Interpreting the Inner-Workings of Vision Models

Yossi Gandelsman

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-44
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-44.html

May 7, 2025

Interpreting the Inner-Workings of Vision Models

By

Yossi Gandelsman

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering- Electrical Engineering & Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alexei A. Efros, Chair
Professor Jitendra Malik
Professor Jacob Steinhardt
Professor Phillip Isola

Spring 2025

Interpreting the Inner-Workings of Vision Models

Abstract

Interpreting the Inner-Workings of Vision Models

by

Yossi Gandelsman

Doctor of Philosophy in Engineering- Electrical Engineering & Computer Science

University of California, Berkeley

Professor Alexei A. Efros, Chair

The field of computer vision has recently transitioned from hand-engineering systems to learning them from large-scale datasets via deep learning. This shift motivates a new kind of observational science — closer in spirit to experimental biology than traditional engineering — which aims to discover what is being learned by deep learning models and why these models work. This science analyzes the emergent internal computation in deep vision models, hoping to discover the basic computational blocks that enable visual intelligence.

This thesis presents my initial steps in this observational AI science, focusing on interpreting the internal mechanisms of deep vision models. It showcases how this understanding is used to improve model generalization and unlock new tasks without any additional learning.

I begin with an in-depth analysis of a single vision-language model, CLIP-ViT, and attempt to explain the functionality of two main components in its vision encoder — the attention heads and the neurons. I show that automatic characterization of components is attainable and reveals surprisingly structured and interpretable behavior, such as heads specializing and polysemantic neuron roles. These interpretations enable the removal of spurious features from CLIP, zero-shot image segmentation, and automatic generation of adversarial images. Next, I show that some similar computational components, "Rosetta Neurons", emerge across a diverse set of models trained with different architectures, objectives, and supervision. These findings suggest that certain visual concepts and structures are inherently embedded in the natural world and can be learned by different models regardless of the specific task or architecture. That provides a path to a scalable understanding of vision models that can be used to repair and improve future models.

To my loving family

Contents

Acknowledgments

Navigating the ups and downs of a Ph.D. has been an adventure, but the true highlight has been the incredible people who walked this path with me. Their wisdom, encouragement, and friendship have made all the difference. I am deeply grateful for every conversation, challenge, and shared laugh that shaped these years.

I would first like to thank my advisor, Alyosha Efros, for his immense support along this journey. His honesty, curiosity, and care have been a constant source of inspiration. I thank him for being optimistic when I was skeptical, and for being skeptical when I was optimistic. I was truly lucky to have him as my advisor.

To members of my committee, Jitendra Malik, Jacob Steinhardt, and Phillip Isola, for their insightful feedback and questions that helped refine and strengthen this work. To senior collaborators, Bill Freeman, Amir Globerson, Xiaolong Wang, Trevor Darrell, Angjoo Kanazawa, and Tali Dekel, for shaping my research taste and research questions.

To Michal Irani, for welcoming me into the artful world of computer vision. To Assaf Shocher for teaching me how to take my first gradient descent steps. To Jacob Steinhardt for opening the door to the field of interpretability.

To my Berkeley friends, Evonne Ng and Ethan Weber for being the best nearest neighbors, to Jathushan Rajasegaran for photos and hikes in SF, to Ilija Radosavovic for late-night discussions about the future. To Eli Bronstein, for Batonchiki, tea, and meaningful dinners. To Dave Epstein for Yemenite coffee and to Ashish Kumar for the mandatory afternoon kitchen talks. To Vickie Ye for debugging me when I needed it, to Amil Dravid for helping me learn more about myself – and my research – than I ever expected. To Amir Bar, Hadar Avivi, and Taco, for making me feel at home and being my first Berkeley crew. To Yu Sun, for helping me figure out why I went to grad school in the first place.

To my industry mentors for supporting me during my early steps in research, Inbar Mosseri, Xinlei Chen, Taesung Park, Eli Shechtman, and Miki Rubinstein. To my collaborators that constantly expanded my horizons - Nick Jiang, Anish Kachinthaya, Shubham Goel, Xinyang Han, Sophia Koepke, Boyi Li, Ren Wang, Xueyang Yu, Oran Lang, Shir Amir, Michal Yarom, and Kfir Aberman. To my labmates for discussions that shaped me more than any research project, paper, or talk – Toru Lin, Allan Jabri, Shiry Ginosar, Vongani Maluleke, Justin Kerr, Chung Min Kim, Qianqian Wang, Tim Brooks, Bill Peebles, Tyler Bonnen, Aleks Holynski, and Konpat Preechakul.

Finally, to my parents, for unconditional love and care. To Mille, for helping with every life decision I made. To Liraz, for pushing me to fulfill my dreams and never giving up.

Chapter 1

Introduction

The field of artificial intelligence (AI) has changed immensely in the past 10 years. The early models of intelligence relied on hand-crafted features (e.g., edge detectors) and task-specific algorithms (e.g., object detection). These models were perfectly understood (by construction) – but they just didn't work well. Current models are very different – instead of being engineered, they are learned from large-scale datasets via deep learning. These models work much better, but they come with a price – we don't understand why and how they work.

