Self-Explaining Classifiers with Sparse Autoencoders and LLMs

Diego Bonilla Salvador
Independent Researcher
diegobonila@gmail.com

Abstract

We present a simple, end-to-end pipeline that turns an ordinary image classifier into a *self-explaining* system. Starting from a trained CNN, we freeze the backbone and train a token-space Top-K Sparse Autoencoder (SAE) on a late convolutional feature map. Each SAE feature ("neuron") is then *characterized* by (i) pointwise mutual information (PMI) with classes, (ii) a *logit-lens* projection that maps its decoder direction to class logits, and (iii) a closed-form causal Δ^- score that estimates the class-logit drop under feature ablation. We convert each neuron's numeric profile and top-activating thumbnails into a short, human-readable *concept* using a small LLM. At inference, we explain a prediction by surfacing the top-activating concepts and summarizing their evidence into one succinct sentence. The approach is faithful by construction for linear heads (closed-form Δ^-), local to the input (instance-conditioned top-K features), and model-agnostic at training time (requires only frozen features). We provide math, algorithms, and a reference implementation covering training the classifier, training the SAE, computing per-neuron summaries, LLM conversion to natural-language concepts, and an at-inference "why" generator.

1 Introduction

Deep classifiers are accurate but opaque. Classic saliency and class activation methods (e.g., Grad-CAM) highlight pixels that matter but do not expose the *internal concepts* used for the decision [1]. Later work linked hidden units to semantic concepts (Network Dissection) [2] or to user-defined concepts (TCAV) [3], but these typically operate on neurons or linear probes, not task-aligned *features* learned *post hoc* for interpretability.

Recent progress in *sparse autoencoders* (SAEs) shows that dictionary learning on intermediate activations can recover more monosemantic features, especially in LLMs [6, 7, 8]. We adapt these ideas to CNNs for vision, pair them with a lightweight causal/evidence toolkit (PMI, logit-lens, closed-form Δ^-), and use a compact LLM to translate per-feature statistics into human concepts, yielding *faithful, localized, and human-readable* rationales per input.

Our contributions.

- A practical pipeline to obtain *instance-conditioned* natural-language explanations from a standard classifier, via a Top-K SAE trained in token space on a late conv map.
- A *closed-form* causal attribution for each feature with a linear head (class-logit drop under feature knockout), plus PMI and a logit-lens mapping from decoder atoms to class logits.
- An LLM-based neuron-to-concept conversion that turns numeric profiles (and thumbnails) into short human concepts, which are composed at inference into one-sentence rationales.
- Open, compact implementation: classifier training, SAE training, neuron summarization, LLM conversion, and a "why" script that prints top concepts and a one-sentence explanation.

2 Related Work

Pixel-/region-level explanations. Grad-CAM and its variants localize class-relevant evidence in images but do not label the *features* used for the decision [1]. **Concept-level explanations.** Network Dissection matches units to a bank of semantic segmentations [2], and TCAV measures sensitivity to user-defined concepts [3]. These offer global insights but (a) often rely on predefined concept sets or supervision, and (b) do not directly yield per-instance *composable* evidence tied to the model's classifier head. **Logit lens & tuned lens.** Projecting intermediate representations to the logit space ("logit lens") provides layer-wise beliefs in LLMs; Tuned Lens refines this with learned affine translators [4, 5]. We adopt a *static* logit-lens for CNNs by projecting SAE decoder atoms through the frozen linear head, giving class-aligned directions. **Sparse autoencoders for interpretability.** SAEs have been

shown to extract monosemantic features and mitigate polysemanticity/superposition in LLMs [6, 7, 8, 9, 10]. We apply a token-space Top-K SAE to CNN features, add unit-norm dictionary constraints, dead-feature resampling, and use closed-form Δ^- to connect feature use to class evidence.

Why ours? Unlike pixel heatmaps, we return *named concepts* with *causal scores*. Unlike TCAV/Dissection, our features are learned *from the model's own features*, class-aligned via the head, and used *per-instance* with a closed-form causal approximation. Unlike generic LLM explainers, our LLM only *labels* numerically grounded features and we compose them deterministically.

3 Method

3.1 Notation and setup

Let a frozen classifier provide a late conv map $A \in \mathbb{R}^{B\times C\times H\times W}$. Flatten spatial tokens: $a \in \mathbb{R}^{N\times C}$ with $N = BHW$. Standardize per channel using feature stats $\mu, \sigma \in \mathbb{R}^C$:

$$\tilde{a} \;=\; (a-\mu) \oslash \sigma.$$

We train a Top-K SAE on $\tilde{a}$ and then compute per-feature statistics and thumbnails over a stratified subset. (Implementation mirrors the released scripts for classifier training, SAE training, neuron summarization, and the "why" runner.)

