

## 1 A Additional Related Work

2 To provide further context for our work, we additionally review related studies in the areas of input  
3 attribution methods and mechanistic interpretability in vision transformers.

4 **Input Attribution Methods** Input attribution methods aim to identify the most important input  
5 features for a model’s prediction [1–34]. These methods are used to understand model behavior and  
6 detect potential biases. Among various input attribution methods, gradient-based methods, such as  
7 Integrated Gradients [13] and Layer-wise Relevance Propagation (LRP) [19–21], are widely used for  
8 their scalability and efficiency. In addition, since attention mechanisms in transformer-based models  
9 can offer interpretable features, attention-based methods have also been proposed [30, 31, 35–37].  
10 Input attribution techniques have been extended to interpret internal components of the model, such  
11 as neurons or attention heads [38–47]. For example, attribution patching estimates the causal effect  
12 of a component by replacing its activation with a baseline value using gradients [43]. Recent works  
13 have proposed combining Integrated Gradients with attribution patching to improve the quality of  
14 estimation [46, 47]. In this work, we extend LibraGrad [18], a theoretically grounded, gradient-  
15 based input attribution method, to estimate the importance of each SAE feature, which significantly  
16 improves the faithfulness of the interpretation.

17 **Mechanistic Interpretability in Vision Transformers** Mechanistic interpretability (MI) is an  
18 emerging research field dedicated to interpreting the inner workings of neural networks by reverse-  
19 engineering their computations into human-interpretable mechanisms [48–53]. In particular, to  
20 understand the mechanisms of vision transformers (ViTs), early studies employed representation  
21 analysis, linear probing, and the logit lens to examine the representations learned by ViTs [54–56].  
22 More recent works have also sought to interpret neurons and attention heads in ViTs using natural  
23 language descriptions [57–59]. In addition, with the rise of sparse autoencoders (SAEs), there has  
24 been growing interest in interpreting ViT features through the lens of SAEs [60–67]. While the  
25 aforementioned approaches explain how individual components of ViTs function, they do not offer  
26 a comprehensive, end-to-end understanding of the model from input to output. To address this,  
27 we propose the residual replacement model, which characterizes the entire process by which ViTs  
28 transform input images into predictions.

## 29 B Discussion

30 **Towards Interpreting Vision Models** We find that interpreting vision models is more challenging  
31 than interpreting language models (LMs) for several reasons.

32 First, the features learned by vision models are often difficult to describe in natural language, making  
33 it challenging to provide intuitive explanations. Therefore, we alternatively describe the features  
34 using maximally activating images or patches. For certain types of features, such as curve detectors  
35 and position detectors, we additionally provide human-interpretable patterns to aid understanding.  
36 Nonetheless, it remains difficult to provide concise natural language descriptions for some features,  
37 whereas maximally activating images or patches often reveal clearer patterns. Developing more  
38 effective interpretation strategies, particularly those that leverage visual demonstrations, remains an  
39 important direction for future work.

40 Second, the number of patch tokens in vision models is significantly larger than the number of  
41 tokens typically used in mechanistic interpretability studies of language models [41, 44, 68–71].  
42 This makes it challenging to scalably identify circuits in vision models, especially when considering  
43 token-to-token interactions. To address this, we aggregate patch tokens, enabling the interpretation of  
44 spatially independent feature interactions. While many token-to-token interactions may be redundant  
45 due to the semantic similarity between neighboring patches, some interactions can still play important  
46 roles. For instance, understanding how the [CLS] token interacts with other tokens could provide  
47 valuable insights into the model’s behavior [55]. We leave the investigation of such interactions to  
48 future work.

49 Finally, unlike mechanistic interpretability studies in language models where tasks are typically  
50 predefined (e.g., indirect object identification [68], greater-than [69], subject-verb agreement [46]),  
51 vision models often lack clearly defined tasks. Although one can average circuits across images with  
52 the same class label, such averaging does not guarantee meaningful or interpretable circuits, as even  
53 images from the same class may rely on different pathways for prediction. An interesting direction  
54 for future work is to develop methods for identifying meaningful circuits corresponding to implicitly  
55 defined tasks in vision models.

56 **Comparison with Related Work** Olah et al. [48] attempted to reverse-engineer the computations  
57 of CNNs by analyzing the weights of InceptionV1 and identifying how features in early layers  
58 compositionally combine to form higher-level features in deeper layers, resulting in what they called  
59 “circuits.” For example, they identified curve detector features that are equivariant to transformations,  
60 demonstrating how different features, such as various angles of lines in earlier layers, combine to  
61 form a curve detector [72–75]. Our residual replacement model enables similar analysis of features  
62 and circuits in ViTs by observing the model’s residual stream. Furthermore, we find that similar  
63 features and circuits emerge in ViTs, despite their distinct architectures. From the perspective of  
64 the universality hypothesis [48, 76, 77], this work suggests that CNNs and ViTs may share similar  
65 mechanisms for processing visual information, although the representations in the early layers of  
66 ViTs tend to be more patch-wise compared to those in CNNs.

67 To mechanistically interpret large language models (LLMs), Marks et al. [46] proposed a method for  
68 identifying sparse feature circuits, where the nodes correspond to SAE features. They demonstrated  
69 how LLMs achieve subject-verb agreement across relative clauses by identifying the specific circuits  
70 responsible for this behavior. Concurrently, Ameisen et al. [78] proposed constructing attribution  
71 graphs within a replacement model equipped with a cross-layer transcoder (CLT) [79–81]. Leveraging  
72 the replacement model, they identified various circuits in LLMs, including those responsible for  
73 multi-step reasoning and arithmetic [82]. Inspired by these works, we aim to construct similarly  
74 interpretable sparse circuits in ViTs. However, unlike LLMs, ViTs operate on a much larger number  
75 of tokens, making it infeasible to identify circuits at scale using the same methodology. Therefore,  
76 we simplify the process by focusing on the residual stream of ViTs [83], which enables us to identify  
77 circuits in a more parsimonious and scalable manner.



## 78 C Training Sparse Autoencoders

### 79 C.1 Training

80 **Formulation** We apply a TopK SAE [84] to each layer of the residual stream in ViTs. As shown in  
 81 Equation (1), the TopK SAE maps a residual stream representation  $\mathbf{x} \in \mathbb{R}^d$  to a sparse representation  
 82  $\mathbf{z} \in \mathbb{R}^f$  by selecting the top- $k$  features from  $\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b}_{\text{pre}})$ , where  $f$  denotes the number of sparse  
 83 features and  $k$  is the sparsity level (i.e., the number of selected features). Following Gao et al. [84], we  
 84 train the TopK SAE with a combination of two loss functions: a reconstruction loss and an auxiliary  
 85 loss. The reconstruction loss  $\mathcal{L}_{\text{recon}}$  measures the discrepancy between the original input  $\mathbf{x}$  and its  
 86 reconstruction  $\hat{\mathbf{x}}$  produced by the decoder:

$$\mathcal{L}_{\text{recon}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2. \quad (1)$$

87 For normalized evaluation, we report the Fraction of Variance Unexplained (FVU), defined as:

$$\text{FVU} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}{\|\mathbf{x}\|_2^2}, \quad (2)$$

88 which allows comparison across layers and models regardless of the input scale [85, 86]. The  
 89 auxiliary loss  $\mathcal{L}_{\text{aux}}$  encourages the SAE to represent residual information using dead (inactive) features.  
 90 Specifically, we define the reconstruction error as  $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$  and estimate it via  $\hat{\epsilon} = \mathbf{W}_{\text{dec}} \mathbf{z}_{\text{dead}}$ ,  
 91 where  $\mathbf{z}_{\text{dead}} = \text{TopK}_{\text{aux}}(\text{ReLU}(\text{Dead}(\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b}_{\text{pre}}))))$ . The auxiliary loss is then defined as:

$$\mathcal{L}_{\text{aux}} = \|\epsilon - \hat{\epsilon}\|_2^2. \quad (3)$$

92 The total loss is given by:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \alpha \mathcal{L}_{\text{aux}}, \quad (4)$$

93 where  $\alpha$  is a hyperparameter controlling the weight of the auxiliary loss. While the reconstruction loss  
 94 ensures faithful input reconstruction, the auxiliary loss discourages feature inactivity (dead feature)  
 95 and promotes more effective utilization of the sparse feature set.

96 **Implementation Details** We train the TopK SAE using the Adam optimizer with a learning rate  
 97 of  $2 \times 10^{-4}$ ,  $\beta_1 = 0.9$ , and  $\beta_2 = 0.999$ . The model is trained for 50 epochs with a batch size of  
 98 512. Hyperparameters are set as follows:  $\alpha = 1/32$ ,  $k_{\text{aux}} = 256$ . Decoder normalization is applied at  
 99 each training step, and the input  $\mathbf{x}$  is normalized to have zero mean and unit variance. We conduct a  
 100 hyperparameter sweep over the sparsity level  $k$  and the expansion rate  $R = f/d$  to investigate scaling  
 101 behavior. We use the following backbone models: supervised ViT-B/16 [87], DINOv2 ViT-B/14 (with  
 102 register tokens) [56, 88], and CLIP ViT-B/16 [89]. Training and evaluation are performed on the  
 103 ImageNet-1K [90] training/validation dataset, respectively. For each model, we collect all tokens  
 104 ([CLS], patch tokens, and register tokens (if present)) from each layer to train the TopK SAE.

### 105 C.2 Analysis

106 **Scaling Law** To understand the scaling behavior of the TopK SAE in ViTs, we extend the scaling  
 107 law analysis of Gao et al. [84] to vision transformers. Specifically, we fit a joint scaling law of the  
 108 reconstruction loss with respect to the number of latent features  $f$  and the sparsity level  $k$  using the  
 109 following functional form:

$$L(f, k) = \exp(\alpha + \beta_k \log(k) + \beta_f \log(f) + \gamma \log(k) \log(f)) + \exp(\zeta + \eta \log(k)), \quad (5)$$

110 where  $\alpha$ ,  $\beta_k$ ,  $\beta_f$ ,  $\gamma$ ,  $\zeta$ , and  $\eta$  are the parameters to be fitted. We find that this scaling law fits well  
 111 across all models and layers, explaining at least 99% of the variance. The fitted losses are visualized as  
 112 contour plots in Figures 1 to 3, where the x- and y-axes correspond to the expansion rate  $R = f/d$  and  
 113 the sparsity level  $k$ , respectively. We additionally mark the contour line corresponding to  $\text{FVU} = 0.15$ ,  
 114 which indicates the threshold below which the reconstruction loss is considered acceptable. Our  
 115 chosen hyperparameters are denoted with a cross. Moreover, Figure 4 shows the  $\text{FVU} = 0.15$  contour  
 116 line across layers. We observe that deeper layers generally require higher sparsity or more latent  
 117 features to reach the same reconstruction performance, suggesting that they encode more complex  
 118 information. An exception is found in the penultimate and final layers, which require fewer features,  
 119 possibly due to their proximity to the output and reduced representational complexity.

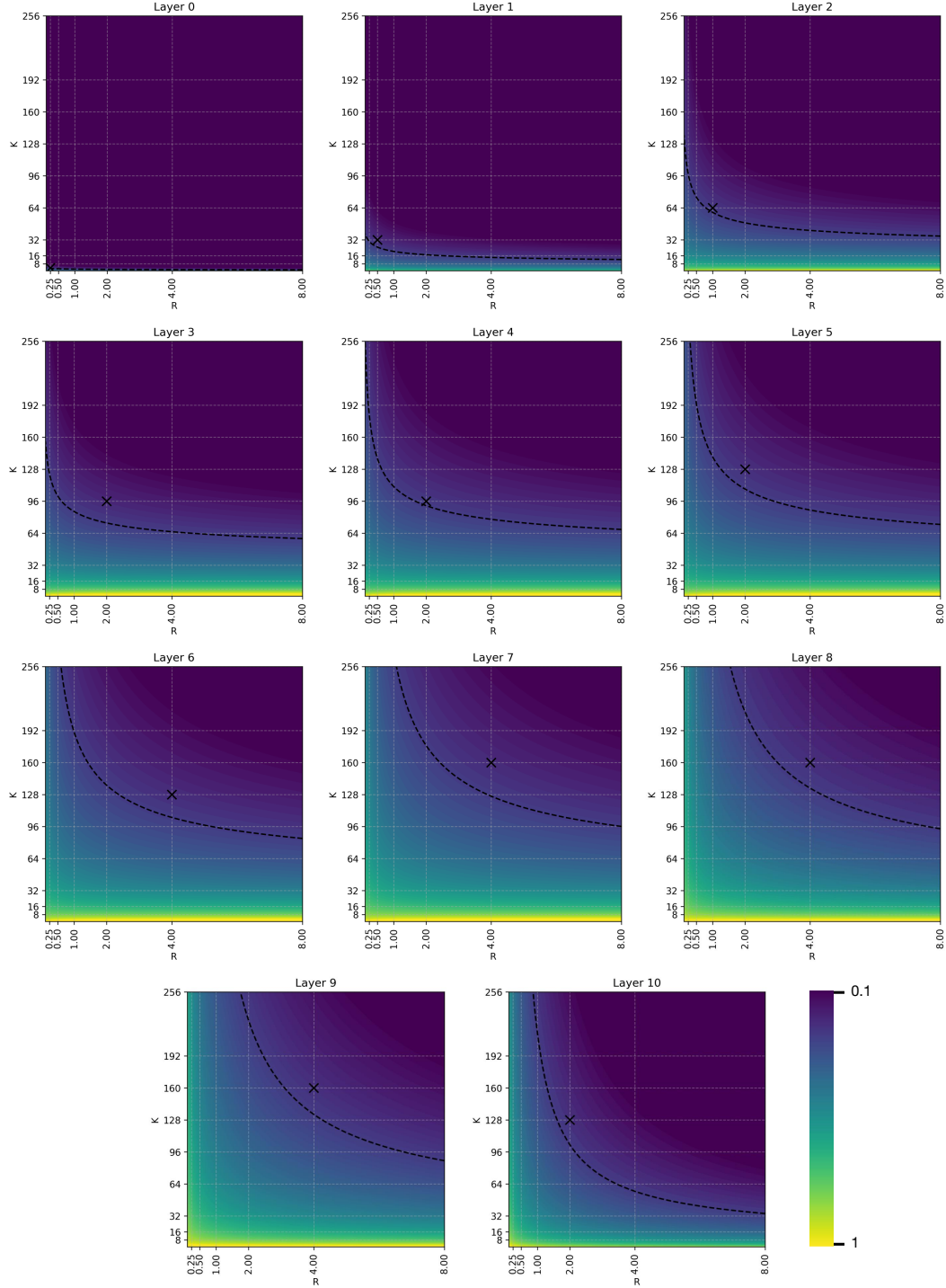


Figure 1: Contour plots of the fitted scaling law for reconstruction loss with respect to the expansion rate  $R = f/d$  and the sparsity level  $k$  (ViT). Dotted line indicates the contour line corresponding to  $FVU = 0.15$ . The cross indicates the chosen hyperparameters for the SAE.

