reused across the network, while remaining queryable for individual samples and classes. Across CLIP ViT-B/32 and RN50, we recover compact multi-concept rules that recur across samples and are frequently shared across classes. The composed CONCEPTOME retains part of the model computation, while interventions show that prediction-linked pathways have greater influence on predictions and final representations than matched active-random controls, particularly in later layers.

Limitations The CONCEPTOME provides an approximate view of model computation and naturally depends on the quality of the underlying SAE concepts. Spatial aggregation and binarization simplify the concept representation by discarding information about location and activation magnitude. In addition, the learned rules describe recurring cross-layer dependencies and should not be interpreted as establishing every individual connection as a causal mechanism, although our interventions provide evidence that the recovered structure is functionally relevant. Finally, our current evaluation is limited to two CLIP backbones and selected ImageNet subsets.

Future work A natural next step is to extend CONCORD to a broader range of architectures and model scales. More interestingly, comparing the CONCEPTOMEs recovered from related models could reveal which concept structures are preserved, reorganized, or newly introduced as models change in size, architecture, pruning, or distillation. The same representation could also support comparisons between natural and unusual inputs, for example by studying whether adversarial or distribution-shifted samples follow systematically different pathways through the network.

AI USE STATEMENT

Generative AI tools were used as supporting tools throughout parts of this work, including code assistance and debugging, refinement of manuscript wording, LaTeX and figure preparation, and literature search. The accompanying web interface was largely generated with AI assistance based on author-provided requirements, design choices, iterative feedback, and validation. The core research ideas, methodology, experimental setup, and conclusions were developed by the authors. All AI-assisted suggestions, code, analyses, and written material were reviewed and tested by the authors. Final decisions regarding the methodology, experiments, interpretation of results, and manuscript content were made by the authors.

REPRODUCIBILITY STATEMENT

Our experiments use publicly accessible ImageNet-1K data (Russakovsky et al., 2015) and pretrained CLIP models available through OpenCLIP (Radford et al., 2021; Ilharco et al., 2021). The main paper and Appendices A and B describe the model and representation choices, SAE training, rule-learning procedures, intervention protocols, evaluation metrics, and experimental settings needed to reproduce our experiments. The supplementary material provides all source code and configuration files needed to train SAEs, learn rules, and perform the interventional evaluation. Together, these resources enable reproduction of the discovered CONCEPTOME and its evaluation.

REFERENCES

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning, 2023. Transformer Circuits Thread.

Kirill Bykov, Laura Kopf, Shinichi Nakajima, Marius Kloft, and Marina M.-C. Höhne. Labeling neural representations with inverse recognition. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. doi: 10.23915/distill.00024.003.

Hang Chen, Jiaying Zhu, Xinyu Yang, and Wenya Wang. Rethinking circuit completeness in language models: And, or, and adder gates. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, 2023.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Remi Cadene, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Jonas Fischer, Anna Olah, and Jilles Vreeken. What's in the box? exploring the inner life of neural networks with robust rules. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 2021.

Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, 2025.

Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, 2019.

Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. *arXiv preprint arXiv:2406.03662*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Yaojie Hu and Jin Tian. Neuron dependency graphs: A causal abstraction of neural networks. In *International Conference on Machine Learning*, 2022.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, 2018.

Jinyeong Kim, Junhyeok Kim, Yumin Shim, Joohyeok Kim, Sunyoung Jung, and Seong Jae Hwang. Interpreting vision transformers via residual replacement model. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Matthew Kowal, Richard P. Wildes, and Konstantinos G. Derpanis. Visual concept connectome (vcc): Open world concept discovery and their interlayer connections in deep models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Dahee Kwon, Sehyun Lee, and Jaesik Choi. Granular concept circuits: Toward a fine-grained circuit discovery for concept representations. In *IEEE/CVF International Conference on Computer Vision*, 2025.

Michael A. Lepori, Alexa R. Tartaglini, Wai Keen Vong, Thomas Serre, Brenden M. Lake, and Ellie Pavlick. Beyond the doors of perception: Vision transformers represent relations between objects. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Tang Li, Yanlin Chen, Mengmeng Ma, and Xi Peng. Inside the visual mind: Neuroscience-motivated concept circuits for interpreting and steering vision transformers. In *International Conference on Machine Learning*, 2026.