The shift in AI from manually designing models to learning them from data has created a new type of science – observational *AI science*. This science is intellectually closer to experimental biology than to traditional engineering – it aims to discover *what* is being learned by deep learning models and *why* these models work on some tasks while failing on others. Discovering what is learned will hopefully allow us to extract the basic computational blocks of intelligence. It also enables the detection of model limitations and makes models more reliable. Understanding how these models work – extracting the emergent algorithms that enable generalization, provides a path to making these models truly intelligent and steering them toward our goals.

1.1 Opening the Black Box

I present initial directions for understanding the high-level algorithms that emerge in deep vision models. My approach for opening these black boxes is based on the fact that *deep neural networks are programs* – sequences of deterministic computations that are applied to the input to produce an output. To extend a program and to find and correct bugs in it, one should be able to read its code easily. The code in deep learning is the weights, but they are not easily interpretable by humans. I show preliminary results that indicate that some of the computation in deep neural networks can be reverse-engineered and lifted to a level of abstraction that humans can understand.

First, I focus on interpreting one widely-used deep vision model - the CLIP image encoder [64]. I investigate the encoder by analyzing how individual model components affect the final representation. I show that the image representation in CLIP can be decomposed as a sum across individual image patches, model layers, and attention heads. I then use CLIP's text representation to interpret the

summands by automatically describing them using text. Then, I extend my analysis beyond CLIP, and show that similar computations emerge in other models trained on different vision tasks, on different datasets, and with different architectures. I present a method to automatically draw these connections between model components and to allow the interpretation of components of other vision models.

Thesis overview

In Chapter 2, I focus on interpreting the attention head of CLIP. I characterize each head's role by automatically finding text representations that span its output space, which reveals property-specific roles for many heads (e.g. location or shape). Next, interpreting the image patches, I uncover an emergent spatial localization within CLIP. Finally, I use this understanding to remove spurious features from CLIP and to create a strong zero-shot image segmenter.

In Chapter 3, I shift from analyzing CLIP's attention layers to analyzing the complementary components - the neurons. I show that the method of Chapter 2 (i.e. the flow from a neuron through the residual stream to the output) or the indirect effects (overall contribution) fails to capture the neurons' function in CLIP. Therefore, I present the "second-order lens", analyzing the effect flowing from a neuron through the later attention heads, directly to the output. I then describe neurons by decomposing their effects into sparse sets of text representations. The sets reveal polysemantic behavior - each neuron corresponds to multiple, often unrelated, concepts (e.g. ships and cars). Exploiting this neuron polysemy, I mass-produce "semantic" adversarial examples by generating images with concepts spuriously correlated to the incorrect class. The results indicate that an automated interpretation of neurons can be used for model deception and for introducing new capabilities.

In Chapter 4, I aim to extend the capabilities to automatically interpret neurons to models beyond CLIP. I demonstrate the existence of common neurons ("Rosetta Neurons") across a range of models with different architectures, different tasks (generative and discriminative), and different types of supervision (class-supervised, text-supervised, self-supervised). I present an algorithm for mining a dictionary of such neurons across several popular vision models: Class Supervised-ResNet50 [34], DINO-ResNet50, DINO-ViT [11], MAE [35], CLIP-ResNet50 [64], BigGAN [10], StyleGAN-2 [41], and StyleGAN-XL [72]. The Rosetta Neurons facilitate model-to-model translation, enabling inversion-based image manipulations and editing without the need for specialized training. These findings suggest that certain visual concepts and structures are inherently embedded in the natural world and can be learned by different models regardless of the specific task or architecture.

Finally, In Chapter 5, I discuss future directions for the AI science. I discuss directions for enabling automated and scalable interpretability of deep neural networks, and possible future use cases for interpretability.

Chapter 2

Interpreting CLIP's Attention Layers

2.1 Introduction

Recently, [64] introduced CLIP, a class of neural networks that produce image representations from natural language supervision. As language is more expressive than previously used supervision signals (e.g. object categories) and CLIP is trained on a lot more data, its representations have proved useful on downstream tasks including classification [96], segmentation [51], and generation [70]. However, we have a limited understanding of what information is actually encoded in these representations.

To better understand CLIP, we design methods to study its internal structure, focusing on CLIP-ViT [17]. Our methods leverage several aspects of CLIP-ViT's architecture: First, the architecture uses *residual* connections, so the output is a sum of individual layer outputs. Moreover, it uses *attention*, so the output is also a sum across individual locations in the image. Finally, the representation lives in a joint vision-language space, so we can label its directions with text. We use these properties to decompose the representation into text-explainable directions that are attributed to specific attention heads and image locations.

As a preliminary step, we use the residual structure to investigate which layers have a significant direct effect on the output. We find that ablating all layers but the last 4 attention layers has only a small effect on CLIP's zero-shot classification accuracy (Section 2.3). We conclude that the CLIP image representation is primarily constructed by these late attention layers.

We next investigate the late attention layers in detail, leveraging the language space to uncover interpretable structure. We propose an algorithm, TEXTSPAN, that finds a basis for each attention head where each basis vector is labeled by a text description. The resulting bases reveal specialized roles for each head: for example, one head's top 3 basis directions are *A semicircular arch*, *A isosceles triangle* and *oval*, suggesting that it specializes in shapes (Figure 3.1(a)).