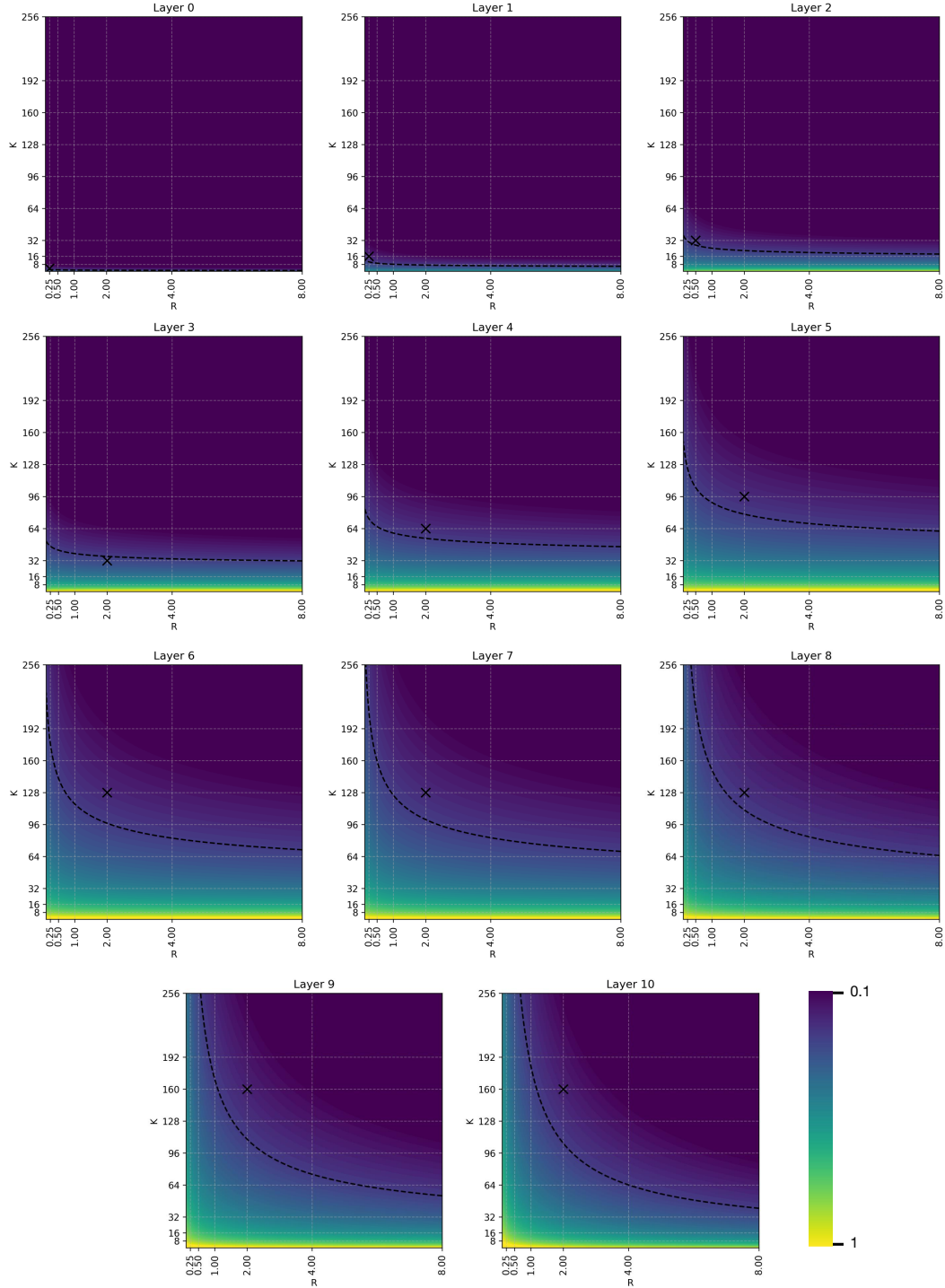


Figure 2: Contour plots of the fitted scaling law for reconstruction loss with respect to the expansion rate  $R = f/d$  and the sparsity level  $k$  (DINOv2). Dotted line indicates the contour line corresponding to  $FVU = 0.15$ . The cross indicates the chosen hyperparameters for the SAE.

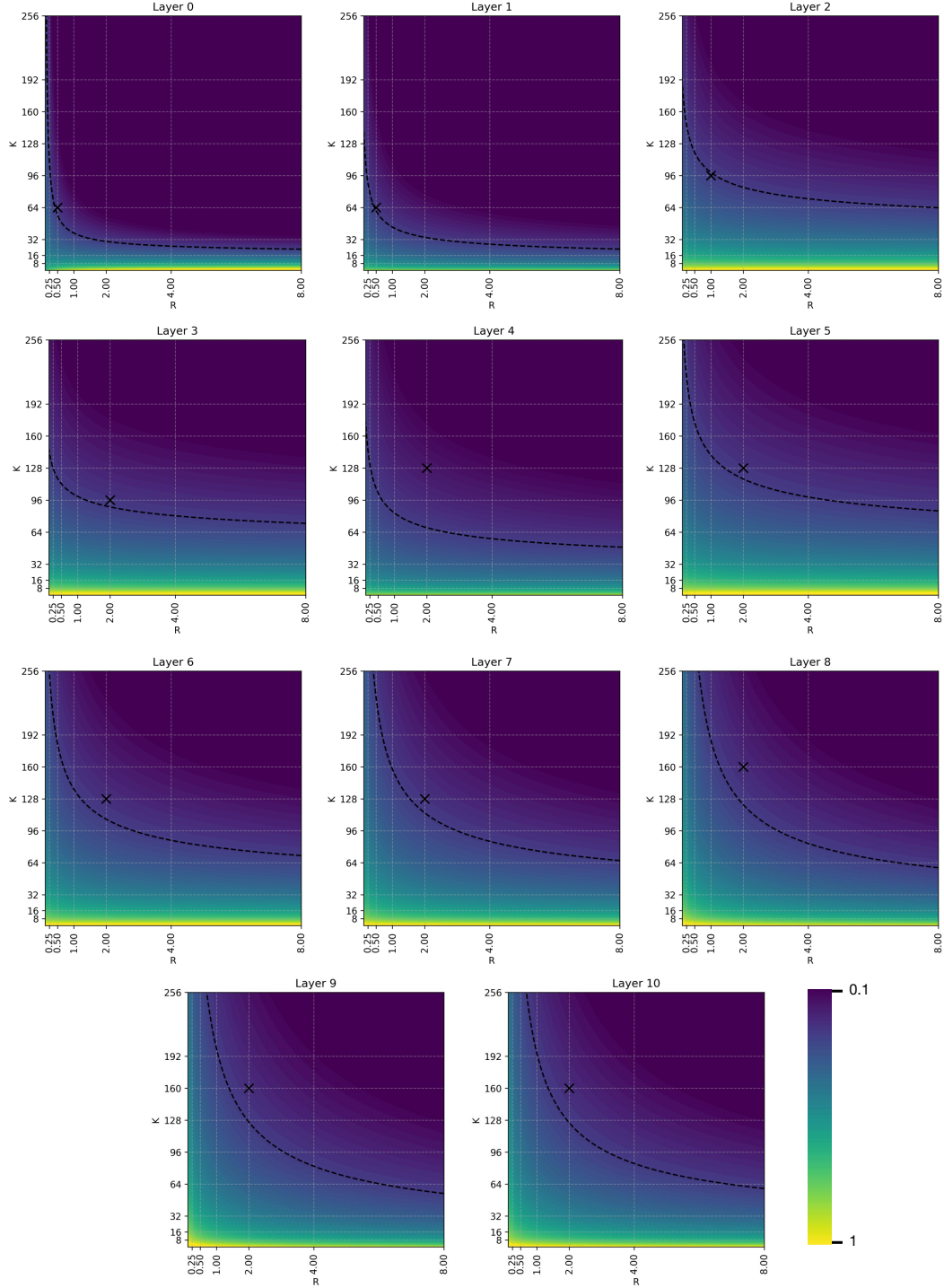


Figure 3: Contour plots of the fitted scaling law for reconstruction loss with respect to the expansion rate  $R = f/d$  and the sparsity level  $k$  (CLIP). Dotted line indicates the contour line corresponding to  $FVU = 0.15$ . The cross indicates the chosen hyperparameters for the SAE.

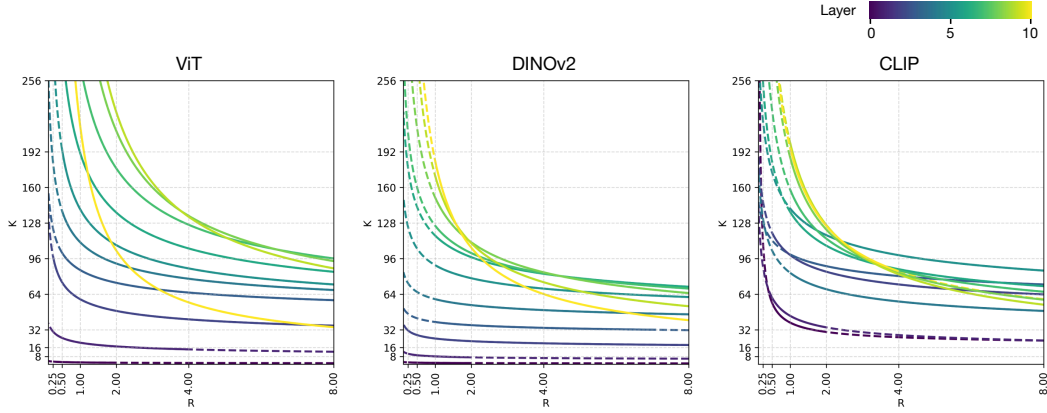


Figure 4: FVU = 0.15 contour line across layers. Dotted lines are extrapolated from the scaling law.

120 **Activation Analysis** To better understand the features learned by the TopK SAE, we analyze the  
 121 activation patterns of the features. For each latent feature, we compute its activation frequency (i.e.,  
 122 how often the feature is selected) and its average activation value [61]. As shown in Figures 5 to 7,  
 123 we find a strong positive correlation between activation frequency and average activation value:  
 124 features with higher values tend to be activated more frequently, indicating their greater importance.  
 125 Additionally, we observe that intermediate layers often exhibit a bimodal distribution in both activation  
 126 frequency and value. This suggests that they specialize into two groups of features: those that are  
 127 consistently informative and those that are only selectively used.

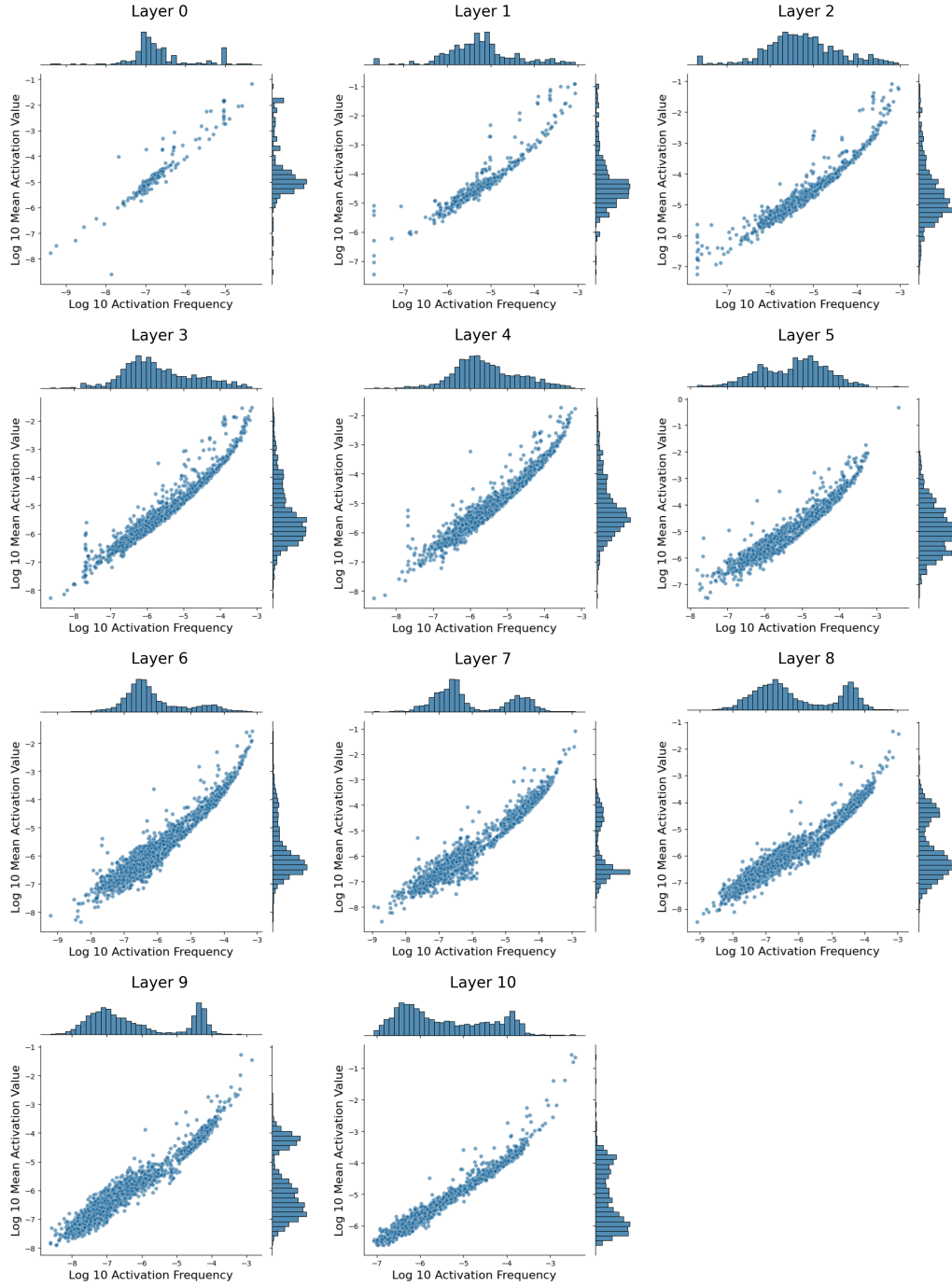


Figure 5: Activation frequency and mean activation value of the TopK SAE features (ViT).

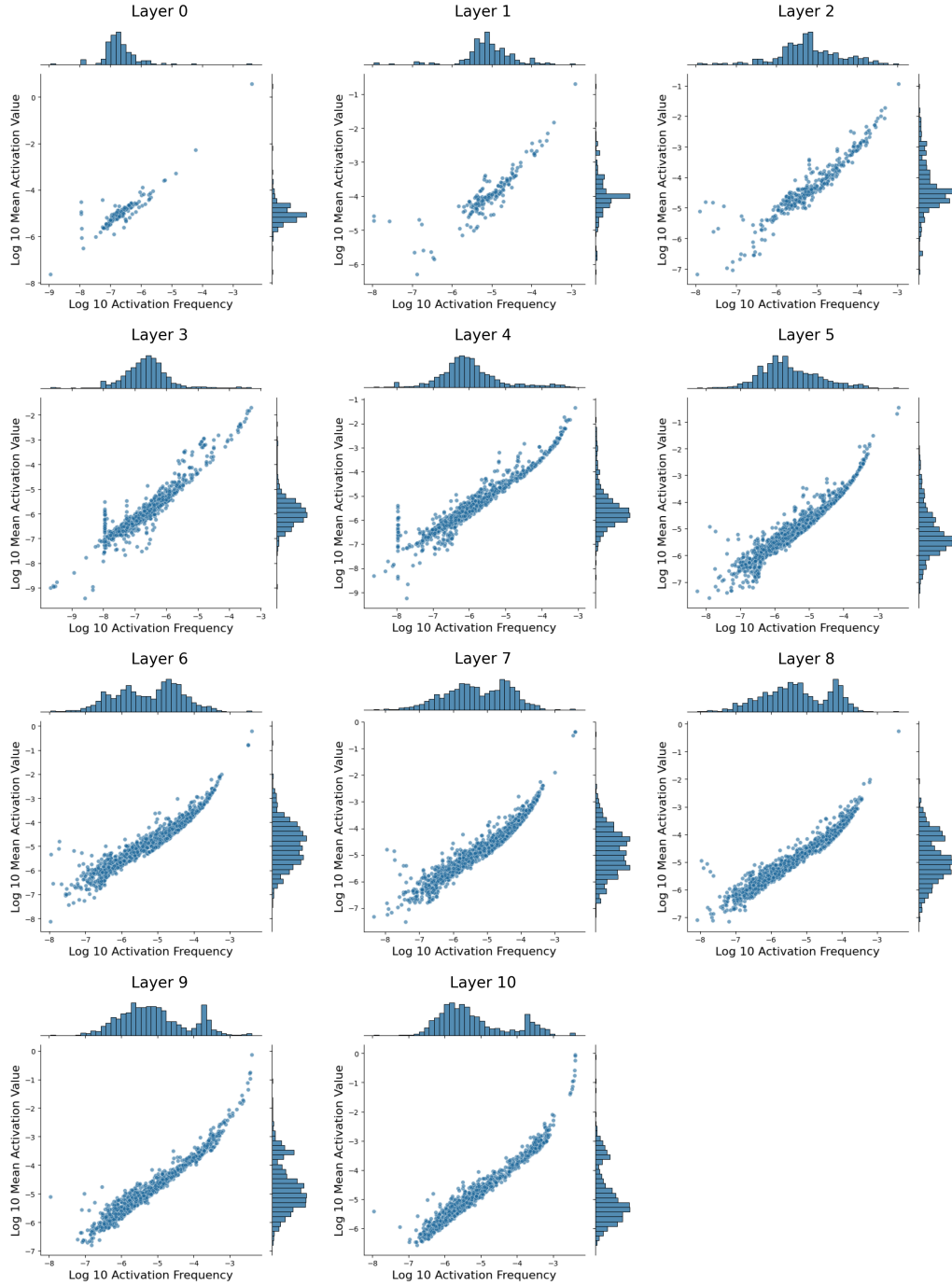


Figure 6: Activation frequency and mean activation value of the TopK SAE features (DINOv2).