Alireza Makhzani and Brendan Frey. k-sparse autoencoders. In *International Conference on Learning Representations*, 2014.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, 2025.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *Advances in Neural Information Processing Systems*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Nils Philipp Walter, Jonas Fischer, and Jilles Vreeken. Finding interpretable class-specific patterns through efficient neural search. In *AAAI Conference on Artificial Intelligence*, 2024.

Yifan Wang, Yifei Liu, Yingdong Shi, Changming Li, Anqi Pang, Sibei Yang, Jingyi Yu, and Kan Ren. Discovering influential neuron path in vision transformers. In *International Conference on Learning Representations*, 2025.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2014.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *European Conference on Computer Vision*, 2018.

Nina Żukowska, Wolfgang Stammer, Bernt Schiele, and Jonas Fischer. Seeing through circuits: Faithful mechanistic interpretability for vision transformers. *arXiv preprint arXiv:2604.14477*, 2026.

A CONCORD TRAINING DETAILS

This section provides additional training details for CONCORD introduced in Section 4.1, including the Monte Carlo approximation of the objective, the regularization terms, the straight-through gradient estimator, the selection of deterministic rule bank, and source-masked training.

A.1 MONTE CARLO OBJECTIVE

Recall that $G \sim p_\theta$ denotes a deterministic rule graph induced by the continuous parameters θ. The expectations and variance in Eq. 1 and 3 are approximated using M independently sampled graphs,

$$G_1, \ldots, G_M \sim p_\theta.$$

For an input x, let

$$\mathcal{L}^{(m)}_{\text{target}}(x) = \mathcal{L}_{\text{target}}(x, G_m), \qquad \overline{\mathcal{L}_{\text{target}}}(x) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}^{(m)}_{\text{target}}(x).$$

The Monte Carlo estimate of the per-sample objective is

$$\widehat{\mathcal{L}_{\text{CONCORD}}}(x) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{L}_{\text{target}}^{(m)}(x) + \lambda_{\text{var}}\frac{1}{M}\sum_{m=1}^{M}\left(\mathcal{L}_{\text{target}}^{(m)}(x) - \overline{\mathcal{L}_{\text{target}}}(x)\right)^2 + \lambda_{\text{bin}}\mathcal{R}_{\text{bin}}(W_H, W_T) + \lambda_{\text{rule}}\mathcal{R}_{\text{rule}}(W_H, W_T). \tag{4}$$

The first term averages the target concept-presence loss across sampled binary graphs, while the variance term discourages solutions whose loss depends strongly on a particular graph sample. During training, we minimize the minibatch average of $\widehat{\mathcal{L}_{\text{CONCORD}}}(x)$.

A.2 REGULARIZATION

The regularization in Eq. 3 acts on the continuous edge probabilities to encourage discrete connections and compact rules.

Binarization regularization The regularizer $\mathcal{R}_{\text{bin}}$ encourages the continuous edge probabilities in W_H and W_T to approach either 0 or 1. For an edge probability $w \in [0, 1]$, we define

$$r_{\text{bin}}(w) = \min\{r(w), r(w-1)\},$$

where

$$r(w) = \kappa_{\text{bin}}|w| + \gamma_{\text{bin}}w^2.$$

The complete binarization regularizer is

$$\mathcal{R}_{\text{bin}}(W_H, W_T) = \sum_{W\in\{W_H,W_T\}}\sum_{i,j} r_{\text{bin}}(W_{ij}).$$

Its minima at 0 and 1 encourage the learned edge probabilities to become increasingly discrete during training.

Rule complexity regularization To favor compact rules, we penalize large heads and tails. For a rule-concept weight matrix

$$W \in [0, 1]^{R\times D},$$

we define

$$\mathcal{R}_{\text{size}}(W) = \frac{1}{R}\sum_{r=1}^{R}\left[\max\left(1, \sum_{j=1}^{D}\left(W_{rj} - \frac{1}{D}\right)\right)\right]^2.$$

The resulting rule-complexity regularizer is

$$\mathcal{R}_{\text{rule}}(W_H, W_T) = \mathcal{R}_{\text{size}}(W_H) + \mathcal{R}_{\text{size}}(W_T^\top).$$

This encourages compact rule heads and tails while accounting for the initialization baseline of the edge probabilities.

A.3 STRAIGHT-THROUGH GRADIENT ESTIMATION

Sampling binary edges, discretizing the rule bias, and applying binary rule activations make the forward computation non-differentiable. We therefore maintain continuous parameters for optimization and use straight-through estimators during the backward pass.