3.2 Top-K token-space SAE

Encoder preactivation:

$$s \;=\; W_e\,\tilde{a} + b_e \in \mathbb{R}^{N\times M}, \quad r = \mathrm{ReLU}(s).$$

Top-K sparsity keeps, per token, the K largest entries of r, producing sparse code $z \in \mathbb{R}^{N\times M}$. Decoder with unit-norm columns $D \in \mathbb{R}^{C\times M}$:

$$\hat{a} \;=\; D\,z + b_d.$$

We minimize MSE reconstruction: $\mathcal{L} = \|\hat{a} - \tilde{a}\|_2^2$. We periodically renormalize decoder columns and resample "dead" latents by snapping decoder atoms to randomly sampled standardized tokens; encoder rows are aligned and biases reset. (All implemented in our training loop.)

3.3 Linking features to classes

With a global average pooled feature $p \in \mathbb{R}^C$ and linear head $\mathrm{logits} = W_{\text{out}}p + b$, decoder atoms live in standardized space. Map atom d_i to unstandardized space via $d_i^{\text{unstd}} = d_i \odot \sigma$. The class-direction for feature i is

$$w_i \;=\; W_{\text{out}}\, d_i^{\text{unstd}} \in \mathbb{R}^K.$$

This is our *logit-lens* for CNN features (static, no training), aligning each feature to class logits.

3.4 Closed-form causal knockout Δ^-

Let $z_i(x)$ denote dense ReLU activation (no Top-K mask) of feature i aggregated over spatial tokens (e.g., max or mean). For class c, the (approximate) logit change from ablating feature i is

$$\Delta_i^-(c) \;\approx\; -\,\overline{z}_i \;\cdot\; w_i[c],$$

where $\overline{z}_i$ is the mean (over images where the feature is "on") of z_i, and $w_i[c]$ comes from the logit-lens projection above. This holds exactly for linear heads and serves as a cheap, forward-free causal proxy in practice.

3.5 Per-class association via PMI

Define indicator $I_i(x) = \mathbf{1}[z_i(x) > \tau_i]$ where τ_i is a high quantile (e.g., $q = 0.99$). With Laplace smoothing λ, let $P(c)$ be base class frequency and $P(c \mid I_i{=}1)$ the conditional class distribution. The pointwise mutual information:

$$\mathrm{PMI}_i(c) \;=\; \log \frac{P(c \mid I_i = 1)}{P(c)}.$$

We z-score $\mathrm{PMI}_i(c)$, $\Delta_i^-(c)$, and $w_i[c]$ across classes and combine them into a ranking score

$$S_i(c) \;=\; z\text{-}\mathrm{PMI}_i(c) \;+\; 2\, z(\Delta_i^-(c)) \;+\; z(w_i[c]).$$

We export top classes per feature using $S_i(c)$.

3.6 From numbers to words, then to *why*

For each feature i, we assemble: sparsity, threshold, top classes $\{(c, S_i(c), \mathrm{PMI}_i(c), \Delta_i^-(c), w_i[c])\}$, and the top-activating thumbnails; this JSONL is fed to a small LLM which returns a short concept label (e.g., "open circle", "zig-zag edge"). At inference, we find the input's top-activating features, retrieve their concepts, and emit a one-sentence rationale summarizing the decisive cues for the predicted class. (See our scripts for LLM prompt and the runtime explainer.)

4 Theory and Discussion

Faithfulness under a linear head. If the classifier head is linear and the pooled feature depends approximately linearly on token activations near the observed input, then the class logit can be locally decomposed along decoder directions. The closed-form $\Delta_i^-(c)$ measures the contribution of feature i to class c in expectation over inputs where the feature is present, without additional forward passes. This gives a *faithful* ranking among features for each class.

Instance conditioning. We threshold and select features per input using $z_i(x)$ (max/mean over tokens), which provides *local*, instance-specific concepts, unlike global unit-concept maps. The final one-sentence rationale is a deterministic, LLM-*free* composition of selected feature concepts (the LLM is used only once offline to name features).

Comparison to prior art. Grad-CAM highlights regions but not named features [1]. Network Dissection needs a labeled concept bank [2]. TCAV needs user concept sets and directional derivatives [3]. Our system learns features *from the model*, aligns them to classes via the head (logit-lens), quantifies evidence causally (Δ^-), and names them succinctly via an LLM. In spirit we echo dictionary-learning/SAE work in LLMs [6, 7, 8], but we target CNNs and per-instance explanations.

5 Algorithms

Algorithm 1: Train token-space Top-K SAE on frozen features

Algorithm 1 Top-K SAE training (token space)

Inputs: Frozen backbone; dataset; per-channel (μ, σ); SAE width M, density K/M
for epoch = 1, …, E **do**
  for mini-batch images x **do**
    $A \leftarrow \text{backbone}(x) \in \mathbb{R}^{B\times C\times H\times W}$, $a \leftarrow \text{reshape}(A) \in \mathbb{R}^{(BHW)\times C}$
    $\tilde{a} \leftarrow (a-\mu) \oslash \sigma$
    $s \leftarrow W_e\tilde{a} + b_e, \;\; r \leftarrow \mathrm{ReLU}(s)$
    $z \leftarrow \mathrm{TopK}(r; K)$ ▷ keep top-K per token
    $\hat{a} \leftarrow Dz + b_d$; $\mathcal{L} \leftarrow \|\hat{a} - \tilde{a}\|_2^2$
    Update W_e, b_e, D, b_d by AdamW; renormalize columns of D
    Update usage-EMA; if dead features: resample atoms from random tokens
  end for
end for