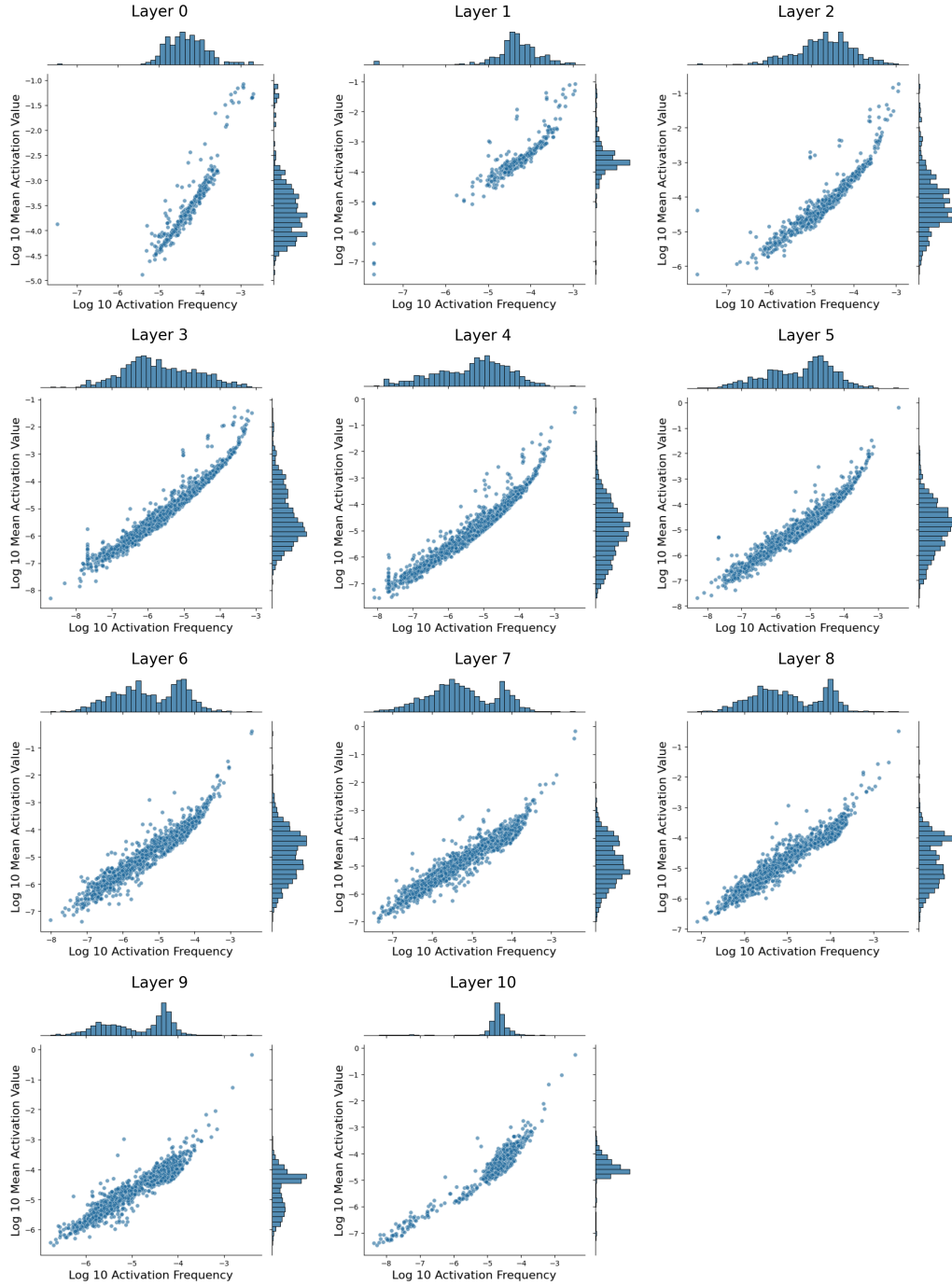


Figure 7: Activation frequency and mean activation value of the TopK SAE features (CLIP).



## D Systematic Feature Analysis

### D.1 Visualization

In vision models, visualizing maximally activated images is a common method for interpreting learned features<sup>1</sup> [60, 65–67, 96]. However, we observe that this approach is often insufficient for understanding the features learned by the TopK SAE, particularly in early layers. Early-layer features frequently activate in only a small number of patches (sometimes just a single patch) within an image, rendering the maximally activated image uninformative. To address this, we adopt a visualization strategy inspired by language models, where maximally activated tokens are commonly used to interpret features [46, 97, 98]. Specifically, we visualize the maximally activated patches for each feature, which provides a more localized and interpretable view. Additionally, we provide class label and logit lens interpretation, which are especially informative for late-layer features. Throughout the analysis in §2.2, we employ visualizations such as Figures 8 and 9 to illustrate the characteristics of the learned features.

### D.2 Categorization

**Definition of Categories** To systematically analyze the features learned by SAE, we categorize them into several groups based on their characteristics. Since feature categorization is inherently subjective, different researchers may interpret the same feature differently. To ensure consistency in our analysis, we define clear criteria for each category. These definitions are provided below.

- **Line:** In cases where a linear or curvilinear structure is captured within a patch.
- **Shape:** In cases where a distinct geometric shape (*e.g.*, circle or rectangle) is observed within a patch.
- **Color:** In cases where consistent coloration is observed, abrupt color transitions are detected, or other color-related features are present.
- **Texture:** In cases where a recurring texture or repetitive pattern is present.
- **Semantic:** In cases where a consistent high-level semantic concept is observed across images, even if not easily described by a single word.
- **Object:** In cases where a consistent, concrete, and identifiable object or entity is captured (*e.g.*, sky, ground, dog, dog’s nose).
- **Background:** In cases where the entire area of the image excluding the primary object is captured.
- **Positional:** In cases where a fixed spatial location within the image is consistently captured, independent of the image’s content.
- **Miscellaneous:** In cases where there is a clear visual commonality among patches, yet the pattern does not align well with any of the predefined categories above.
- **Polysemantic:** In cases where multiple distinct groups of patches or images each capture different semantic or visual attributes.
- **Uninterpretable:** In cases where no common or interpretable features can be identified at the patch or image level.

**More Results on Categorization** We provide additional results on the categorization of features learned by SAE in DINOv2 and CLIP. As shown in Figure 10, the features learned by DINOv2 and CLIP exhibit patterns similar to those observed in ViT (§2.2).

### D.3 Case Studies Details

**Curve Detectors** To generate *radial tuning curves* [73], we create synthetic images in which each patch contains a curve with varying angles and curvatures. For each curve detector, we compute the maximum activation of the feature across all patches in each image and then take the maximum across all images sharing the same angle. Finally, we plot the maximum activation as a function of the curve angle, as shown in Figure 3 of the main text.

<sup>1</sup>We also consider specialized feature visualization methods that optimize the input image to maximize the activation of a specific feature [91–93]. However, this approach has been shown to be less effective than using maximally activated images and tends to produce less interpretable results in ViTs [94, 95].

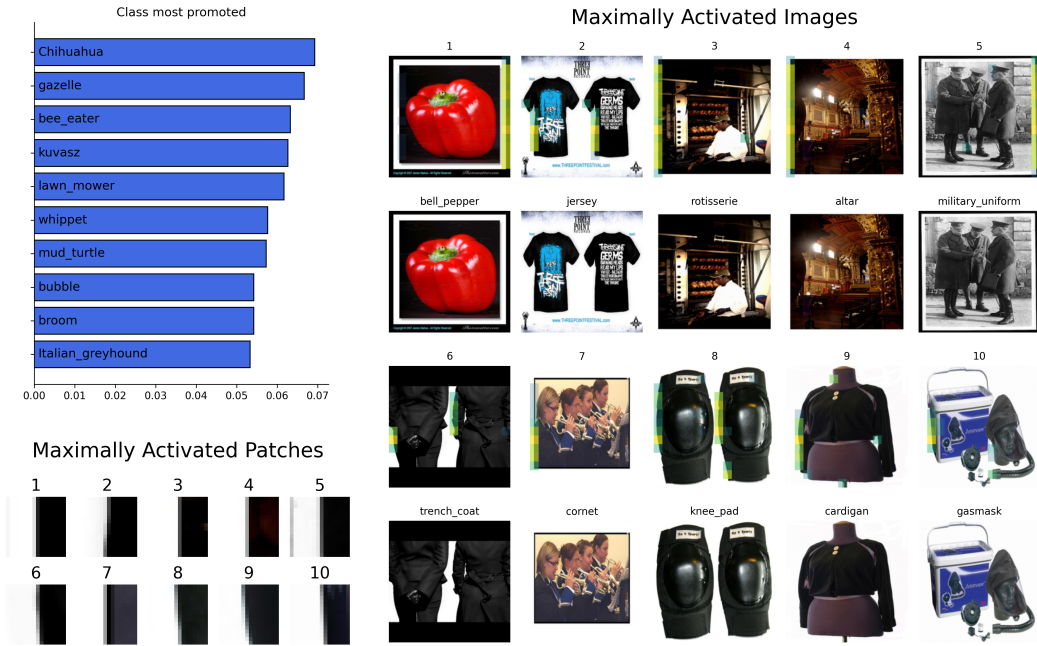


Figure 8: Visualization Example (1).

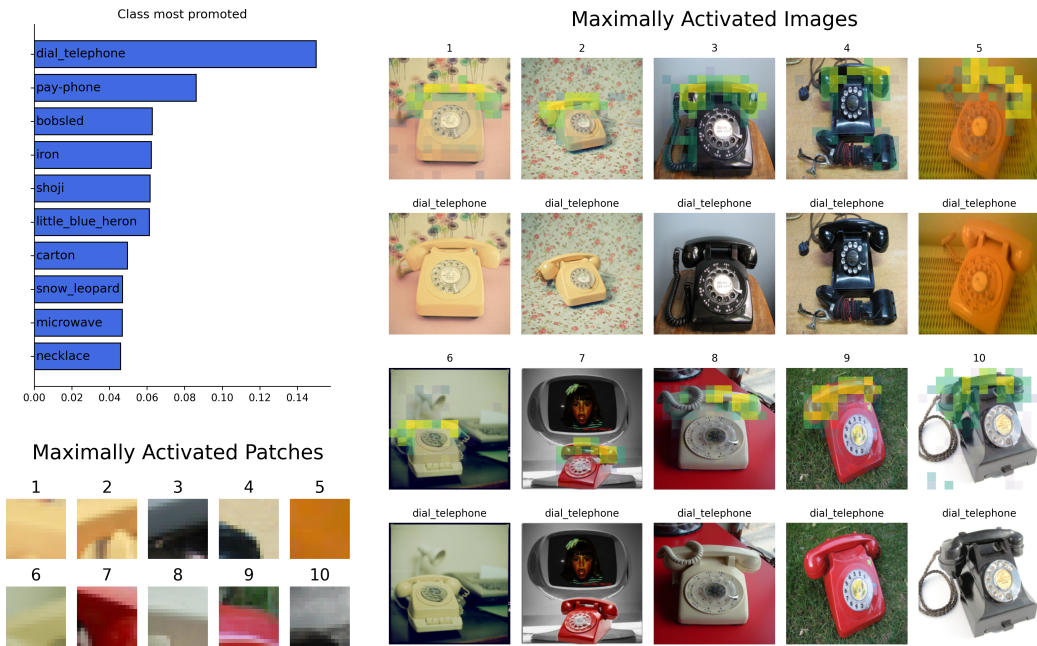


Figure 9: Visualization Example (2).

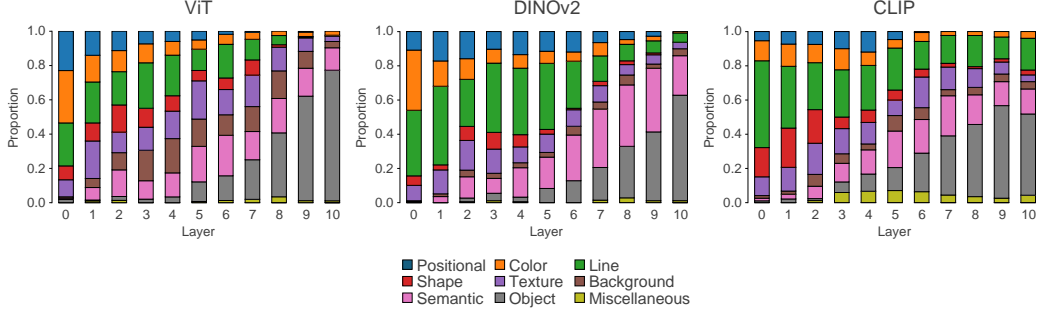


Figure 10: Category Proportions for ViT, DINOv2, and CLIP.

**Position Detectors** To automatically identify position detectors [99], we compute the mutual information between two variables: the activation of a feature and the position of a patch within the image. The mutual information is defined as:

$$I(act, pos) = \frac{1}{T} \cdot \sum_{pos=1}^T \left[ f_r^{(pos)} \cdot \log \frac{f_r^{(pos)}}{f_r} + \left(1 - f_r^{(pos)}\right) \cdot \log \frac{1 - f_r^{(pos)}}{1 - f_r} \right], \quad (6)$$

where  $f_r^{(pos)}$  denotes the activation frequency of feature  $n$  at position  $pos$ ,  $f_r$  is the overall activation frequency of feature  $n$ , and  $T$  is the total number of patches in the image. We identify position-sensitive features by selecting those with mutual information exceeding a threshold, i.e.,  $I(act, pos) > \tau$ . For the analysis in the main text, we use  $\tau = 0.05$ , although a range of reasonable thresholds yields similar results due to the clear separation between position detectors and other features. To visualize the position detectors, we plot the average activation of each feature across spatial positions, as shown in Figure 4 of the main text.

#### D.4 More Feature Examples

We present additional examples of features learned by the TopK SAE in Figures 11 to 22. These examples include various types of features, such as ripple patterns, V shapes, horizontal lines, hands, and watermarks. We optionally visualize the maximally activated patches when doing so aids in interpreting the feature.