Let $\boldsymbol{\rho} \in \{0, 1\}^R$ denote the binary activations of the rule units and let g denote the upstream gradient. We decompose

$$g^+ = \max(g, 0), \qquad g^- = \min(g, 0).$$

Rule-to-tail layer For the output layer, we use a standard STE and pass the upstream gradient through the binary activation unchanged,

$$\nabla_{u_{\text{out}}}\mathcal{J} = g,$$

where $\mathcal{J}$ denotes the scalar training objective.

Head-to-rule layer For the hidden rule layer, we use asymmetric gradient routing. Gradients to the rule pre-activations are propagated only through rules that were active in the forward pass,

$$\nabla_u \mathcal{J} = \boldsymbol{\rho} \odot g. \tag{5}$$

Consequently, both positive and negative gradients modify head weights only for active rules.

The bias receives a different credit signal. For sample i, define

$$\chi_i = \mathbb{I}[\mathbf{b} = u_i] \odot \mathbb{I}[u_i < 0],$$

which denotes the implementation condition under which the bias of an inactive rule may be updated for sample i. Summing over a mini-batch $\mathcal{B}$, its gradient is

$$\nabla_{\mathbf{b}} \mathcal{J} = \sum_{i \in \mathcal{B}} \left[\boldsymbol{\rho}_i \odot g_i^+ + (1 - \boldsymbol{\rho}_i) \odot g_i^- \odot \chi_i \right]. \tag{6}$$

Thus, positive gradients update the bias only for active rules. Negative gradients update the weights of active rules but do not modify their bias. For inactive rules, negative gradients may modify the bias only when

$$b = u_i \qquad \text{and} \qquad u_i < 0,$$

while their head weights remain unchanged.

This asymmetric routing prevents inactive rules from receiving arbitrary updates to their heads while still allowing their activation condition to adapt when the bias prevents a rule from firing.

A.4 Selecting a deterministic rule bank

The learned parameters define a distribution over binary rule graphs. For deployment and evaluation, however, we require a single deterministic rule bank for each layer pair. We therefore select one representative binary graph from the learned distribution using held-out selection data.

During training, the held-out data are used to choose the retained checkpoint. From this checkpoint, we draw multiple deterministic binary graphs,

$$G_1, \ldots, G_S \sim p_{\theta^\star}.$$

and evaluate them on the same selection data. The best-performing sampled graph according to the predefined selection criterion is retained as the final rule bank,

$$\mathcal{R}^{(\ell,\ell+1)}.$$

This yields a deterministic rule bank that preserves the discrete rule semantics used during training and can be directly composed across layers.

A.5 Learning under composition

Once rule banks are composed, the concepts available at layer ℓ are restricted by the structure propagated through the preceding rule banks. We therefore train each rule bank directly on source-masked representations.

Let

$$m^{(\ell)}(x) \in \{0,1\}^{d_\ell}$$

denote the concept mask induced at layer ℓ by the preceding rule structure. The filtered concept-presence representation is

$$\widetilde{c}^{(\ell)}(x) = c^{(\ell)}(x) \odot m^{(\ell)}(x),$$

where $\odot$ denotes element-wise multiplication.

To determine the corresponding target representation, we apply the same mask to the continuous concept activations,

$$\widetilde{z}^{(\ell)}_{j,p}(x) = m^{(\ell)}_j(x) z^{(\ell)}_{j,p}(x).$$

Table 3: TopK SAE configurations used for each backbone.

Backbone	Blocks	E	k	Dictionary size m	Training
ViT-B/32	1–4	6	128	4,608	20 epochs
	5–7	8	128	6,144	20 epochs
	8–12	8	160	6,144	20 epochs
RN50	1–3	8	96	2,048	20k updates
	4–7	8	128	4,096	20k updates
	8–10	8	128	8,192	20k updates
	11–13	8	160	8,192	20k updates
	14–16	8	160	16,384	10k updates

The decoder associated with the SAE at layer ℓ is denoted by h_ℓ. We decode the masked concept representation back into the model activation space,

$$\widetilde{a}^{(\ell)}(x) = h_\ell\left(\widetilde{z}^{(\ell)}(x)\right),$$

and propagate it through the original computation between the two selected layers,

$$\widetilde{a}^{(\ell+1)}(x) = f_\ell\left(\widetilde{a}^{(\ell)}(x)\right).$$

The propagated activation is then encoded using the SAE at layer $\ell + 1$,

$$\widetilde{z}^{(\ell+1)}(x) = g_{\ell+1}\left(\widetilde{a}^{(\ell+1)}(x)\right),$$

and converted to image-level concept presence using the same criterion as for the original representation,

$$\widetilde{c}_j^{(\ell+1)}(x) = \mathbb{I}\left[\max_p \widetilde{z}_{j,p}^{(\ell+1)}(x) > 0\right].$$