We present two applications of these identified head roles.First, we can reduce spurious correlations by removing heads associated with the spurious cue; we apply this on the Waterbirds dataset

This work was originally published as *Interpreting CLIP's Image Representation via Text-Based Decomposition*, Gandelsman et al, at ICLR 2024 [25]

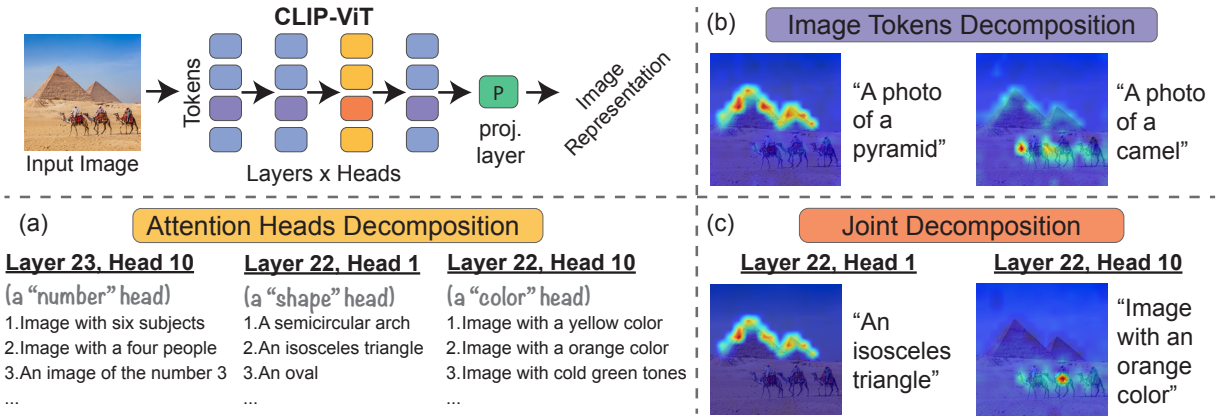

Figure 2.1: **CLIP-ViT image representation decomposition.** By decomposing CLIP's image representation as a sum across individual image patches, model layers, and attention heads, we can (a) characterize each head's role by automatically finding text-interpretable directions that span its output space, (b) highlight the image regions that contribute to the similarity score between image and text, and (c) present what regions contribute towards a found text direction at a specific head.

[71] to improve worst-group accuracy from 48% to 73%. Second, the representations of heads with a property-specific role can be used to retrieve images according to that property; we use it to perform retrieval based on discovered senses of similarity, such as color, location, and texture.

We next exploit the spatial structure provided by attention layers. Each attention head's output is a weighted sum across image locations, allowing us to decompose the output across these locations. We use this to visualize how much each location writes along a given text direction (Figure 3.1(b)). This yields a zero-shot image segmenter that outperforms existing CLIP-based zero-shot methods.

Finally, we consider the spatial structure jointly with the text basis obtained from TEXTSPAN. For each direction in the basis, the spatial decomposition highlights which image regions affect that basis direction. We visualize this in Figure 3.1(c), and find that it validates our text labels: for instance, the regions with triangles are the primary contributors to a direction that is labeled as *isosceles triangle*.

In summary, we interpret CLIP's image representation by decomposing it into text-interpretable elements that are attributed to individual attention heads and image locations. We discover property-specific heads and emergent localization, and use our discoveries to reduce spurious cues and improve zero-shot segmentation, showing that understanding can improve downstream performance.

2.2 Related Work

Vision model explainability. A widely used class of explainability methods produces heatmaps to highlight parts in the input image that are most significant to the model output [77, 84, 8, 86, 52, 12]. While these heatmaps are useful for explaining the relevance of specific image regions to the output, they do not show how attributes that lack spatial localization (e.g. object size or shape) affect the output. To address this, a few methods interpret models by finding counterfactual edits using

generative models [27, 49, 1]. All these methods aim to explain the output of the model without interpreting its intermediate computation.

Intermediate representations interpretability. An alternate way to explain vision models is to study their inner workings. One approach is to invert intermediate features into the input image space [16, 53, 29]. Another approach is to interpret individual neurons [5, 3, 18] and connections between neurons [60]. These approaches interpret models by relying only on visual outputs.

Few methods use text to interpret intermediate representations in vision models. [37] provide text descriptions for image regions in which a neuron is active. [91] project model features into a bank of text-based concepts. More closely to us, a few methods analyze CLIP's intermediate representations via text—[29] find multimodal neurons in CLIP that respond to different renditions of the same subject in images. [54] study entanglement in CLIP between images of words and natural images. We differ from these works by using CLIP's intrinsic language-image space and by exploiting decompositions in CLIP's architecture for interpreting intermediate representations.

Contrastive vision-language models. Contrastive vision-and-language models like CLIP [64] and ALIGN [40] showed promising zero-shot transfer capabilities for downstream tasks, including OCR, geo-localization, and classification [88]. Moreover, CLIP representations are used for segmentation [51], querying 3D scenes [42], and text-based image generation [66, 70]. We aim to interpret what information is encoded in these representations.

2.3 Decomposing CLIP Image Representation into Layers

We start by presenting the CLIP model [64] and describe how the image representation of CLIP-ViT is computed. We show that this representation can be decomposed into direct contributions of individual layers of the image encoder ViT architecture. Through this decomposition, we find that the last few attention layers have most of the direct effects on this representation.