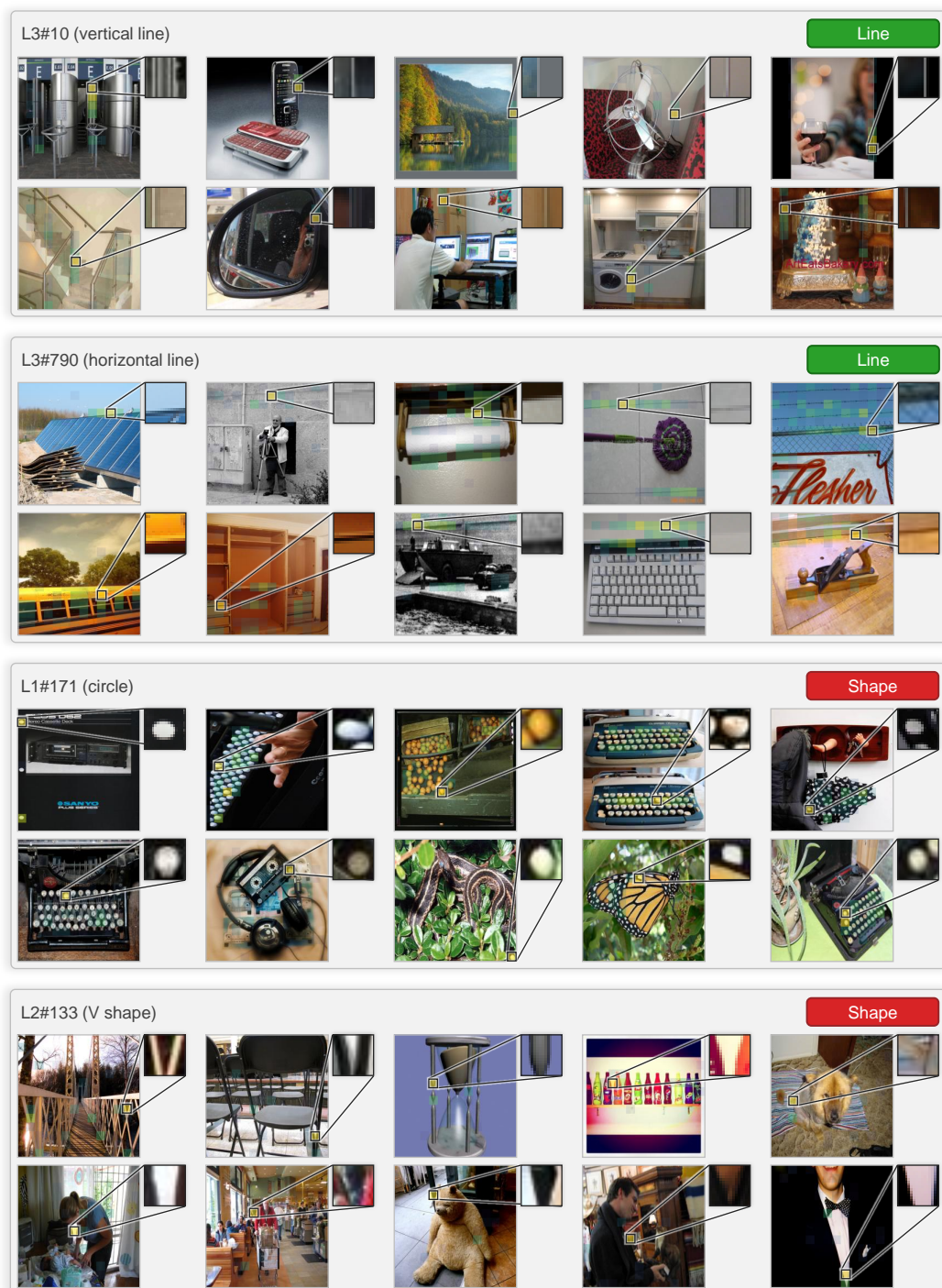


Figure 11: More ViT Feature Examples (Line and Shape).



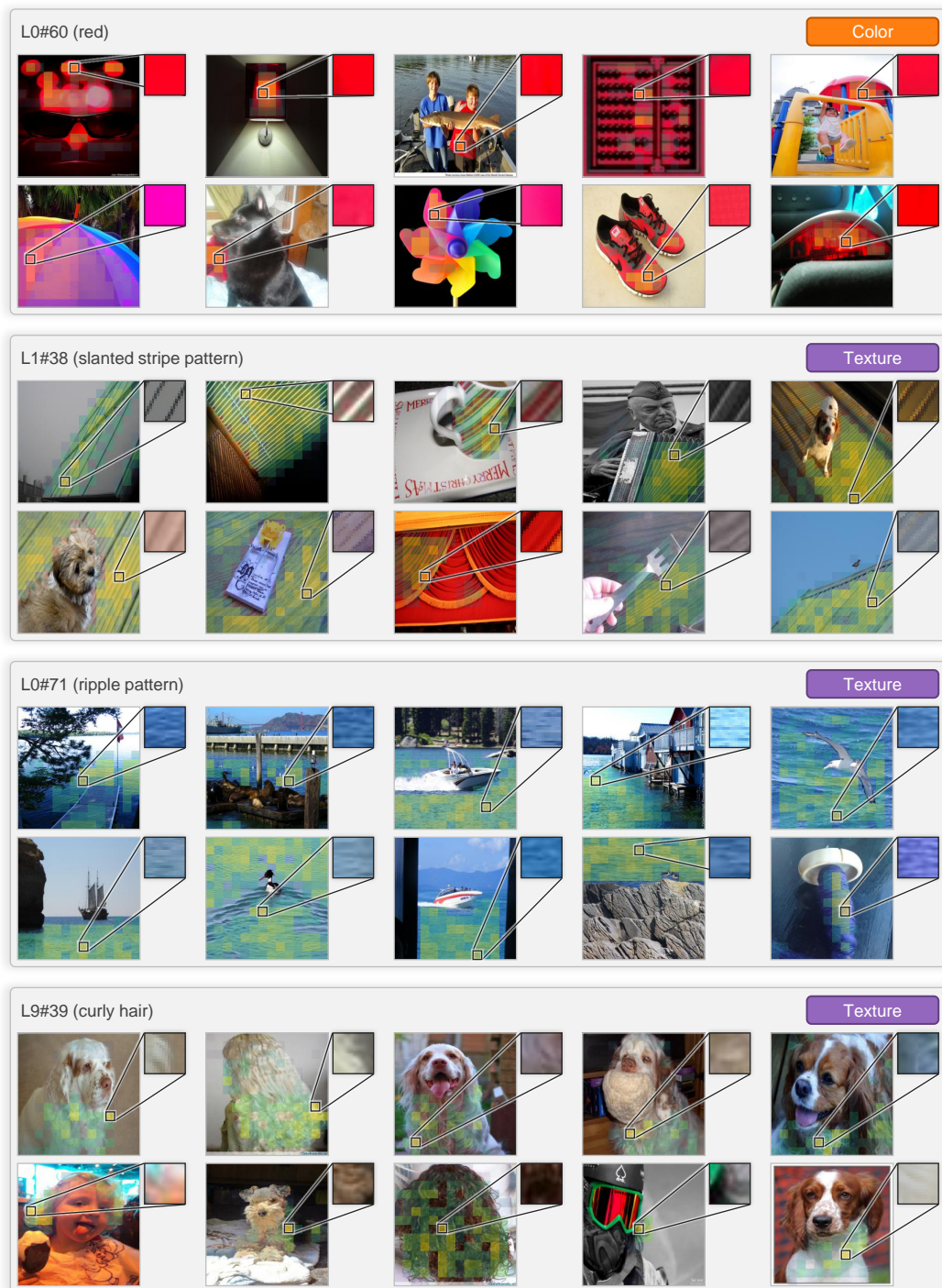


Figure 12: More ViT Feature Examples (Color and Texture).

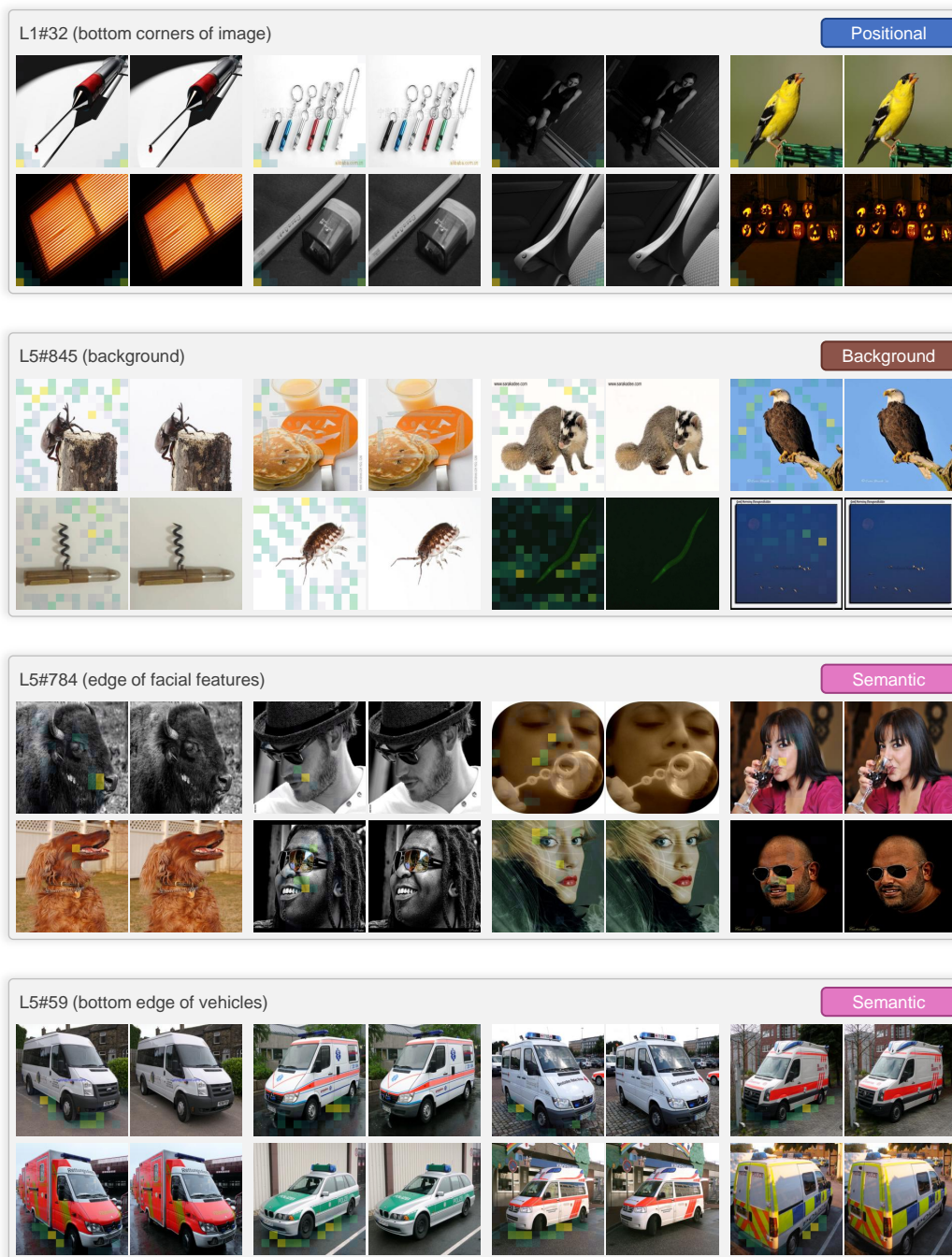


Figure 13: More ViT Feature Examples (Positional, Background, and Semantic).

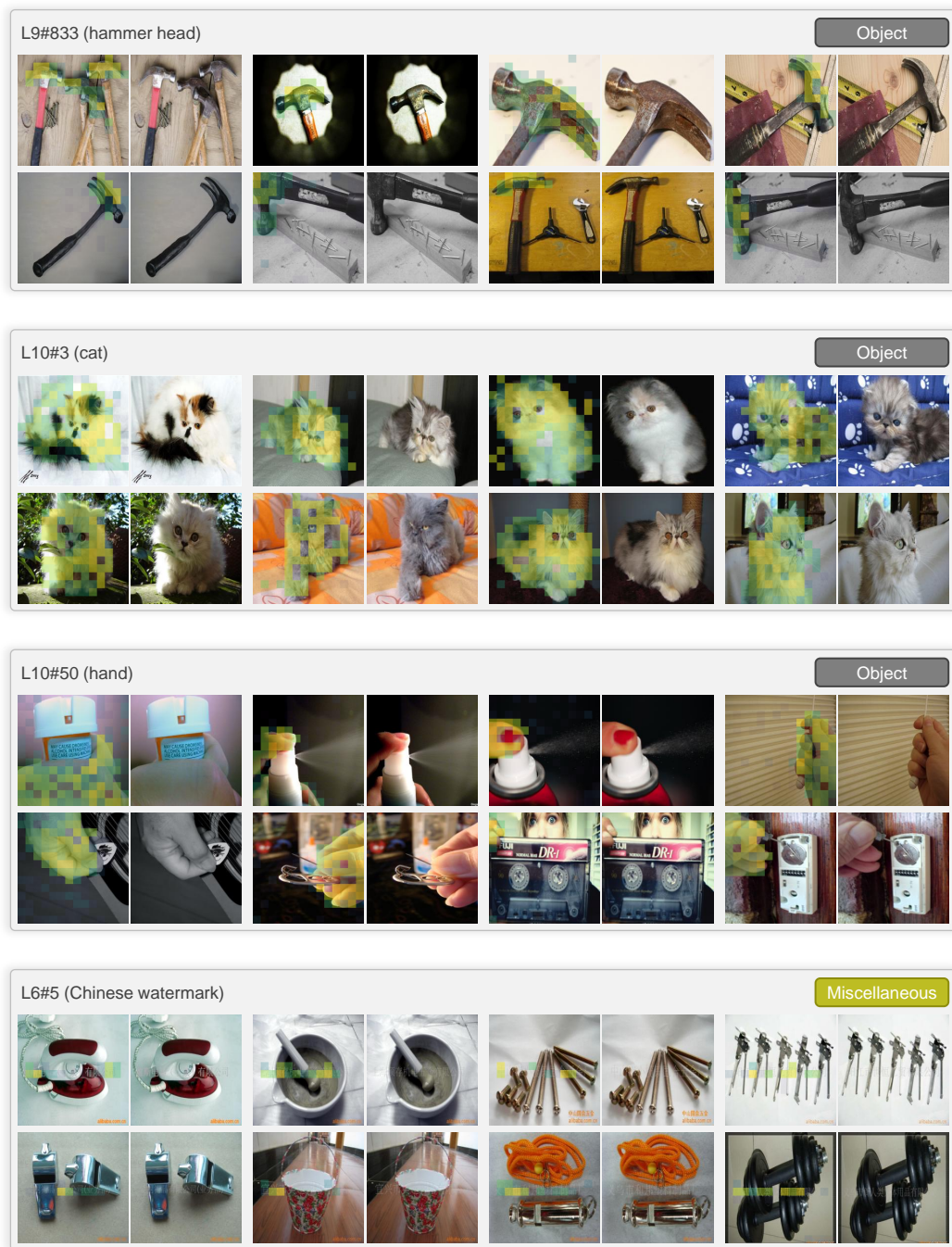


Figure 14: More ViT Feature Examples (Object and Miscellaneous).



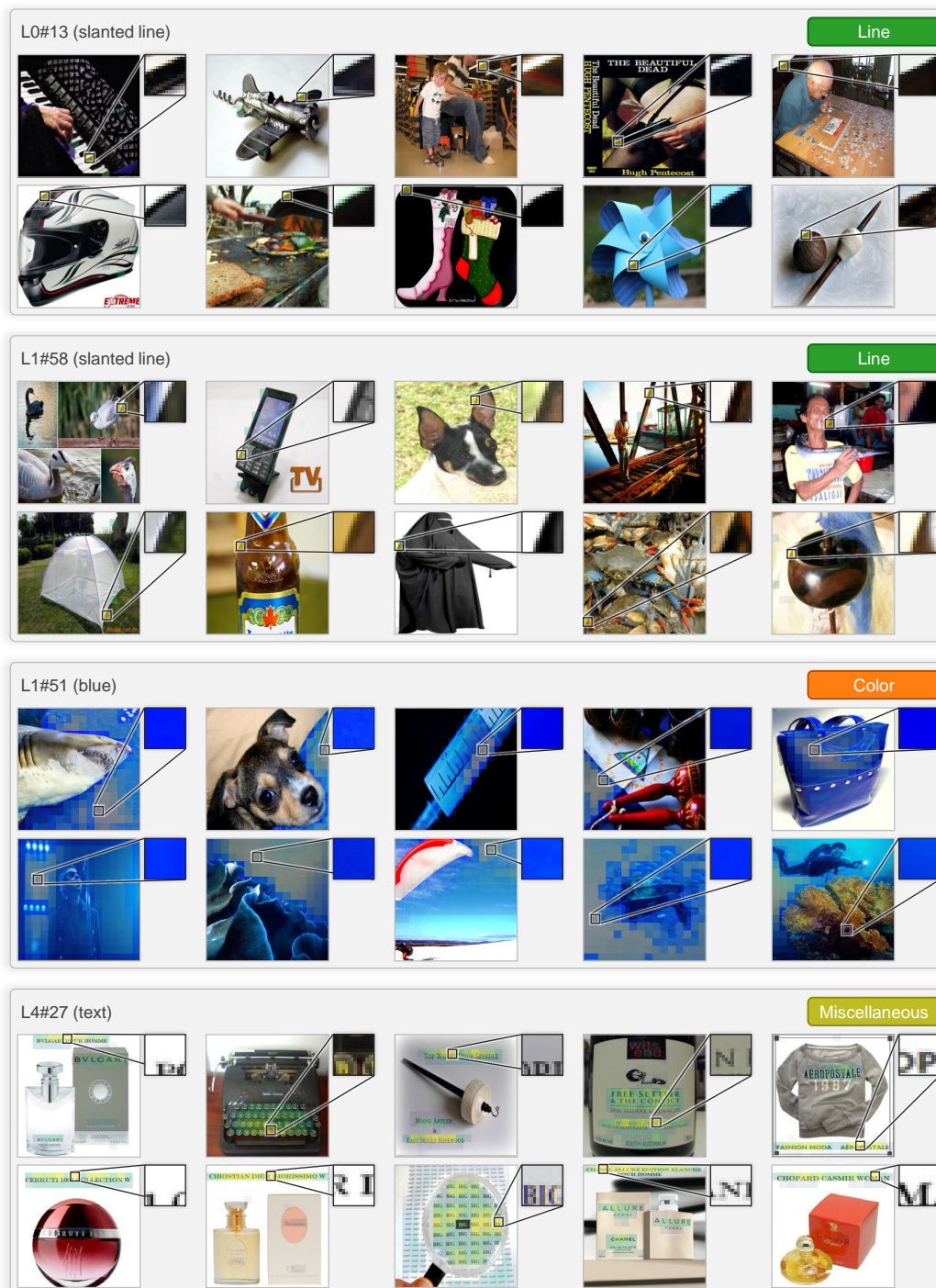


Figure 15: More DINOv2 Feature Examples (Line, Color, and Miscellaneous).