Each rule bank is therefore trained on the pair $\left(\widetilde{c}^{(\ell)}(x), \widetilde{c}^{(\ell+1)}(x)\right)$. The intervention is introduced at the source layer, while the corresponding target is determined by propagating the masked representation through the original network. We train CONCORD on these pairs using the same objective defined in Eq. 1.

B EXPERIMENTAL DETAILS

B.1 SAE CONFIGURATIONS

We train a separate TopK SAE for each block, with dictionary size $m = Ed$, where d is the block representation dimension and E the expansion factor. Table 3 summarizes the architecture-specific configurations.

Both architectures use a learning rate of 2×10^{-4}, with sparsity controlled directly by TopK. RN50 training uses an activation buffer of 256,000 vectors. For each backbone, the same SAE checkpoints are shared across all three downstream datasets.

B.2 EVALUATION METRICS: CONCORD

Let H_r and T_r denote the head and tail of rule r. We consider a rule active when its deterministic rule neuron activates,

$$\rho_r(x) = \mathbb{I}[h_r(x) > \epsilon],$$

where $h_r(x)$ denotes the rule-neuron output. This is the implemented notion of a satisfied head and does not require every concept in H_r to be present.

The evaluated rule bank is

$$\mathcal{R} = \left\{ r : |H_r| > 0,\ |T_r| > 0,\ \sum_{x \in \mathcal{D}_{\text{train}}} \rho_r^{\text{chain}}(x) > 0 \right\}.$$

Rule eligibility is determined using activations propagated through the composed rule chain, while the validation metrics below are computed from concepts obtained from the original backbone.

Structural complexity We report the number of retained rules and their mean head and tail sizes,

$$N_{\text{rules}} = |\mathcal{R}|, \qquad \overline{|H|} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} |H_r|, \qquad \overline{|T|} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} |T_r|.$$

Support For $N = |\mathcal{D}_{\text{val}}|$, rule support is

$$\text{Supp}(r) = \frac{1}{N}\sum_{x\in\mathcal{D}_{\text{val}}} \rho_r(x), \qquad \overline{\text{Supp}} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \text{Supp}(r).$$

Tail consistency Let $b_j(x)$ indicate whether tail element j is present. For SAE concepts,

$$b_j(x) = \mathbb{I}[z_j(x) > 10^{-12}],$$

while at the endpoint,

$$b_j(x) = \mathbb{I}[\arg\max f(x) = j],$$

using the original model prediction. For each tail element,

$$P(t_j \mid r) = \frac{\sum_{x\in\mathcal{D}_{\text{val}}} \rho_r(x) b_j(x)}{\sum_{x\in\mathcal{D}_{\text{val}}} \rho_r(x)},$$

and the reported rule-level quantity is

$$P(\text{tail} \mid \text{head})_r = \frac{1}{|T_r|}\sum_{j\in T_r} P(t_j \mid r).$$

This measures the expected fraction of tail elements present when the rule activates, rather than the probability that the complete tail is present simultaneously. Endpoint rules contain a single tail class.

Normalized class entropy Let $\hat{y}(x)$ denote the original model prediction and C the number of classes observed in the validation set. We first compute the within-class activation rate

$$q_{r,c} = \frac{\sum_x \rho_r(x)\mathbb{I}[\hat{y}(x) = c]}{\sum_x \mathbb{I}[\hat{y}(x) = c]}, \qquad p_{r,c} = \frac{q_{r,c}}{\sum_{c'=1}^{C} q_{r,c'}}.$$

The normalized class entropy is

$$H_{\text{class}}(r) = -\frac{\sum_{c=1}^{C} p_{r,c}\log p_{r,c}}{\log C}, \qquad 0\log 0 := 0.$$

A value of 0 indicates activation in only one class, while 1 corresponds to equal within-class activation rates across all observed classes.

For the aggregate results, tail consistency and class entropy are averaged equally across rules for which they are defined. Rules with zero validation support remain in the structural and support statistics, but their consistency and entropy are undefined. Support and tail consistency are multiplied by 100 when reported as percentages.