CLIP-ViT Preliminaries

Contrastive pre-training. CLIP is trained to produce visual representations from images I coupled with text descriptions t. It uses two encoders—a transformer-based text encoder M_{text} and an image encoder M_{image}. Both M_{text} and M_{image} map to a shared vision-and-language latent space, allowing us to measure similarity between images and text via cosine similarity:

$$\text{sim}(I, t) = \langle M_{\text{image}}(I), M_{\text{text}}(T) \rangle / (||M_{\text{image}}(I)||_2 ||M_{\text{text}}(t)||_2) \tag{2.1}$$

Given a batch of images and corresponding text descriptions $\{(I_i, t_i)\}_{i \in \{1,\dots,k\}}$, CLIP is trained to maximize the similarity of the image representation $M_{\text{image}}(I_i)$ to its corresponding text representation $M_{\text{text}}(t_i)$, while minimizing $\text{sim}(M_{\text{image}}(I_i), M_{\text{text}}(t_j))$ for every $i \neq j$ in the batch.

Zero-shot classification. CLIP can be used for zero-shot image classification. To classify an image given a fixed set of classes, each name of a class (e.g. "Chihuahua") is mapped to a fixed template (e.g. "An image of a {class}") and encoded by the CLIP text encoder. The prediction for a given image is the class whose text description has the highest similarity to the image representation.

CLIP image representation. Several architectures have been proposed for computing CLIP's image representation. We focus on the variant that incorporates ViT [17] as a backbone. Here a vision transformer (ViT) is applied to the input image $I \in \mathbb{R}^{H\times W\times 3}$ to obtain a d-dimensional representation $\mathsf{ViT}(I)$. The CLIP image representation $M_{\text{image}}(I)$ is a linear projection of this output to a d'-dimensional representation in the joint vision-and-language space. Formally, denoting the projection matrix by $P \in \mathbb{R}^{d'\times d}$:

$$M_{\text{image}}(I) = P\mathsf{ViT}(I) \tag{2.2}$$

Both the parameters of the ViT and the projection matrix P are learned during training.

ViT architecture. ViT is a residual network built from L layers, each of which contains a multi-head self-attention (MSA) followed by an MLP block. The input I is first split into N non-overlapping image patches. The patches are projected linearly into N d-dimensional vectors, and positional embeddings are added to them to create the *image tokens* $\{z_i^0\}_{i\in\{1,...,N\}}$. An additional learned token $z_0^0 \in \mathbb{R}^d$, named the *class token*, is also included and later used as the output token.

Formally, the matrix $Z^0 \in \mathbb{R}^{d\times(N+1)}$, with the tokens $z_0^0, z_1^0, ..., z_N^0$ as columns, constitutes the initial state of the residual stream. It is updated for L iterations via these two residual steps:

$$\hat{Z}^l = \mathsf{MSA}^l(Z^{l-1}) + Z^{l-1}, \quad Z^l = \mathsf{MLP}^l(\hat{Z}^l) + \hat{Z}^l. \tag{2.3}$$

We denote the first column in the residual stream Z^l, corresponding to the class token, by $[Z^l]_{cls}$. The output of the ViT is therefore $[Z^L]_{cls}$.

MLP neurons in CLIP. The MLP layers are applied separately on each image token and the class token. They consist of an input linear layer, parametrized by $W_{\text{in}}^l \in \mathbb{R}^{N\times d}$, followed by a GELU non-linearity σ and an output linear layer, parametrized by $W_{\text{out}}^l \in \mathbb{R}^{d\times N}$. Here l is the layer number and N is the width (number of neurons) of the MLP. We next analyze the contributions of each individual neuron $n \in \{1, ..., N\}$ for each layer.

Decomposition into layers

The residual structure of ViT allows us to express its output as a sum of the direct contributions of individual layers of the model. Recall that the image representation $M_{\text{image}}(I)$ is a linear projection of the ViT output. By unrolling Eq. 2.3 across layers, the image representation can be written as:

$$M_{\text{image}}(I) = P\mathsf{ViT}(I) = P\left[Z^0\right]_{cls} + \underbrace{\sum_{l=1}^{L} P\left[\mathsf{MSA}^l(Z^{l-1})\right]_{cls}}_{\text{MSA terms}} + \underbrace{\sum_{l=1}^{L} P\left[\mathsf{MLP}^l(\hat{Z}^l)\right]_{cls}}_{\text{MLP terms}} \tag{2.4}$$

Eq. 2.4 decomposes the image representation into *direct contributions* of MLPs, MSAs, and the input class token, allowing us to analyze each term separately. We ignore here the *indirect effects* of the output of one layer on another downstream layer. We use this decomposition (and further decompositions) to analyze CLIP's representations in the next sections.

Both here and in Eq. 2.3, we ignore a layer-normalization term to simplify derivations. We address layer-normalization in detail in Section A.

	Base accuracy	+ MLPs ablation
ViT-B-16	70.22	67.04
ViT-L-14	75.25	74.12
ViT-H-14	77.95	76.30

Table 2.1: **MLPs mean-ablation.** We simultaneously replace all the direct effects of the MLPs with their average taken across ImageNet's validation set. This results in only a small reduction in zero-shot classification performance.