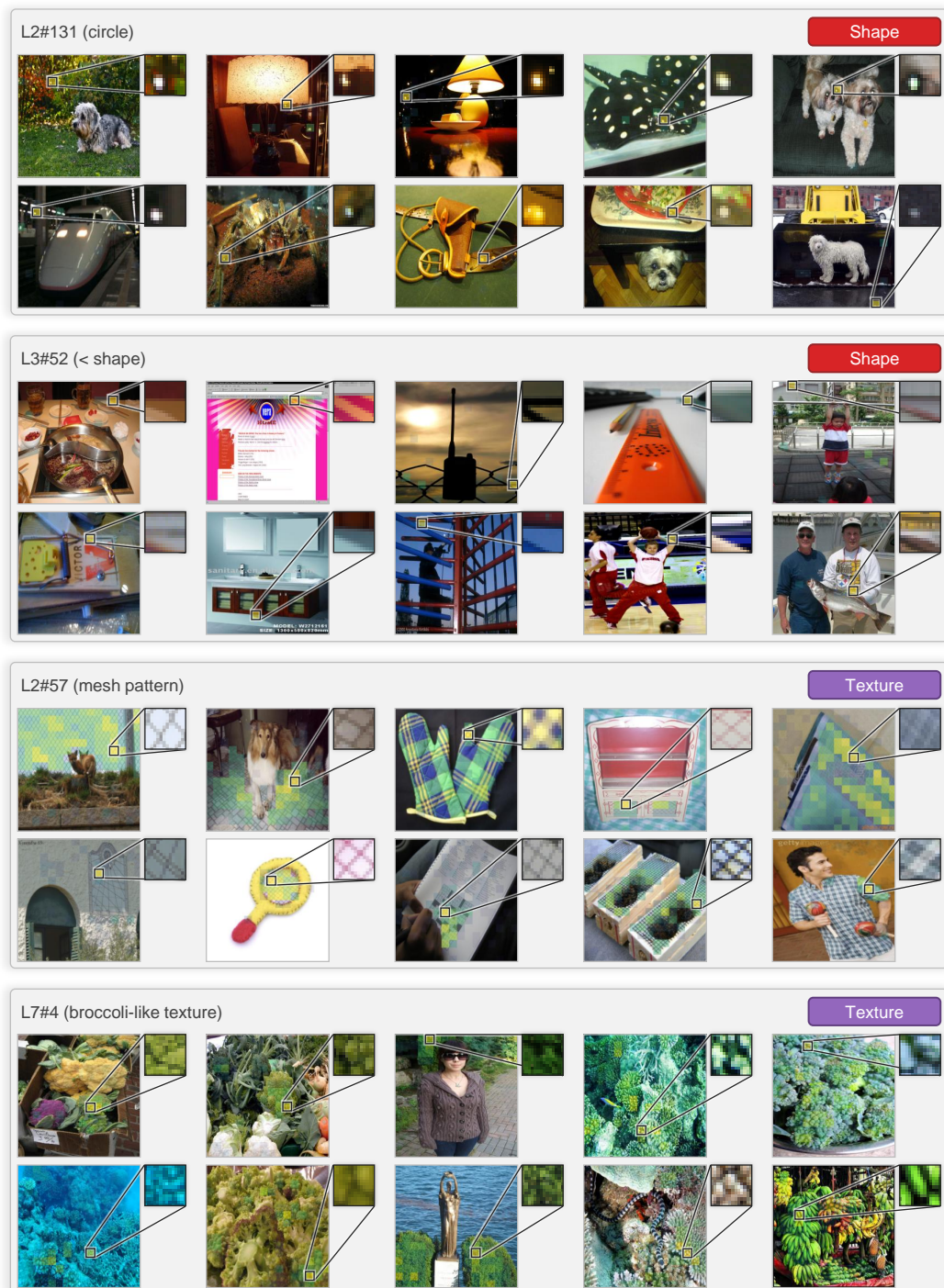


Figure 16: More DINOv2 Feature Examples (Shape and Texture).

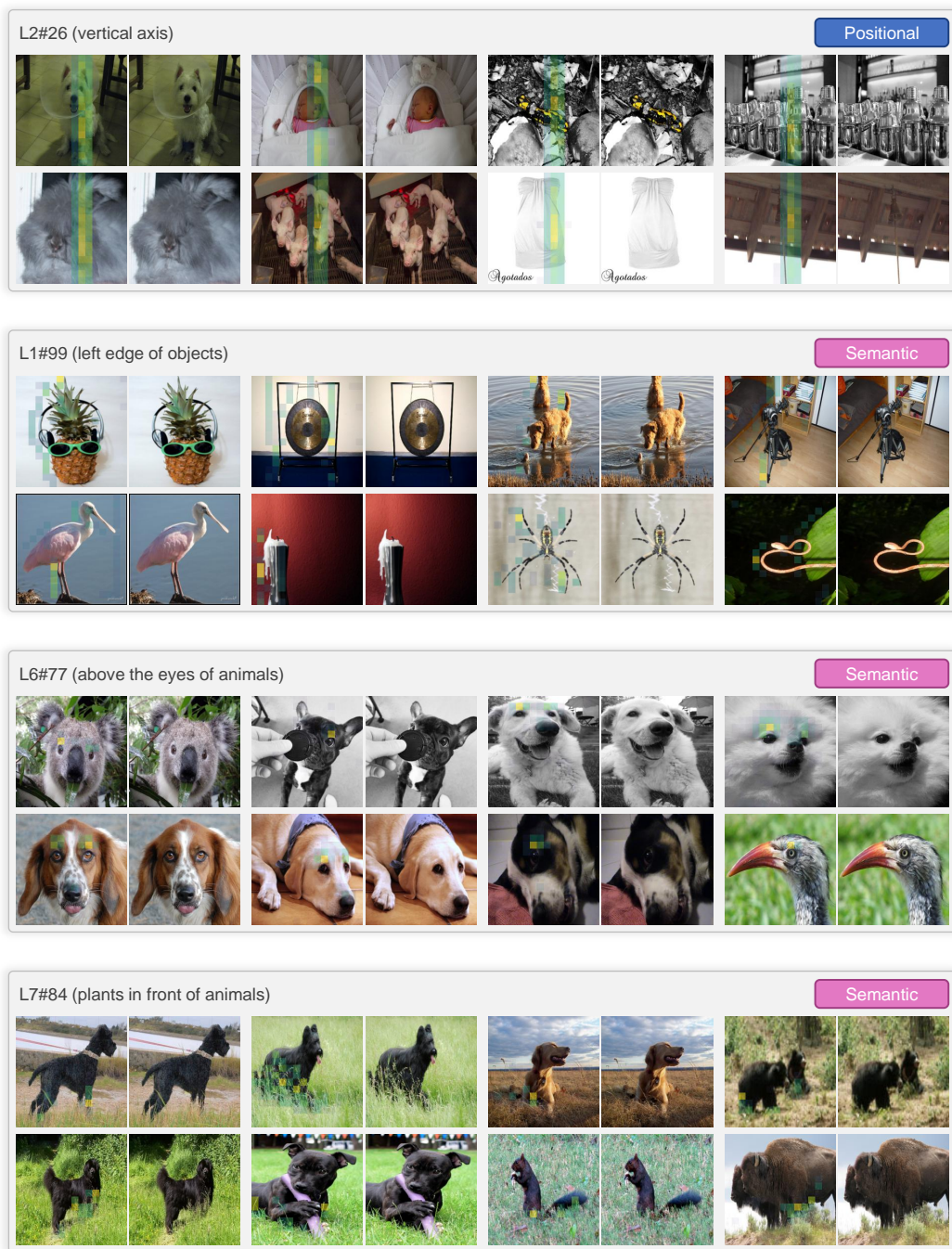


Figure 17: More DINOv2 Feature Examples (Positional and Semantic).



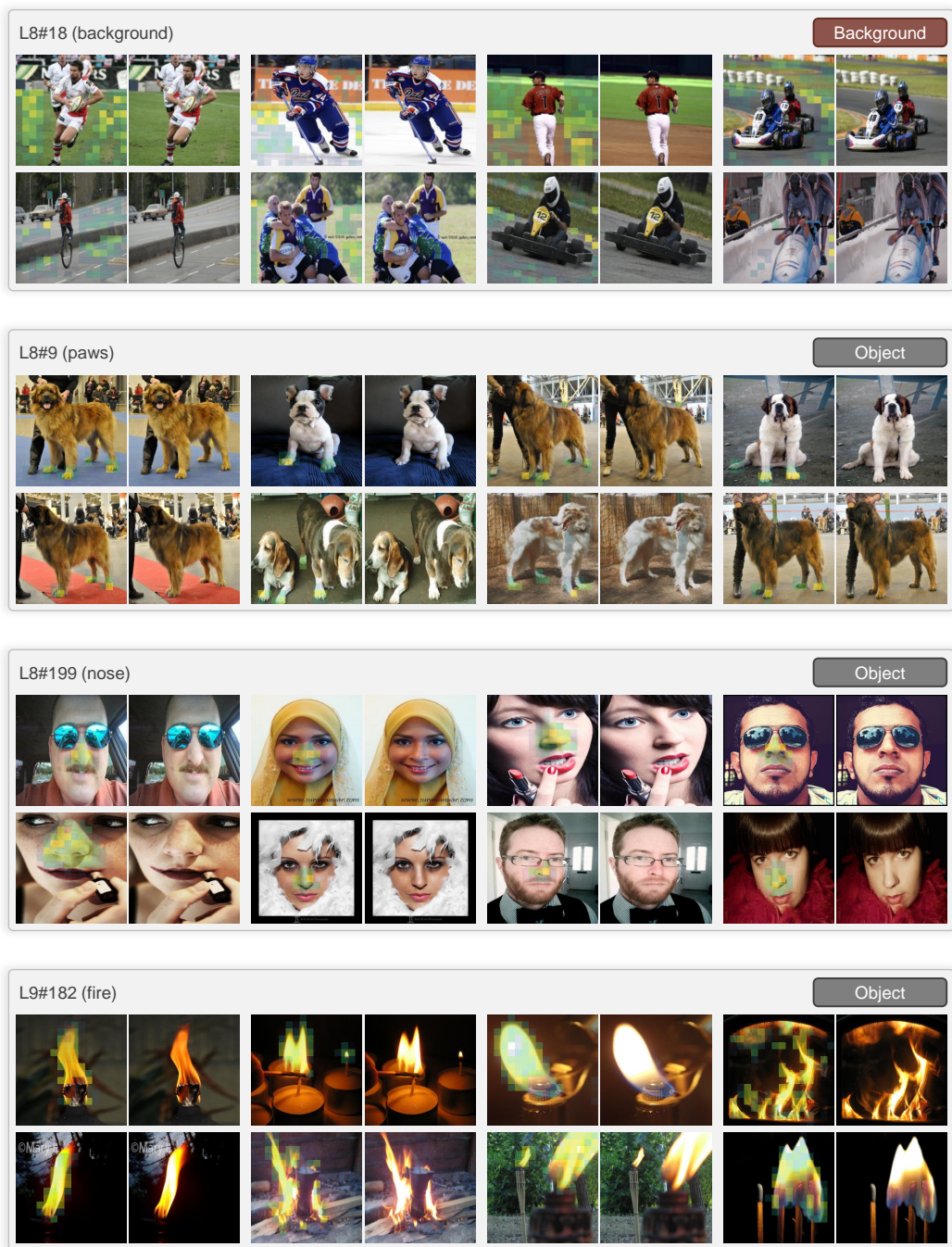


Figure 18: More DINOv2 Feature Examples (Background and Object).

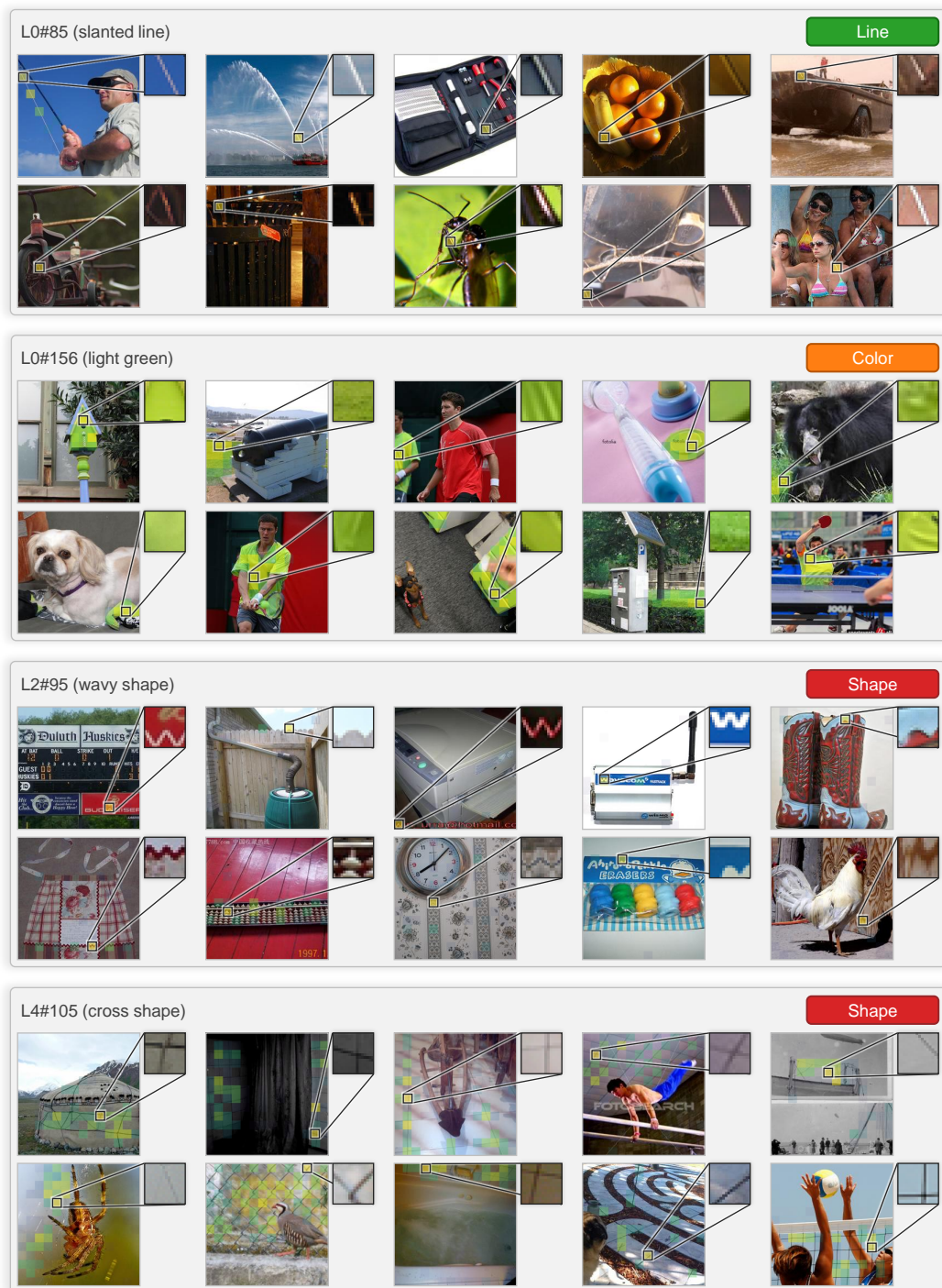


Figure 19: More CLIP Feature Examples (Line, Color, and Shape).