B.3 Evaluation Metrics: ConceptTome

Let $f(x) \in \mathbb{R}^K$ denote the original model logits and

$$\hat{y}(x) = \arg\max_k f_k(x)$$

its prediction. We consider three intervened models. f_G retains the concepts selected by the activated ConceptTome, $f_\varnothing$ replaces all concepts in the evaluated intervention scope with the reference baseline, and $f_{\setminus G}$ removes the selected concepts using the complementary masks. All reported interventions use median replacement while preserving the SAE reconstruction residual; hence, the empty reference does not correspond to setting hidden activations to zero.

Active-concept retention For filter pair ℓ, let $m_{\ell j}(x)$ indicate whether concept j is selected by the activated CONCEPTOME and $a_{\ell j}(x)$ whether it is active before that pair's intervention. We compute

$$AR_\ell = \frac{\sum_x \sum_j m_{\ell j}(x) a_{\ell j}(x)}{\sum_x \sum_j a_{\ell j}(x)}, \qquad AR = \frac{1}{L} \sum_{\ell=1}^{L} AR_\ell.$$

Counts are pooled across samples within each filter pair and then averaged equally across pairs; the classification endpoint is excluded.

Prediction agreement Agreement measures whether the filtered backbone preserves the original model prediction,

$$A = \frac{1}{N} \sum_{x \in \mathcal{D}} \mathbb{I}\left[\arg\max_k f_{G,k}(x) = \hat{y}(x) \right].$$

This compares predictions of the filtered and original backbones and does not measure ground-truth accuracy.

Faithfulness and completeness Following Marks et al. (2025), we quantify retained computation using the mean-centered logit of the original predicted class,

$$s(g, x) = g_{\hat{y}(x)}(x) - \frac{1}{K} \sum_{k=1}^{K} g_k(x), \qquad S(g) = \frac{1}{N} \sum_x s(g, x).$$

Let

$$\mathrm{clip}(u) = \min(1, \max(0, u)).$$

Faithfulness is

$$F = \mathrm{clip}\left(\frac{S(f_G) - S(f_\varnothing)}{S(f) - S(f_\varnothing)} \right),$$

while the reported removal-based completeness is

$$C = 1 - \mathrm{clip}\left(\frac{S(f_{\setminus G}) - S(f_\varnothing)}{S(f) - S(f_\varnothing)} \right).$$

High faithfulness indicates that retaining the CONCEPTOME preserves the original computation, while high completeness indicates that removing it leaves little computation beyond the empty reference. The normalization is applied to dataset-mean scores rather than individual sample ratios. If

$$|S(f) - S(f_\varnothing)| \le 10^{-12},$$

the corresponding score is undefined. The complementary masks used for $f_{\setminus G}$ are fixed from the retained-CONCEPTOME pass.

Tail disappearance. Tail disappearance is evaluated separately as a pairwise intervention on the original backbone rather than on the sequentially filtered chain. For sample x and layer pair ℓ, let $T_\ell(x)$ denote the union of downstream concepts connected to fired rules. We evaluate only downstream concepts that are initially active,