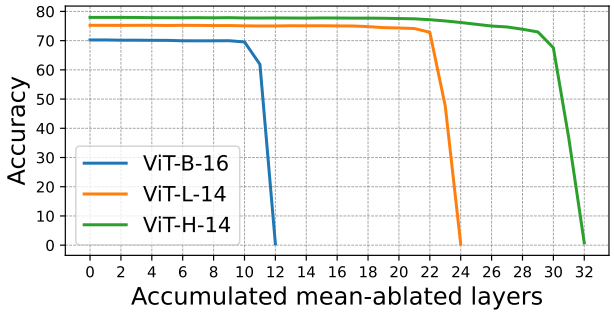

Figure 2.2: **MSAs accumulated mean-ablation.** We replace all the direct effects of the MSAs up to a given layer with their average taken across the ImageNet validation set. Only the replacement of the last few layers causes a large decrease in accuracy.

Evaluating the direct contribution of layers. As a preliminary investigation, we study which of the components in Eq. 2.4 significantly affect the final image representation, and find that the large majority of the direct effects come from the *late attention layers*.

To study the direct effect of a component (or set of components), we use mean-ablation [57], which replaces the component with its mean value across a dataset of images. Specifically, we measure the drop in zero-shot accuracy on a classification task before and after ablation. Components with larger direct effects should result in larger accuracy drops.

In our experiments, we compute means for each component over the ImageNet (IN) validation set and evaluate the drop in IN classification accuracy. We analyze the OpenCLIP ViT-H-14, L-14, and B-16 models [38], which were trained on LAION-2B [74].

MLPs have a negligible direct effect. Table 2.1 presents the results of simultaneously mean-ablating all the MLPs. The MLPs do not have a significant direct effect on the image representation, as ablating all of them leads to only a small drop in accuracy (1%-3%).

Only the last MSAs have a significant direct effect. We next evaluate the direct effect of different MSA layers. To do so, we mean-ablate all MSA layers up to some layer l. Figure 2.2 presents the results: removing all the early MSA layers (up to the last 4) does not change the accuracy significantly. Mean-ablating these final MSAs, on the other hand, reduces the performance drastically.

In summary, the direct effect on the output is concentrated in the last 4 MSA layers. We therefore focus only on these layers in our subsequent analysis, ignoring the MLPs and the early MSA layers.

Fine-grained decomposition into heads and positions

We present a more fine-grained decomposition of the MSA blocks that will be used in the next two sections. We focus on the output at the class token, as that is the only term appearing in Eq. 2.4. Following elhage2021mathematical, we write the MSA output as a sum over H independent

attention heads and the N input tokens:

$$\left[\mathsf{MSA}^l(Z^{l-1})\right]_{cls} = \sum_{h=1}^{H}\sum_{i=0}^{N} x_i^{l,h}, \quad x_i^{l,h} = \alpha_i^{l,h} W_{VO}^{l,h} z_i^{l-1} \tag{2.5}$$

where $W_{VO}^{l,h} \in \mathbb{R}^{d' \times d'}$ are transition matrices and $\alpha_i^{l,h} \in \mathbb{R}$ are the attention weights from the class token to the i-th token ($\sum_{i=0}^{N} \alpha_i^{l,h} = 1$). Therefore, the MSA output can be decomposed into direct effects of individual heads and tokens.

Plugging the MSA output definition in Eq. 3.2 into the MSA term in Eq. 2.4, we obtain:

$$\sum_{l=1}^{L} P\left[\mathsf{MSA}^l(T^{l-1})\right]_{cls} = \sum_{l=1}^{L}\sum_{h=1}^{H}\sum_{i=0}^{N} c_{i,l,h}, \quad c_{i,l,h} = P x_i^{l,h} \tag{2.6}$$

In other words, the total direct effect of all attention blocks is the result of contracting the tensor c across all of its dimensions. By contracting along only some dimensions, we can decompose effects in a variety of useful ways. For instance, we can contract along the spatial dimension i to get a contribution for each head: $c_{\text{head}}^{l,h} = \sum_{i=0}^{N} c_{i,l,h}$. Alternatively, we can contract along layers and heads to get a contribution from each image token: $c_{\text{token}}^{i} = \sum_{l=1}^{L}\sum_{h=1}^{H} c_{i,l,h}$.

The quantities $c_{i,l,h}$, $c_{\text{head}}^{l,h}$ and c_{token}^{i} all live in the d'-dimensional joint text-image representation space, which allows us to interpret them via text. For instance, given text description t, the quantity $\langle M_{\text{text}}(t), c_{\text{head}}^{l,h}\rangle$ intuitively measures the similarity of that head's output to description t.

2.4 Decomposition into Attention Heads

Motivated by the findings in Section 2.3, we turn to understanding the late MSA layers in CLIP. We use the decomposition into individual attention heads (Section 2.3), and present an algorithm for labeling the latent directions of each head with text descriptions. Examples of this labeling are depicted in Table 2.2 and Figure 2.4, with the labeling for all 64 late attention heads given in Section A.