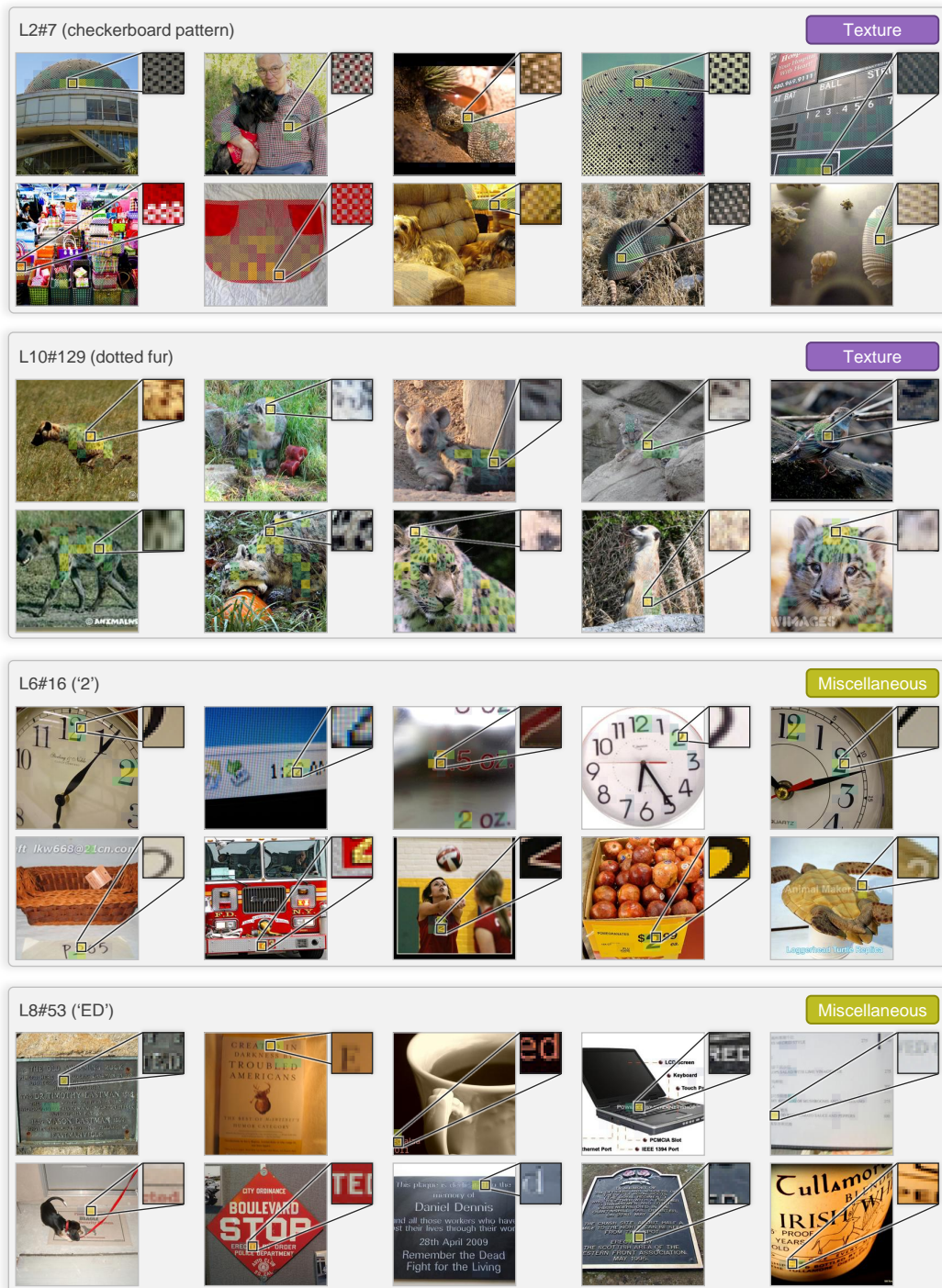


Figure 20: More CLIP Feature Examples (Texture and Miscellaneous).

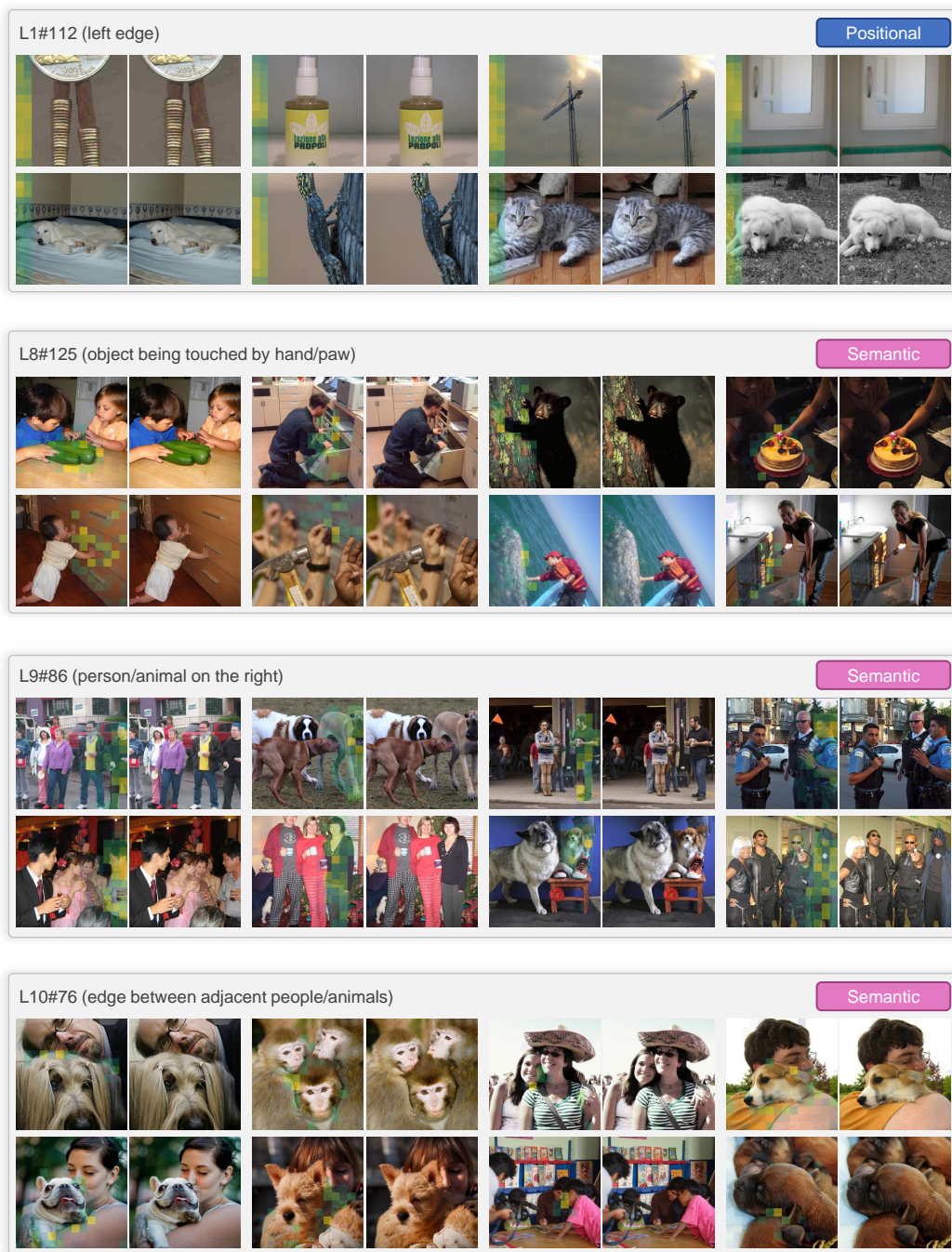


Figure 21: More CLIP Feature Examples (Positional and Semantic).



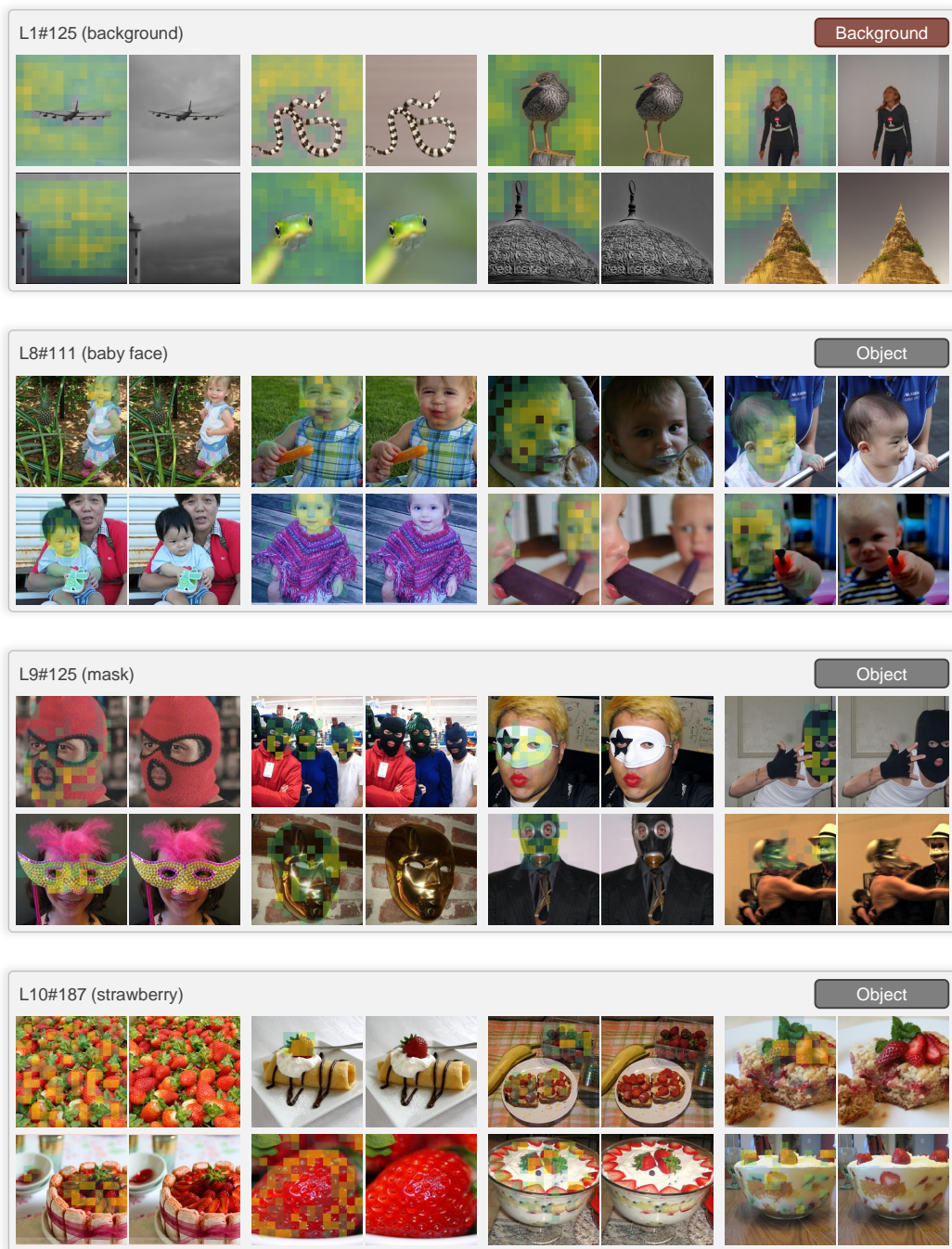


Figure 22: More CLIP Feature Examples (Background and Object).

Table 1: Circuit Evaluation for Various Node Selection Rules. We evaluate the faithfulness and completeness of the circuits using 1,500 randomly sampled images from the ImageNet validation set.

Rule	Faithfulness (%)			1 - Completeness (%)		
	ViT	DINOv2	CLIP	ViT	DINOv2	CLIP
top- $k$	94.1	85.1	82.3	99.6	99.8	99.7
top- $p$	95.0	85.4	82.9	99.7	99.7	99.7
threshold	93.6	84.6	82.2	99.6	99.8	99.7

## E Residual Replacement Model

### E.1 Design Choices

In §3.1, we select the top- $k$  features per layer based on the importance of their edges to the selected downstream nodes  $\mathcal{V}_{\ell+1}$ . We also consider two alternative selection strategies: (1) selecting the top  $p$ -percent of features in each SAE, and (2) selecting features until the cumulative importance of the selected features exceeds a threshold, i.e.,  $\sum_{\mathbf{u} \in \mathcal{V}_\ell} \sum_{\mathbf{d} \in \mathcal{V}_{\ell+1}} \mathbf{I}(\mathbf{u} \rightarrow \mathbf{d}) > \tau$ . As shown in Table 1, these alternative selection rules yield no significant differences in the faithfulness or completeness of the resulting circuits.

### E.2 Technical Challenges Details

**Scalability of Edge Importance Estimation** Let  $T$  be the number of tokens in the input image, and let  $f_u$  and  $f_d$  denote the number of features in the upstream and downstream SAEs, respectively. Let  $\mathbf{u}_t$  and  $\mathbf{d}_t$  represent the upstream and downstream features of the  $t$ -th token. Since we aggregate features across all tokens, each node is defined as the average over all tokens, i.e.,  $\mathbf{u} = \frac{1}{T} \sum_{t=1}^T \mathbf{u}_t$  and  $\mathbf{d} = \frac{1}{T} \sum_{t=1}^T \mathbf{d}_t$ .

The importance of the edge  $(\mathbf{u}, \mathbf{d})$  is computed as:

$$\mathbf{I}(\mathbf{u} \rightarrow \mathbf{d}) = \sum_{i=1}^T \sum_{j=1}^T \mathbf{I}(\mathbf{u}_i \rightarrow \mathbf{d}_j) = \sum_{i=1}^T \sum_{j=1}^T \nabla_{\mathbf{d}_j} m \nabla_{\mathbf{u}_i} \mathbf{d}_j (\mathbf{u}_i - \mathbf{u}'_i).$$

This requires computing the Jacobians for  $T^2$  token pairs per edge, resulting in a total of  $T f_u \times T f_d$  feature pairs. In practice, this would require  $\mathcal{O}(T \times f_d)$  backpropagation steps, which becomes computationally infeasible for large  $T$ .

To mitigate this, we apply the Jacobian-vector product trick to efficiently compute the Jacobian of the downstream features with respect to the upstream features:

$$\sum_{i=1}^T \sum_{j=1}^T \nabla_{\mathbf{d}_j} m \nabla_{\mathbf{u}_i} \mathbf{d}_j (\mathbf{u}_i - \mathbf{u}'_i) = \sum_{i=1}^T \nabla_{\mathbf{u}_i} \left[ \sum_{j=1}^T (\nabla_{\mathbf{d}_j} m) \cdot \mathbf{d}_j \right] (\mathbf{u}_i - \mathbf{u}'_i).$$

Note that  $\nabla_{\mathbf{d}_j} m$  is treated as a constant during computing the Jacobian of  $\mathbf{d}_j$  with respect to  $\mathbf{u}_i$ . This formulation reduces the complexity to  $\mathcal{O}(f_d)$  backward passes, significantly accelerating computation compared to the naïve approach. As a result, we can compute the full edge importance in a few seconds for a single image using a single RTX A6000 GPU.

**Noisy Gradients** Unlike language models, ViTs often suffer from noisy gradients, making gradient-based interpretations less stable [26, 28, 29]. To address this issue, we adopt LibraGrad [18], a recent method designed to stabilize gradients. They found that some modules in modern transformer architectures, such as attention mechanisms and layer normalizations, can disrupt the gradient flow, leading to noisy gradients. LibraGrad mitigates this issue by applying a gradient pruning technique, which removes the noisy gradients while preserving the informative ones. An important theoretical property of LibraGrad is that it satisfies *FullGrad-completeness* [16], meaning it decomposes the model output into the exact contributions from each input feature along with a bias term:

$$m(\mathbf{a}; \mathbf{b}) = \nabla_{\mathbf{a}} m(\mathbf{a}; \mathbf{b})^T \mathbf{a} + \nabla_{\mathbf{b}} m(\mathbf{a}; \mathbf{b})^T \mathbf{b},$$



where  $\mathbf{a}$  is the input feature,  $\mathbf{b}$  is the bias term, and  $m(\mathbf{a}; \mathbf{b})$  is the model output. This completeness property enables us to assign an explicit contribution to each input feature, akin to Layer-wise Relevance Propagation (LRP) [21, 23, 25–27]. We extend this decomposition to intermediate features, allowing us to quantify the contribution of each SAE feature to the final prediction through gradient-based analysis.