$$E_\ell(x) = \left\{ j \in T_\ell(x) : c_j^{(\ell+1)}(x) = 1 \right\}.$$

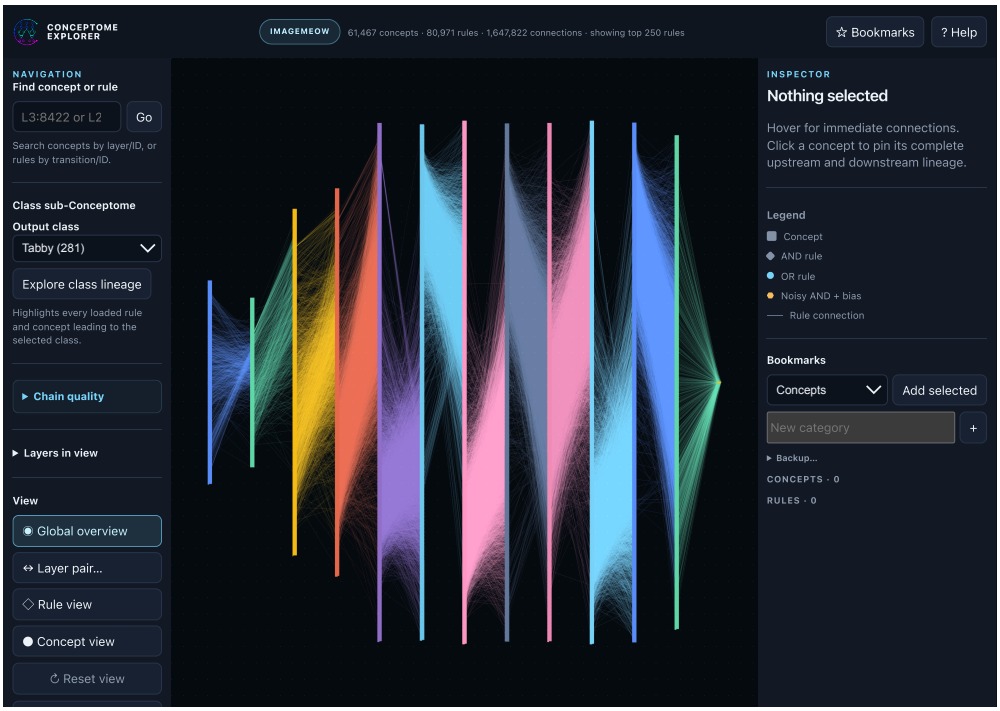
We suppress the union of the corresponding fired rules' upstream antecedents, propagate the intervened representation through the original model, and denote downstream presence afterward by $c_j^{(\ell+1),\mathrm{off}}(x)$. Tail disappearance is

$$D = \frac{\sum_{\ell,x} \sum_{j \in E_\ell(x)} \mathbb{I}\left[c_j^{(\ell+1),\mathrm{off}}(x) = 0 \right]}{\sum_{\ell,x} |E_\ell(x)|}.$$

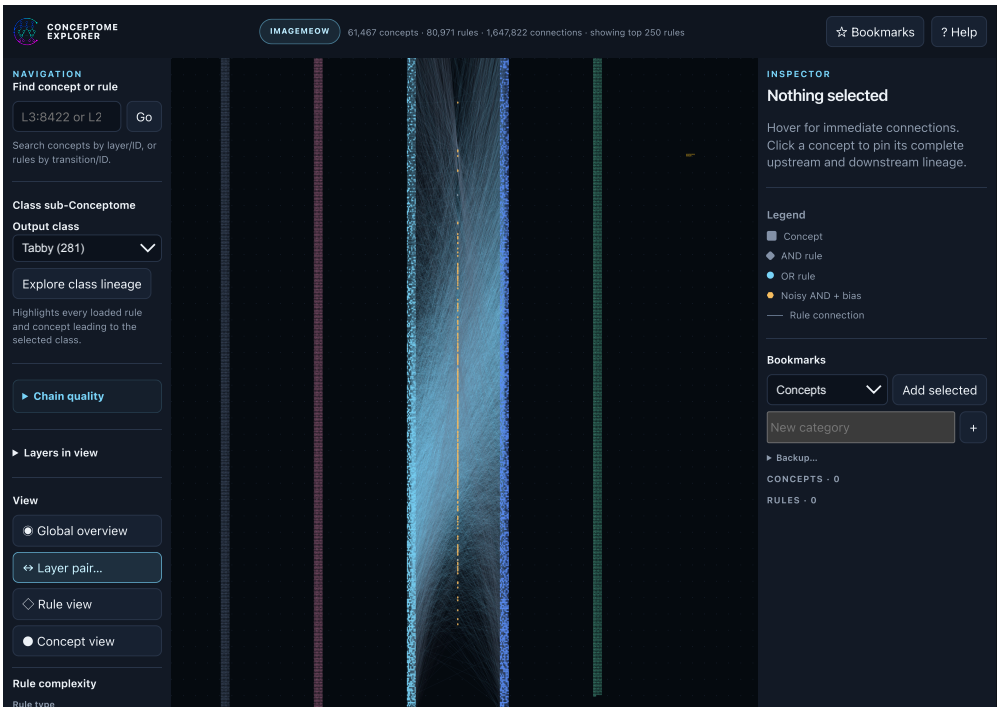
This pools sample–pair–tail observations rather than averaging equally across rules. Shared tail concepts are counted once per sample and layer pair. The intervention preserves the reconstruction residual, and image-level masks are broadcast across token or spatial positions.

C INTERACTIVE CONCEPTOME EXPLORER

We develop an interactive interface for exploring recovered CONCEPTOMEs at different levels of detail. The explorer supports navigation through the full layered structure, individual layer pairs, rules, and concepts, together with representative concept visualizations and rule statistics. It also supports image-level analysis by highlighting the structure active for a selected image and enabling interventions on concepts and rules to inspect their effect on the model output. Figures 7–9 show representative views of the interface.