Our labeling reveals that some heads exhibit specific semantic roles, e.g. "counting" or "location", in which many latent directions in the head track different aspects of that role. We show how to exploit these labeled roles both for property-specific image retrieval and for reducing spurious correlations.

Text-interpretable decomposition into heads

We decompose an MSA's output into text-related directions in the joint representation space. We rely on two key properties: First, the output of each MSA block is a sum of contributions of individual attention heads, as demonstrated in Section 2.3. Second, these contributions lie in the joint text-image representation space and so can be associated with text.

L21.H11 ("Geo-locations")	**L23.H10** ("Counting")	**L22.H8** ("Letters")
Photo captured in the Arizona desert	Image with six subjects	A photo with the letter V
Picture taken in Alberta, Canada	Image with four people	A photo with the letter F
Photo taken in Rio de Janeiro, Brazil	An image of the number 3	A photo with the letter D
Picture taken in Cyprus	An image of the number 10	A photo with the letter T
Photo taken in Seoul, South Korea	The number fifteen	A photo with the letter X
L22.H11 ("Colors")	**L22.H6** ("Animals")	**L22.H3** ("Objects")
A charcoal gray color	Curious wildlife	An image of legs
Sepia-toned photograph	Majestic soaring birds	A jacket
Minimalist white backdrop	An image with dogs	A helmet
High-contrast black and white	Image with a dragonfly	A scarf
Image with a red color	An image with cats	A table
L23.H12 ("Textures")	**L22.H1** ("Shapes")	**L22.H2** ("Locations")
Artwork with pointillism technique	A semicircular arch	Urban park greenery
Artwork with woven basket design	An isosceles triangle	Cozy home interior
Artwork featuring barcode arrangement	An oval	Urban subway station
Image with houndstooth patterns	Rectangular object	Energetic street scene
Image with quilted fabric patterns	A sphere	Tranquil boating on a lake

Table 2.2: **Top-5 text descriptions extracted per head by our algorithm.** Top 5 components returned by TEXTSPAN applied to ViT-L, for several selected heads. See Section A for results on all the heads.

Recall from Section 2.3 that the MSA terms of the image representation (Eq. 2.4) can be written as a sum over heads, $\sum_{l,h} c^{l,h}_{\text{head}}$. To interpret a head's contribution $c^{l,h}_{\text{head}}$, we will find a set of text descriptions that explain most of the variation in the head's output (the head "principal components").

To formalize this, we take input images $I_1, ..., I_K$ with associated head outputs $c_1, ..., c_K$ (for simplicity, we fix the layer l and head h and omit it from the notation). As $c_1, ..., c_K$ are vectors in the joint text-image space, each text input t defines a direction $M_{\text{text}}(t)$ in that space. Given a collection of text directions $\mathcal{T}$, let $\text{Proj}_{\mathcal{T}}$ denote the projection onto the span of $\{M_{\text{text}}(t) \mid t \in \mathcal{T}\}$. We define the *variance explained by* $\mathcal{T}$ as the variance under this projection:

$$V_{\text{explained}}(\mathcal{T}) = \frac{1}{K}\sum_{k=1}^{K} \| \text{Proj}_{\mathcal{T}}(c_k - c_{\text{avg}})\|_2^2, \text{ where } c_{\text{avg}} = \frac{1}{K}\sum_{k=1}^{K} c_k. \tag{2.7}$$

We aim to find a set of m descriptions $\mathcal{T}$ for each head that maximizes $V_{\text{explained}}(\mathcal{T})$. Unlike regular PCA, there is no closed-form solution to this optimization problem, so we take a greedy approach.

Greedy algorithm for descriptive set mining. To approximately maximize the explained variance in Eq. 2.7, we start with a large pool of candidate descriptions $\{t_i\}_{i=1}^{M}$ and greedily select from it to obtain the set $\mathcal{T}$.

Algorithm 1: TEXTSPAN

Input: Head (l, h) contribution $c^{l,h}_{\text{head}}$ for K images stacked as rows in a matrix $C \in \mathbb{R}^{K \times d'}$, a pool of M text descriptions $\{t_i\}_{i=1}^{M}$, their corresponding CLIP text representations $R \in \mathbb{R}^{M \times d'}$ (projected to the head output space), and basis size m

Output: A set of text descriptions $\mathcal{T}$ and projected representations $C' \in \mathbb{R}^{K \times d'}$

Initialization: $C' \leftarrow \mathbf{0}_{K \times d'}$, $\mathcal{T} \leftarrow \phi$

for *i in* $[1, ..., m]$ **do**

$D \leftarrow RC^T$

$j^* \leftarrow_{j=1}^{M} (D[j])$

$\mathcal{T} \leftarrow \mathcal{T} \cup \{t_{j^*}\}$

for *k in* $[1, ..., K]$ **do**

$C'[k] \leftarrow C'[k] + \frac{\langle C[k], R[j^*] \rangle}{||R[j^*]||^2} R[j^*]$

$C[k] \leftarrow C[k] - \frac{\langle C[k], R[j^*] \rangle}{||R[j^*]||^2} R[j^*]$

for *k in* $[1, ..., M]$ **do**

$R[k] \leftarrow R[k] - \frac{\langle R[k], R[j^*] \rangle}{||R[j^*]||^2} R[j^*]$

Our algorithm, TEXTSPAN, is presented in Alg. 1. It starts by forming the matrix $C \in \mathbb{R}^{K \times d'}$ of outputs for head (l, h), as well as the matrix $R \in \mathbb{R}^{M \times d'}$ of representations for the candidate descriptions, projected onto the span of C. In each round, TEXTSPAN computes the dot product between each row of R and the head outputs C, and finds the row with the highest variance $R[j^*]$ (the first "principle component"). It then projects that component away from all rows and repeats the process to find the next components. The projection step ensures that each new component adds variance that is orthogonal to the earlier components.