However, extending FullGrad-completeness to SAE features requires careful handling of the SAE’s normalization step. In standard SAEs, the input representations are normalized by dividing by the standard deviation during encoding and re-scaled during reconstruction. This normalization step introduces nonlinearity that can break the completeness property. Concretely, given an input  $\mathbf{x} \in \mathbb{R}^d$ , the SAE computes the normalized representation as:

$$\bar{\mathbf{x}} = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})},$$

where  $\mu(\mathbf{x})$  and  $\sigma(\mathbf{x})$  are the mean and standard deviation of the input, respectively. Similarly, during reconstruction, the SAE applies a re-scaling step:

$$\hat{\mathbf{x}} = \sigma(\mathbf{x})\hat{\mathbf{x}} + \mu(\mathbf{x}),$$

where  $\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{pre}}$  is the reconstructed representation. To preserve FullGrad-completeness, we block the backpropagation path through the standard deviation normalization parameters  $\sigma(\mathbf{x})$ . Under this modification, the contribution of the SAE features and the error term can be faithfully computed using a gradient-based decomposition.

Finally, while the bias term in FullGrad reflects the impact of higher-order interactions between intermediate features [16], we do not consider this term in our main analysis. Incorporating such interactions remains an interesting direction for future work.

### E.3 Feature Similarity Analysis

To further investigate the Granny Smith circuit in §3.3 (Figure 5 of the main text), we conduct two simple analyses to evaluate the similarity and continuity of features within the circuit.

First, we test whether features are preserved across layers through the residual stream. For each feature in a given layer, we identify the feature in the next layer whose decoder vector has the highest cosine similarity with it. We then check whether this most similar feature is included in the circuit. This analysis helps verify if the circuit retains geometrically similar features across layers [100, 101].

Second, we examine whether these cosine-similar pairs correspond to strong connections by inspecting whether the highest-weighted edges in the circuit connect features with high cosine similarity. This allows us to assess whether semantic similarity is aligned with edge importance.

Our results show that, for all layer pairs, the next-layer feature with the highest cosine similarity is consistently included in the circuit. Moreover, in all layers except for 0-1, 3-4, and 8-9, the highest-weighted edges connect to the features with the greatest cosine similarity. In the cases of layers 0-1 and 8-9, the error term appears to play a significant role in constructing the next-layer feature, which may explain the lack of direct feature preservation. For layers 3-4, the edge weights are relatively low, suggesting that the next-layer features are formed through a compositional combination of multiple upstream features. For example, L3#1283 and L3#1438 are both connected to L4#58 with similar edge weights. While their individual cosine similarities with L4#58’s decoder vector are moderate (0.44 and 0.48, respectively), their combined vector achieves a similarity of 0.62. Considering that the maximum cosine similarity between any single L3 feature and L4#58 is 0.68, this suggests that L3#1283 and L3#1438 jointly contribute to the construction of L4#58.

Overall, by combining the residual replacement model with feature similarity analysis, we can identify which features are preserved across layers through the residual stream and which features are compositionally combined to form higher-level representations [85, 86, 101–104].

### E.4 More Circuit Results

We provide additional qualitative results on the circuits in ViT (Figures 23 to 26), DINOv2 (Figures 27 to 30), and CLIP (Figures 31 to 34). To aid intuitive understanding of the features, we visualize the maximally activated patches for each feature in the first row and the corresponding maximally

232 activated images in the second row. The first column shows the input image along with its maximally  
233 activated patches and activation values.

## 234 **E.5 More Curve Circuits and Position Circuits**

235 We also provide additional examples of curve circuits and position circuits in DINOv2 and CLIP.  
236 We find that the circuits in DINOv2 and CLIP behave similarly to those in ViT. For curve circuits  
237 (Figures 35 and 36), lines at different angles are compositionally combined to form curve detectors.  
238 For position circuits (Figures 37 and 38), position detectors in early layers are combined to construct  
239 more complex detectors in deeper layers. For example, in Figure 37, various vertical position detectors  
240 and other features combine to form a top-and-bottom background detector (L4#1203). In Figure 38,  
241 we observe that a bottom position detector (L3#1515) and an object detector (L3#419) combine to  
242 form a bottom background detector (L4#70). We can also find that the object detector (L3#419) is  
243 influenced by a color detector (L2#84).

## 244 **E.6 Debiasing Spurious Correlations**

245 In this section, we provide additional details on the debiasing procedure. We construct the top-3  
246 feature circuits for seven ImageNet classes: hummingbird, freight car, koala, fireboat, hard  
247 disc, gondola, and racket, which are known to exhibit spurious correlations with frequently  
248 co-occurring features. For each class, we manually identify one spurious feature within the circuit  
249 and ablate it. Specifically, we ablate the following features: L9#1210 (bird feeder for hummingbird),  
250 L9#2371 (graffiti for freight car), L9#1369 (eucalyptus for koala), L9#1648 (water jet for fireboat),  
251 L9#2867 (label for hard disc), L9#307 (house/river for gondola), and L9#855 (tennis court/player for  
252 racket).

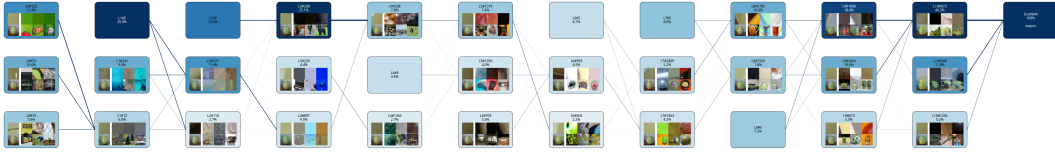


Figure 23: Teapot circuit in ViT. The circuit reveals that the model initially attends to the *grayish tone* of the teapot (L1#12) and its *specular highlights* (L1#241) in the lower layers. As it progresses through the intermediate layers, the model gradually captures the overall shape of the teapot (see the heatmap of the input image in the sub-graph L3#208  $\rightarrow$  L4#258  $\rightarrow$  L5#1374  $\rightarrow$  L6#603). In the higher layers, information about the teapot’s *handle* (L8#2785) and *body* (L8#2560) is integrated, leading to the model’s prediction of a teapot.



Figure 24: Street sign circuit in ViT. As the input progresses through the early and intermediate layers, the model gradually develops an understanding of the text written on the street sign. For instance, at L0#140, the characters are perceived as a *stripe pattern*; at L1#42, they are interpreted as a combination of *diagonal lines*; and at L2#10, as a composition of *curves*. Starting from L3, the model begins to recognize the characters as alphabetical and Arabic numerals. This understanding of the text is subsequently integrated with the *wide board* feature (L8#3063), allowing the model to grasp the overall appearance of the *signboard* (L9#1321). Finally, this representation is combined with the *signal light* features (L9#2068, L10#1492) to yield the prediction of a traffic light.

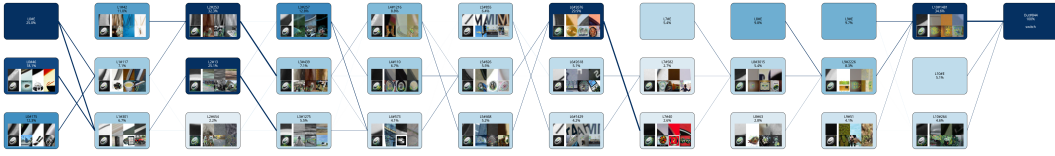


Figure 25: Switch circuit in ViT. Up to the intermediate layers, the model captures information about the areas surrounding the switch—both laterally (L2#253, L3#439) and vertically (L2#13, L3#1275). This information is integrated to form a representation of the switch’s *rounded shape* (L6#2676, L7#40). Beginning from layer 5, the model starts recognizing the Roman numerals written on the switch (L5#855, L6#1429). The information combines with the switch’s round shape (L7#40, L8#63), leading the model to predict the presence of a switch.

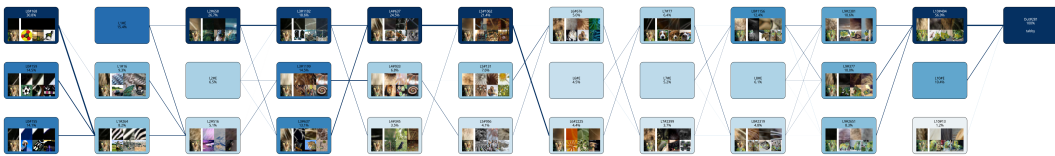


Figure 26: Tabby circuit in ViT. Up to layer 7, the model captures features related to *fur texture* (L1#16, L2#516) and the cat’s *whiskers* (L2#658, L3#1102, L4#637, L5#1062). In layer 8, both the *whiskers* (L8#2319) and the *cat’s eye* (L8#1156) significantly contribute to the model’s ability to identify the *cat’s face* in layer 9 (L9#2381). Notably, the detection of the *cat’s stripes* (L9#377) and *animal fur* (L9#2651) in layer 9 plays an integral role in forming the final representation of a *tabby cat* (L10#404).



Figure 27: Typewriter keyboard circuit in DINOv2. In the early layers, the model interprets the spaces between keys either as lines (e.g., L0#98, L1#69) or as textures (e.g., L2#136, L3#904). In the middle and later layers, the model captures the round and repetitive shapes of the keys (L5#88, L6#1350), leading to the recognition of a *keyboard* (L8#455, L9#1213). By further detecting the *characters* engraved on the keys (L9#1091, L10#275), the model arrives at the prediction of a typewriter.

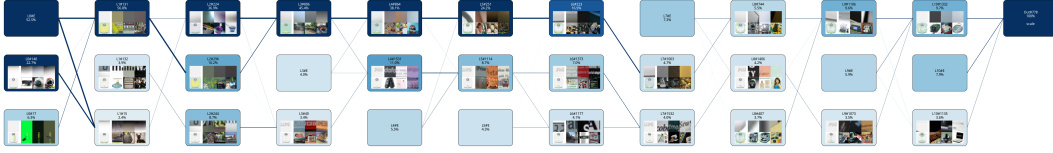


Figure 28: Scale circuit in DINOv2. Through the subgraph  $L1\#131 \rightarrow L2\#296 \rightarrow L3\#806$ , the model captures the white, monotone color of the scale. Simultaneously, it interprets the text printed on the scale via the subgraph  $L1\#132 \rightarrow L2\#244 \rightarrow L3\#40 \rightarrow L4\#1531 \rightarrow L5\#1114$ . Notably, in layer 6, the model appears to separately recognize the logo text (L6#1373) and the numerical values on the dial (L6#1177). In the later layers, the structural form of the scale (L8#744, L9#1106, L10#1332) is integrated with the textual and numerical information (L10#1135), ultimately leading the model to predict a scale.

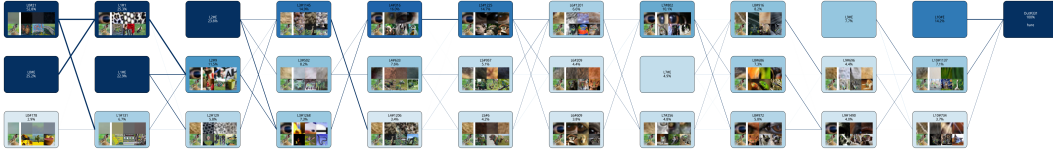


Figure 29: Hare circuit in DINOv2. In the early layers, the model detects the hare's eye as either a texture or a distinct shape (L1#1, L2#9, L3#1268), while simultaneously capturing the texture of the hare's fur (L2#129, L3#1145). In the middle layers, the eye is interpreted more semantically as an *animal eye* (L4#316, L5#1225), and the model begins to represent the overall body of the hare (L4#633, L5#6). In the subsequent layers, the model exhibits increasingly fine-grained understanding of the hare's face by attending to the eye (L6#609), the area below the eye (L6#1201), the region between the eye and nose (L6#209), and the snout (L9#696). In the final layer, the activation of the hare's face (L10#1137) and the animal's whiskers (L10#734) culminates in the model's prediction of a hare.

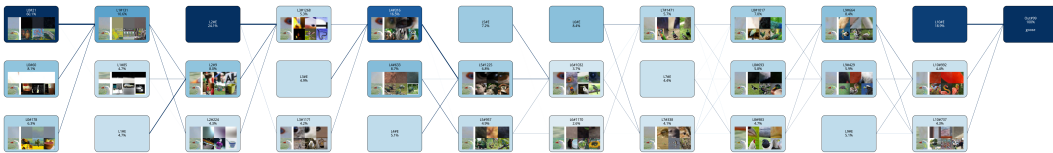


Figure 30: Goose circuit in DINOv2. In the early layers, the model captures the goose's coloration by separating it into white (L0#60), gray (L0#178), and edge regions (L2#224). In the middle layers, the model detects the left (L3#1268) and right (L3#1171) sides of the goose's eye as distinct shapes, which are subsequently integrated in later layers to form a semantic understanding of an *animal eye* (L4#316, L5#1225, L6#1032). In layers 7 and 8, the model attends to the goose's neck (L7#1471), cheek (L7#338), body (L8#1017), and beak (L8#983), ultimately leading to the prediction of a goose.

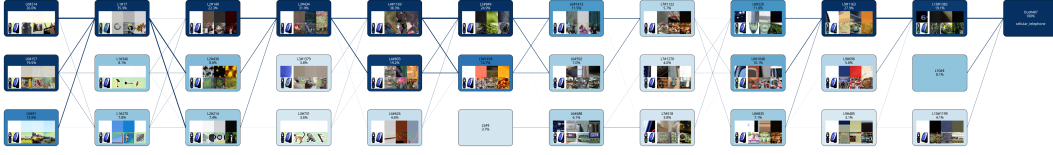


Figure 31: Cellular telephone circuit in CLIP. In the early layers, the model focuses on the blue, uniform color of the cell phone (L3#1379). In the middle layers, it identifies the keypad of the device (L6#688), and subsequently recognizes it as a button-equipped device (L7#518). This understanding is further refined as the model begins to interpret it as a time-displaying device (L8#520) and a communication device (L9#1163, L10#1382), ultimately leading to the prediction of a cell phone.



Figure 32: Tennis ball circuit in CLIP. In the early layers, the model captures the background of the tennis ball (e.g., L0#214, L1#17) as well as the texture of the ball (e.g., L2#41, L3#838). In the intermediate layers, it detects the edges on both sides of the tennis ball (L4#703) and the lower edge (L4#1406), leading to a representation of a *round object* (L5#540). This understanding is further refined into *multiple round objects* (L6#0), and in the subsequent layers, the model recognizes the object as a *ball* (L7#139, L8#274). By identifying the characteristic stripes on the ball (see the maximum activating patches in L9#957), the model ultimately predicts a tennis ball.

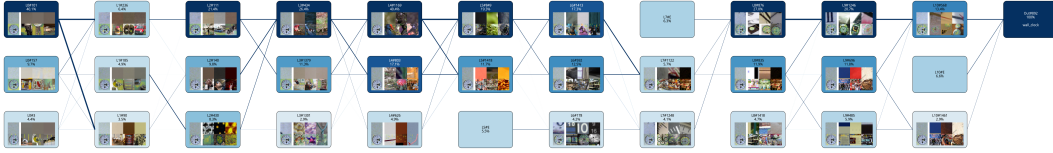


Figure 33: Wall clock circuit in CLIP. In the early layers, the model focuses on the background of the wall clock (L0#101, L0#157, L1#236). In the middle layers, it begins to recognize the *numbers* on the clock (L6#178), followed by the detection of the *tick marks* in the subsequent layer (L7#1248). The model then forms a representation of a *round-shaped clock* (L8#876), eventually activating general *clock* features (L9#1246, L10#568), which leads to the final prediction of a wall clock.