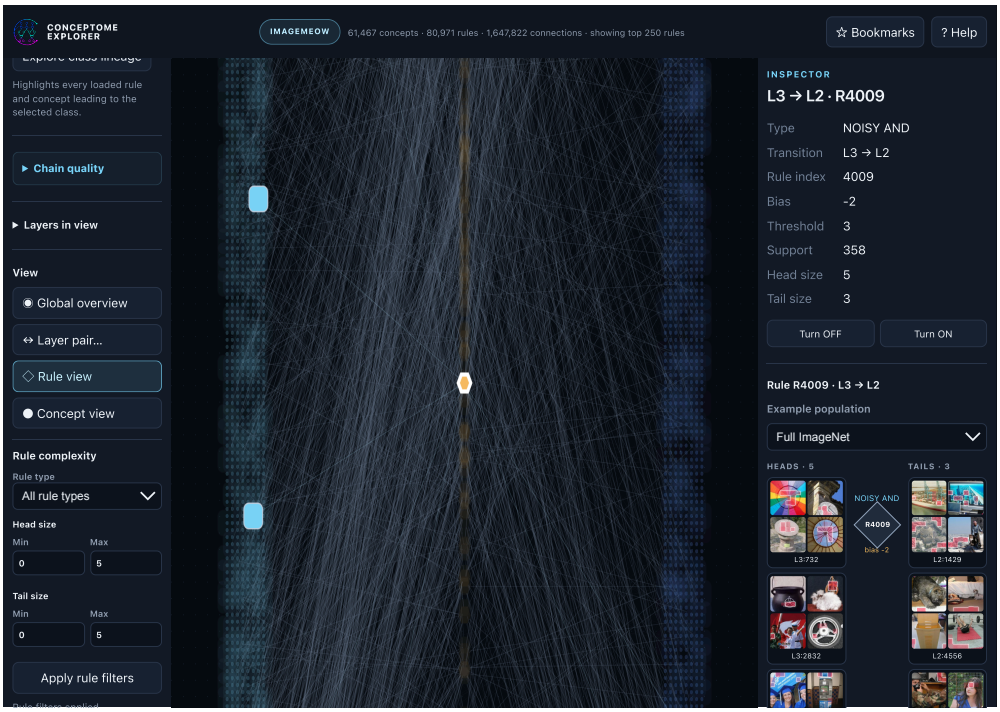


(a) Global view of the layered CONCEPTOME and its cross-layer connectivity.

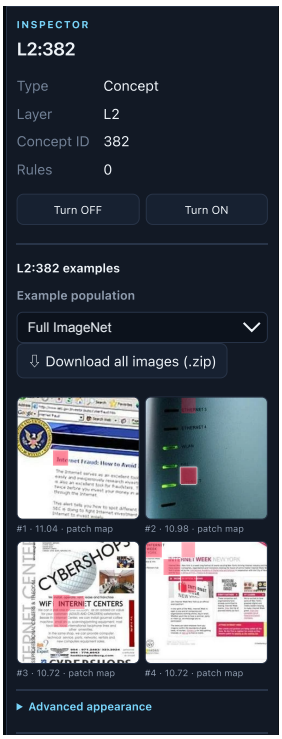


(b) Layer-pair view for inspecting the structure between two selected layers.

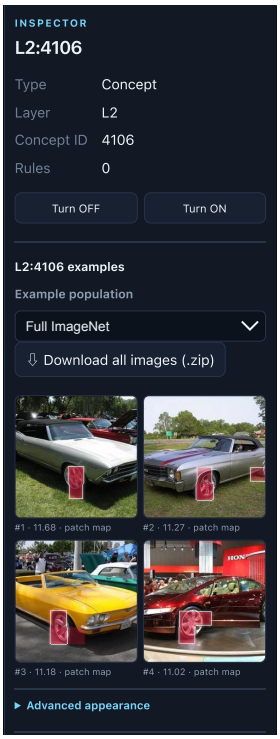
Figure 7: Structural views of the CONCEPTOME Explorer. Users can move from the complete network-wide graph to individual layer pairs while retaining the surrounding context.



(a) Rule-level inspection showing rule properties and representative head and tail concepts.