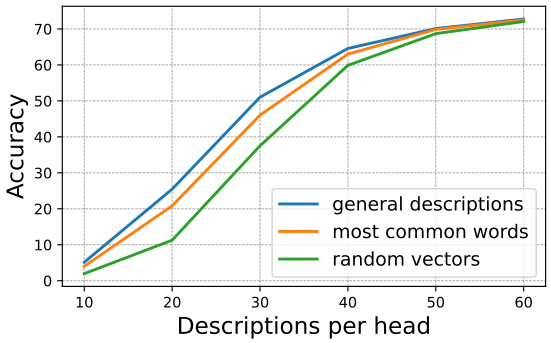

Figure 2.3: ImageNet classification accuracy for the image representation projected to TEXTSPAN bases. We evaluate our algorithm for different initial description pools, and with different output sizes.

TEXTSPAN requires an initial set of descriptions $\{t_i\}_{i=1}^{M}$ that is diverse enough to capture the output space of each head. We use a set of sentences that were generated by prompting ChatGPT-3.5 to produce general image descriptions. After obtaining an initial set, we manually prompt ChatGPT to generate more examples of specific patterns we found (e.g. texts that describe more colors). This results in 3498 sentences. In our experiments, we also consider two simpler baselines—one-word descriptions comprising the most common words in English, and a set of random d'-dimensional vectors that do not correspond to text (see Section A for the ChatGPT prompt and more details about the baselines).

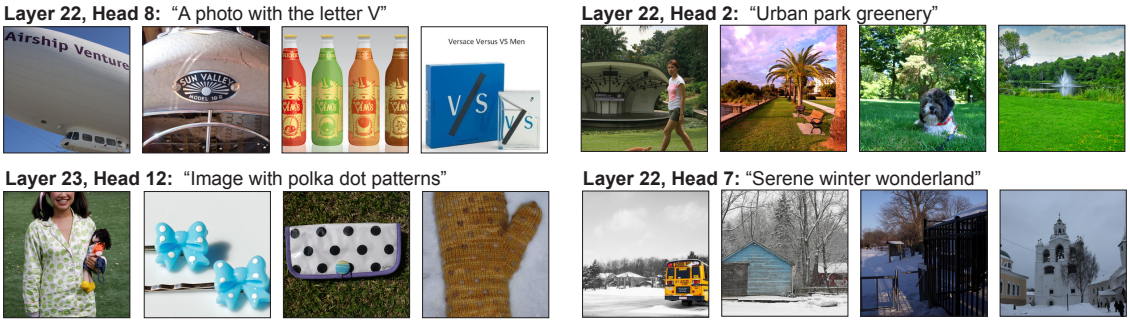

Figure 2.4: **Top-4 images for the top head description found by TEXTSPAN.** We retrieve images with the highest similarity score between $c^{l,h}_{\text{head}}$ and the top text representation found by TEXTSPAN. They correspond to the provided text descriptions. See Figure A.5 in the appendix for randomly selected heads.

Experiments

We apply TEXTSPAN to find a basis of text descriptions for all heads in the last 4 MSA layers. We first verify that this set captures most of the model's behavior and that text descriptions track image properties. We then show that some heads are responsible for capturing specific image properties (see Figure 3.1(1)). We use this finding for two applications—reducing known spurious cues in downstream classification and property-specific image retrieval.

Experimental setting. We apply TEXTSPAN to all the heads in the last 4 layers of CLIP ViT-L, which are responsible for most of the direct effects on the image representation (see Section 2.3). We consider a variety of output sizes $m \in \{10, 20, 30, 40, 50, 60\}$.

We first verify that the resulting text representations capture the important variation in the image representation, as measured by zero-shot accuracy on ImageNet. We simultaneously replace each head's direct contribution $c^{l,h}_{\text{head}}$ with its projection to the text representations $\text{Proj}_{\mathcal{T}(l,h)}\, c^{l,h}_{\text{head}}$ (where $\mathcal{T}(l,h)$ is the obtained text set for head (l,h)). We also mean-ablate all other terms in the representation (MLPs and the rest of the MSA layers).

The results are shown in Fig. 3.4: 60 descriptions per head suffice to reach 72.77% accuracy (compared to 75.25% base accuracy). Moreover, using our ChatGPT-generated descriptions as the candidate pool yields higher zero-shot accuracy than either common words or random directions, for all the different sizes m. In summary, we can approximate CLIP's representation by projecting each head output, a 768-dimensional vector, to a (head-specific) 60-dimensional text-interpretable subspace.