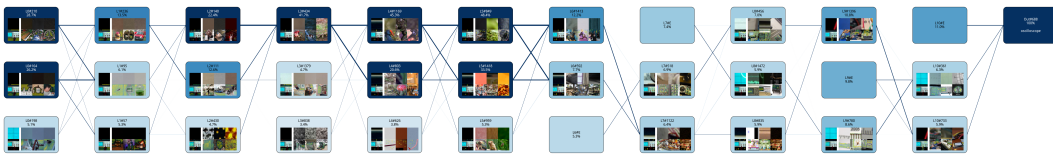


Figure 34: Oscilloscope circuit in CLIP. In the early and middle layers, the model recognizes the background of the oscilloscope (e.g., L0#210, L1#236, L2#111). In the later layers, it identifies the oscilloscope's buttons (L7#518), display (L8#1472), and body (L8#456). Subsequently, the *red line* connected to the oscilloscope (L9#1396) is interpreted as a *wire* (L10#733). This understanding, combined with the oscilloscope-specific features (L10#361), leads the model to predict an oscilloscope.

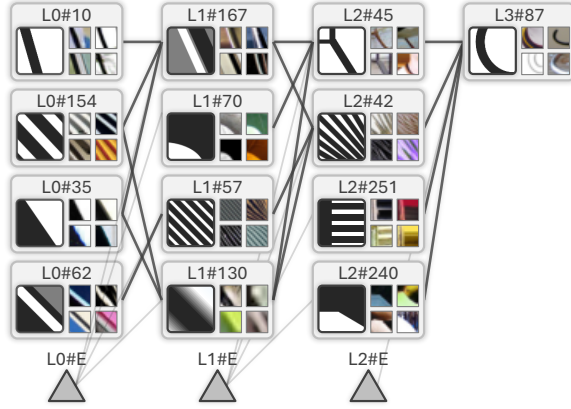


Figure 35: Curve Circuit (L3#87) for DINOv2.

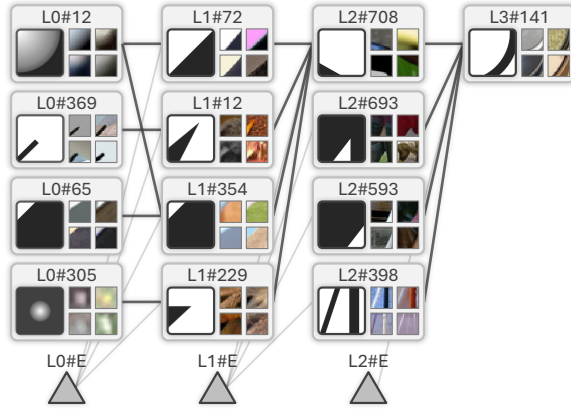


Figure 36: Curve Circuit (L3#141) for CLIP.

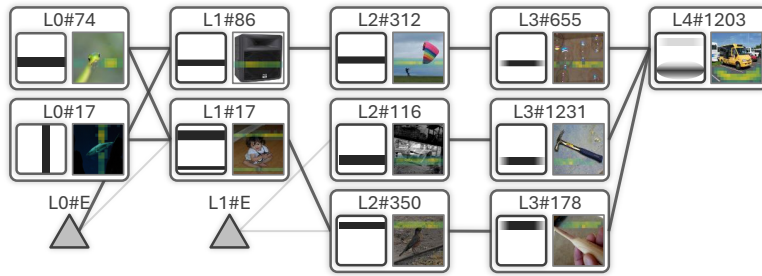


Figure 37: Position Circuit (L4#1203) for DINOv2.

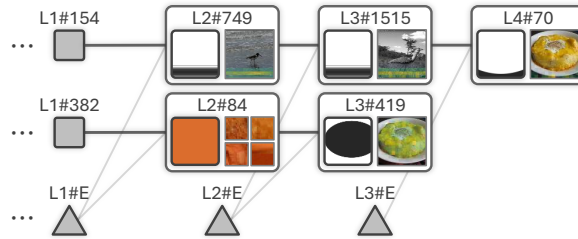


Figure 38: Position Circuit (L4#70) for CLIP.

## References

- [1] Yongjie Wang, Tong Zhang, Xu Guo, and Zhiqi Shen. Gradient based feature attribution in explainable ai: A technical review. *arXiv preprint arXiv:2403.10415*, 2024.
- [2] Shichang Zhang, Tessa Han, Usha Bhalla, and Himabindu Lakkaraju. Building bridges, not walls—advancing interpretability by unifying feature, data, and model component attribution. *arXiv preprint arXiv:2501.18887*, 2025.
- [3] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.
- [4] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.
- [5] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457. IEEE, 2017.
- [6] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *Advances in Neural Information Processing Systems*, 30, 2017.
- [7] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.
- [8] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.
- [9] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2950–2958. IEEE, 2019.
- [10] K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR, 2014.
- [11] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [12] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.
- [13] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.
- [14] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMIR, 2017.
- [15] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.
- [16] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. *Advances in neural information processing systems*, 32, 2019.
- [17] Joseph D Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *Journal of Machine Learning Research*, 22(104):1–54, 2021.
- [18] Faridoun Mehri, Mahdieh Soleymani Baghshah, and Mohammad Taher Pilehvar. Libragrad: Balancing gradient flow for universally better vision transformer attributions. *arXiv preprint arXiv:2411.16760*, 2024.



- 300 [19] S Bach, A Binder, G Montavon, F Klauschen, KR Müller, and W Samek. On pixel-wise  
301 explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS*  
302 *ONE*, 10(7):e0130140, 2015.
- 303 [20] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and  
304 Wojciech Samek. Layer-wise relevance propagation for neural networks with local renor-  
305 malization layers. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th*  
306 *International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9,*  
307 *2016, Proceedings, Part II* 25, pages 63–71. Springer, 2016.
- 308 [21] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-  
309 Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition.  
310 *Pattern recognition*, 65:211–222, 2017.
- 311 [22] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head  
312 self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of*  
313 *the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808,  
314 2019.
- 315 [23] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-  
316 Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting,*  
317 *explaining and visualizing deep learning*, pages 193–209, 2019.
- 318 [24] Zhengxuan Wu and Desmond C Ong. On explaining your explanations of bert: An empirical  
319 study with sequence classification. *arXiv preprint arXiv:2101.00196*, 2021.
- 320 [25] Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and  
321 Lior Wolf. Xai for transformers: Better explanations through conservative propagation. In  
322 *International Conference on Machine Learning*, pages 435–451. PMLR, 2022.
- 323 [26] Reduan Achitibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain,  
324 Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. Attnlrp: Attention-aware  
325 layer-wise relevance propagation for transformers. In *International Conference on Machine*  
326 *Learning*, pages 135–168. PMLR, 2024.
- 327 [27] Leila Arras, Bruno Puri, Patrick Kahardipraja, Sebastian Lapuschkin, and Wojciech Samek.  
328 A close look at decomposition-based xai-methods for transformer language models. *arXiv*  
329 *preprint arXiv:2502.15886*, 2025.
- 330 [28] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian  
331 McWilliams. The shattered gradients problem: If resnets are the answer, then what is the  
332 question? In *International conference on machine learning*, pages 342–350. PMLR, 2017.
- 333 [29] Ann-Kathrin Dombrowski, Christopher J Anders, Klaus-Robert Müller, and Pan Kessel. To-  
334 wards robust explanations for deep neural networks. *Pattern Recognition*, 121:108194, 2022.
- 335 [30] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Proceedings*  
336 *of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197,  
337 2020.
- 338 [31] Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar.  
339 Globenc: Quantifying global token attribution by incorporating the whole encoder layer in  
340 transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the*  
341 *Association for Computational Linguistics: Human Language Technologies*, pages 258–271,  
342 2022.
- 343 [32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining  
344 the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international*  
345 *conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- 346 [33] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and  
347 Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint*  
348 *arXiv:1805.10820*, 2018.



- [34] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. In *International conference on machine learning*, pages 110–119. PMLR, 2021.
- [35] Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 397–406, 2021.
- [36] Yao Qiang, Deng Pan, Chengyin Li, Xin Li, Rhongho Jang, and Dongxiao Zhu. Attcat: Explaining transformers via attentive class activation tokens. *Advances in neural information processing systems*, 35:5052–5064, 2022.
- [37] Junyi Wu, Bin Duan, Weitai Kang, Hao Tang, and Yan Yan. Token transformation matters: Towards faithful post-hoc explanation for vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10926–10935, 2024.
- [38] Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.
- [39] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. *Advances in Neural Information Processing Systems*, 33:5922–5932, 2020.
- [40] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.
- [41] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.
- [42] Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. From attribution maps to human-understandable explanations through concept relevance propagation. *Nature Machine Intelligence*, 5(9): 1006–1019, 2023.
- [43] Neel Nanda. Attribution patching: Activation patching at industrial scale. URL: <https://www.neelnanda.io/mechanistic-interpretability/attribution-patching>, 2023.
- [44] Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.
- [45] Aaqib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.
- [46] 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. *arXiv preprint arXiv:2403.19647*, 2024.
- [47] Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *arXiv preprint arXiv:2403.17806*, 2024.
- [48] 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. <https://distill.pub/2020/circuits/zoom-in>.
- [49] Haiyan Zhao, Fan Yang, Bo Shen, Himabindu Lakkaraju, and Mengnan Du. Towards uncovering how large language model works: An explainability perspective. *arXiv preprint arXiv:2402.10688*, 2024.
- [50] Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety—a review. *arXiv preprint arXiv:2404.14082*, 2024.

- [51] Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.
- [52] Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.
- [53] Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Attention heads of large language models: A survey. *arXiv preprint arXiv:2409.03752*, 2024.
- [54] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.
- [55] Martina G Vilas, Timothy Schaumlöffel, and Gemma Roig. Analyzing vision transformers for image classification in class embedding space. *Advances in neural information processing systems*, 36:40030–40041, 2023.
- [56] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.
- [57] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting clip’s image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.
- [58] Sriram Balasubramanian, Samyadeep Basu, and Soheil Feizi. Decomposing and interpreting image representations via text in vits beyond clip. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 81046–81076. Curran Associates, Inc., 2024. URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/file/93e45db754dd0f82339763055c6cda56-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2024/file/93e45db754dd0f82339763055c6cda56-Paper-Conference.pdf).
- [59] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting the second-order effects of neurons in clip. *arXiv preprint arXiv:2406.04341*, 2024.
- [60] Sukrut Rao, Sweta Mahajan, Moritz Böhle, and Bernt Schiele. Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery. In *European Conference on Computer Vision*, pages 444–461. Springer, 2024.
- [61] Hyesu Lim, Jinho Choi, Jaegul Choo, and Steffen Schneider. Sparse autoencoders reveal selective remapping of visual concepts during adaptation. *arXiv preprint arXiv:2412.05276*, 2024.
- [62] Harrish Thasarathan, Julian Forsyth, Thomas Fel, Matthew Kowal, and Konstantinos Derpanis. Universal sparse autoencoders: Interpretable cross-model concept alignment. *arXiv preprint arXiv:2502.03714*, 2025.
- [63] Samuel Stevens, Wei-Lun Chao, Tanya Berger-Wolf, and Yu Su. Sparse autoencoders for scientifically rigorous interpretation of vision models. *arXiv preprint arXiv:2502.06755*, 2025.
- [64] Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *arXiv preprint arXiv:2502.12892*, 2025.
- [65] Vladimir Zaigrajew, Hubert Baniecki, and Przemyslaw Biecek. Interpreting clip with hierarchical sparse autoencoders. *arXiv preprint arXiv:2502.20578*, 2025.
- [66] Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. *arXiv preprint arXiv:2504.02821*, 2025.

- [67] Sonia Joseph, Praneet Suresh, Ethan Goldfarb, Lorenz Hufe, Yossi Gandelsman, Robert Graham, Danilo Bzdok, Wojciech Samek, and Blake Aaron Richards. Steering clip’s vision transformer with sparse autoencoders. *arXiv preprint arXiv:2504.08729*, 2025.
- [68] Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.
- [69] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.
- [70] Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.
- [71] Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. *Advances in Neural Information Processing Systems*, 37:18506–18534, 2024.
- [72] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. An overview of early vision in inceptionv1. *Distill*, 5(4):e00024–002, 2020.
- [73] Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 5(6):e00024–003, 2020.
- [74] Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. Naturally occurring equivariance in neural networks. *Distill*, 5(12):e00024–004, 2020.
- [75] Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. *Distill*, 6(1):e00024–006, 2021.
- [76] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.
- [77] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.
- [78] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL <https://transformer-circuits.pub/2025/attribution-graphs/methods.html>.
- [79] Trenton Bricken, Siddharth Mishra-Sharma, Jonathan Marcus, Adam Jermyn, Christopher Olah, Kelley Rivoire, and Thomas Henighan. Stage-wise model diffing. *Transformer Circuits Thread*, 2024.
- [80] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 24375–24410. Curran Associates, Inc., 2024. URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/file/2b8f4db0464cc5b6e9d5e6bea4b9f308-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2024/file/2b8f4db0464cc5b6e9d5e6bea4b9f308-Paper-Conference.pdf).
- [81] Xuyang Ge, Fukang Zhu, Wentao Shu, Junxuan Wang, Zhengfu He, and Xipeng Qiu. Automatically identifying local and global circuits with linear computation graphs. *arXiv preprint arXiv:2405.13868*, 2024.

- [82] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermy, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL <https://transformer-circuits.pub/2025/attribution-graphs/biology.html>.
- [83] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- [84] 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. *arXiv preprint arXiv:2406.04093*, 2024.
- [85] Tim Lawson, Lucy Farnik, Conor Houghton, and Laurence Aitchison. Residual stream analysis with multi-layer saes. *arXiv preprint arXiv:2409.04185*, 2024.
- [86] Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. Evolution of sae features across layers in llms. *arXiv preprint arXiv:2410.08869*, 2024.
- [87] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.
- [88] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024.
- [89] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.
- [90] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [91] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.
- [92] Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? a visual exploration. *arXiv preprint arXiv:2212.06727*, 2022.
- [93] Thomas Fel, Thibaut Boissin, Victor Boutin, Agustin Picard, Paul Novello, Julien Colin, Drew Linsley, Tom Rousseau, Rémi Cadène, Lore Goetschalckx, et al. Unlocking feature visualization for deep network with magnitude constrained optimization. *Advances in Neural Information Processing Systems*, 36:37813–37826, 2023.
- [94] Judy Borowski, Roland S Zimmermann, Judith Schepers, Robert Geirhos, Thomas SA Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization. *arXiv preprint arXiv:2010.12606*, 2020.
- [95] Robert Geirhos, Roland S Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don’t trust your eyes: on the (un) reliability of feature visualizations. In *International Conference on Machine Learning*, pages 15294–15330. PMLR, 2024.

- 541 [96] Julien Colin, Lore Goetschalckx, Thomas Fel, Victor Boutin, Jay Gopal, Thomas Serre, and  
542 Nuria Oliver. Local vs distributed representations: What is the right basis for interpretability?  
543 *arXiv preprint arXiv:2411.03993*, 2024.
- 544 [97] Johnny Lin. Neuronpedia: Interactive reference and tooling for analyzing neural networks,  
545 2023. URL <https://www.neuronpedia.org>. Software available from neuronpedia.org.
- 546 [98] Adly Templeton. *Scaling monosemanticity: Extracting interpretable features from claude 3*  
547 *sonnet*. Anthropic, 2024.
- 548 [99] Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. Neurons in large language models:  
549 Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics ACL*  
550 *2024*, pages 1288–1301, 2024.
- 551 [100] Daniil Laptev, Nikita Balagansky, Yaroslav Aksenov, and Daniil Gavrilo. Analyze feature flow  
552 to enhance interpretation and steering in language models. *arXiv preprint arXiv:2502.03032*,  
553 2025.
- 554 [101] Nikita Balagansky, Ian Maksimov, and Daniil Gavrilo. Mechanistic permutability: Match  
555 features across layers. *arXiv preprint arXiv:2410.07656*, 2024.
- 556 [102] Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christo-  
557 pher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits*  
558 *Thread*, 2024.
- 559 [103] Davide Ghilardi, Federico Belotti, Marco Molinari, and Jaehyuk Lim. Accelerating sparse  
560 autoencoder training via layer-wise transfer learning in large language models. In *Proceedings*  
561 *of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*,  
562 pages 530–550, 2024.
- 563 [104] Junxuan Wang, Xuyang Ge, Wentao Shu, Qiong Tang, Yunhua Zhou, Zhengfu He, and  
564 Xipeng Qiu. Towards universality: Studying mechanistic similarity across language model  
565 architectures. *arXiv preprint arXiv:2410.06672*, 2024.