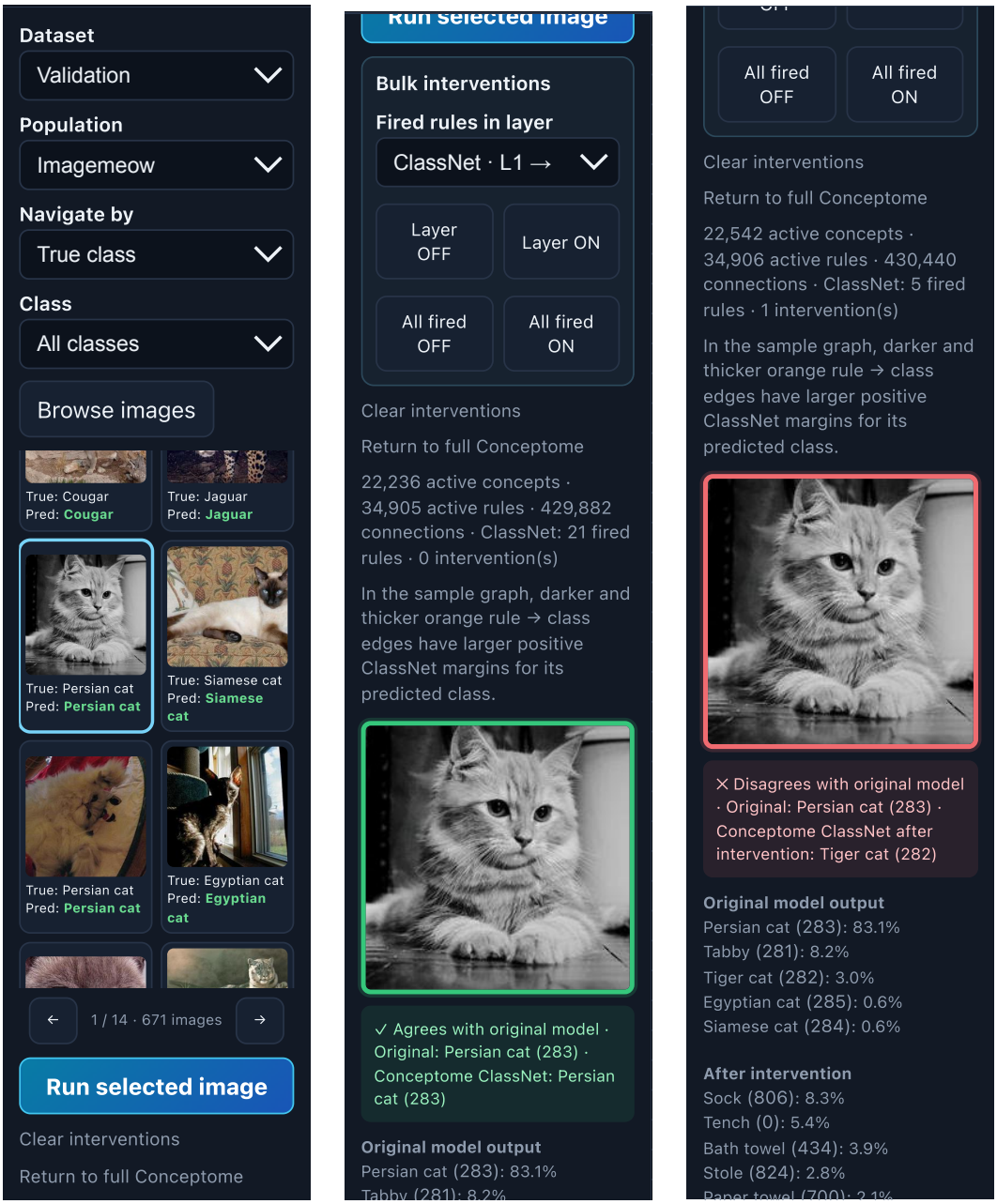


(b) Text-related concept.



(c) Wheel-related concept.

Figure 8: Detailed inspection in the CONCEPTOME Explorer. (a) A selected rule with its type, threshold, support, and representative head and tail concepts. (b,c) Individual concept views with representative examples and spatial attribution maps.



(a) Browsing images and selecting a sample for analysis.

(b) Intervention controls and the original prediction for the selected image.

(c) Prediction changes after intervening on the active structure.

Figure 9: Image-level intervention workflow. A validation image is selected, its active CONCEPTOME structure can be modified through concept- or rule-level interventions, and the resulting changes in the model output are displayed directly.