Algorithm 2: Summarize features and build neuron JSONL

Algorithm 2 Per-feature statistics & export

Inputs: Subset of images (stratified); dense code $z = \mathrm{ReLU}(W_e\tilde{a} + b_e)$
for image x **do**
    compute $z_i(x)$ aggregated over tokens (max/mean)
end for
For each i: threshold τ_i at q-quantile; indicator $I_i(x) = \mathbf{1}[z_i(x) > \tau_i]$
Compute class counts $P(c)$ and $P(c \mid I_i{=}1)$; $\mathrm{PMI}_i(c)$
Compute $w_i = W_{\text{out}}(d_i \odot \sigma)$ (logit-lens)
Compute closed-form $\Delta_i^-(c) \approx -\overline{z}_i\, w_i[c]$ over images where $I_i(x) = 1$
Combine z-scores into $S_i(c)$; rank top classes
Save top thumbnails and JSON object for neuron i

Algorithm 3: Offline LLM concept labeling

Algorithm 3 Neuron $\rightarrow$ short concept

for neuron JSON with top classes, Δ^-, thumbnails **do**
    Build a compact prompt; call a small LLM
    Parse a one-line concept ("SHORT ANSWER")
    Store mapping neuron_id $\mapsto$ concept
end for

Algorithm 4: At-inference "Why did you choose that class?"

Algorithm 4 Instance-conditioned rationale

Given input image x, compute feature map, standardized tokens, dense z
Select top-N features by $z_i(x)$ or by percentile
Retrieve concept strings for these features; read head logits for top-5 classes
Emit a one-sentence rationale summarizing decisive concepts for the top class

6 Implementation Notes

Classifier training. We use a ResNet-18 variant for 1-channel sketches (conv1 adapted, no initial max-pool), data I/O from parquet, AMP, cosine LR, grad-accum, DDP hooks, and probe logging of late features.

SAE training. Token-space Top-K SAE (target density ≈ 0.5–2%), unit-norm decoder, usage-EMA, dead-latent resampling, periodic decoder renorm; feature whitening via per-channel mean/std on the frozen feature map.

Summaries & thumbnails. We cache dense codes, compute per-feature thresholds, PMI, logit-lens w_i, closed-form Δ^-, export JSON and sprites/thumbs for top activations.

Neuron $\rightarrow$ concept (LLM). A small LLM receives numeric stats and thumbnails and returns a one-line concept; we aggregate into a `llm_output.jsonl`.

Why-runner. For an input image, we compute the top features, read concept strings, print top-5 class confidences, and a one-sentence rationale.

7 Evaluation Protocols (Suggested)

- **Faithfulness**: Deletion AUC / Insertion AUC over features: iteratively suppress features with highest $z_i(x)$ and measure logit changes versus Δ^- predictions.
- **Consistency**: Across data augmentations (jitter/crop), check stability of selected concepts and rationale text.
- **Causality sanity checks**: For a class c, features with high negative $\Delta_i^-(c)$ should reduce the c-logit when ablated; control features should not.
- **Human utility**: Rate succinctness and correctness of concept labels against thumbnails; optional TCAV-style human concepts for comparison.

8 Limitations and Ethics

Our Δ^- assumes a linear head and is local; non-linear effects upstream are not captured exactly. LLM-generated concept names, although constrained by numeric evidence, may reflect biases in prompts or data. Explanations describe *how* the model decided, not whether it *should*; users must avoid over-trusting rationales for high-stakes domains.

9 Conclusion

We show that a frozen CNN can be equipped with sparse, class-aligned features and a compact LLM vocabulary to produce short, instance-conditioned, and causally grounded rationales. The method is simple, computationally light, and compatible with standard training. We hope this spurs broader adoption of post-hoc feature dictionaries for faithful explanations.

Code availability. Training the classifier, training the SAE, summarizing neurons, LLM conversion, and the "why" runner are provided as five scripts.

References

[1] Ramprasaath R. Selvaraju et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. In *ICCV*, 2017.

[2] David Bau et al. Network Dissection: Quantifying Interpretability of Deep Visual Representations. In *CVPR*, 2017.

[3] Been Kim et al. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *ICML*, 2018.

[4] Nostalgebraist. Interpreting GPT: the Logit Lens (blog), 2020.

[5] Nora Belrose et al. Eliciting Latent Predictions from Transformers with the Tuned Lens. *arXiv:2303.08112*, 2023.

[6] Matthew Bricken et al. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. 2023.

[7] Anthropic Interpretability Team. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. 2024.

[8] Hoagy Cunningham et al. Sparse Autoencoders Find Highly Interpretable Features in Language Models. *arXiv:2309.08600*, 2023.

[9] Michael Lan et al. Sparse Autoencoders Reveal Universal Feature Spaces Across Large Language Models. *arXiv:2410.06981*, 2024.

[10] Robert Huben et al. Efficient Dictionary Learning with Switch Sparse Autoencoders. *arXiv:2410.08201*, 2024.