Some attention heads capture specific image properties. We report selected head descriptions from TEXTSPAN ($m = 60$) in Table 2.2, with full results in Appendix A. For some heads, the top descriptions center around a single image property like texture (L23H12), shape (L22H1), object count (L23H10), and color (L22H11). This suggests that these heads capture *specific image properties*. We qualitatively verify that the text tracks these image properties by retrieving the images with the largest similarity $\langle M_{\text{text}}(t_i), c^{l,h}_{\text{head}} \rangle$ for the top extracted text descriptions t_i. The results in Fig. 2.4 and A.5 show that the returned images indeed match the text.

Reducing known spurious cues. We can use our knowledge of head-specific roles to manually

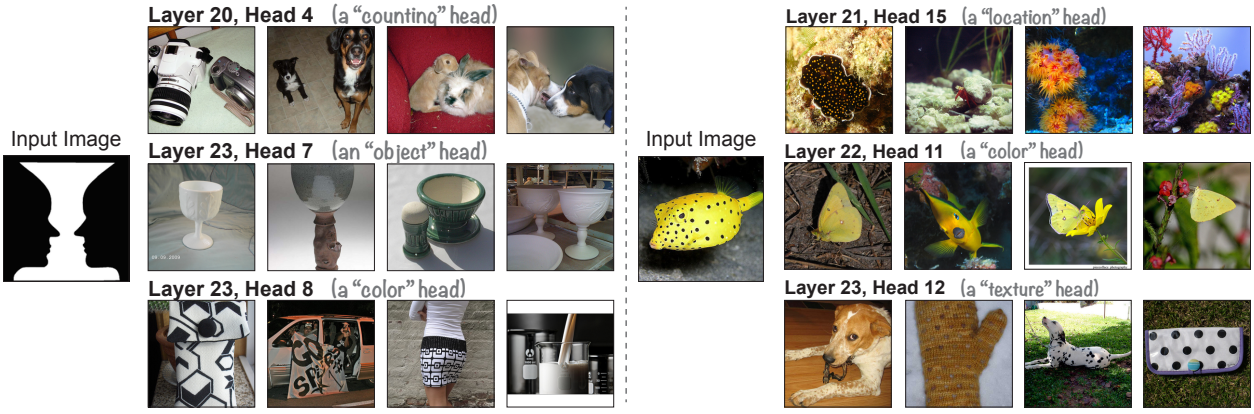

Figure 2.5: **Top-4 nearest neighbors per head and image.** We retrieve the most similar images to an input image by computing the similarity of the direct contributions of individual heads. As some heads capture specific aspects of the image (e.g. colors/objects), retrieval according to this metric results in images that are most similar regarding these aspects. See additional results in the appendix (Fig. A.6).

remove spurious correlations. For instance, if the location is being used as a spurious feature, we can ablate heads that specialize in geolocation to hopefully reduce reliance on the incorrect feature.

We validate this idea on the Waterbirds dataset [71], which combines waterbird and landbird photographs from the CUB dataset [87] with image backgrounds (water/land background) from the Places dataset [95]. Here image background is a spurious cue, and models tend to misclassify waterbirds on land backgrounds (and vice versa).

To reduce spurious cues, we manually annotated the role of each head using the text descriptions from TEXTSPAN, mean-ablated the direct contributions of all "geolocation" and "image-location" heads, and then evaluated the zero-shot accuracy on Waterbirds, computing the worst accuracy across subgroups as in [71]. As a baseline, we also ablated 10 random heads and reported the top accuracy out of 5 trials. As shown in Table 2.3, the worst-group accuracy increases by a large margin—by 25.2% for ViT-L. This exemplifies that the head roles we found with TEXTSPAN help us to design representations with less spurious cues, without any additional training.

Property-based image retrieval. Since some heads specialize to image properties, we can use their representations to obtain a property-specific similarity metric. To illustrate this, for a given head (h, l), we compute the inner product $\langle c^{l,h}_{\text{head}}(I), c^{l,h}_{\text{head}}(I')\rangle$ between a base image I and all other images in the dataset, retrieving the images with the highest similarity. Figure 2.5 shows the resulting nearest neighbors for heads that capture different properties. The retrieved images are different for each head and match the head-specific properties. In the left example, if we use a head that captures color for retrieval, the nearest neighbors are images with black-and-white objects. If we use a head that counts objects, the nearest neighbors are images with two objects.

2.5 Decomposition into Image Tokens

Decomposing the image representation across heads enabled us to answer *what* each head contributes to the output representation. We can alternately decompose the representation across image tokens to tell us *which image regions* contribute to the output for a given text direction $M_{\text{text}}(t)$. We find that these regions match the image parts that t describes, thereby yielding a zero-shot semantic image segmenter. We compare this segmenter to existing CLIP-based zero-shot methods and find that it is state-of-the-art. Finally, we decompose each head's direct contributions into per-head image tokens and use this to obtain fine-grained visualizations of the information flow from input images to output semantic representations.

Decomposing MSA outputs into image tokens. Applying the decomposition from Section 2.3, if we group the terms $c_{i,l,h}$ by position i instead of head (l,h), we obtain the identity $M_{\text{image}}(I) = \sum_{i=0}^{N} c^i_{\text{token}}(I)$, where $c^i_{\text{token}}(I)$ is the sum of the output at location i across all heads (l,h). We empirically find that the contribution of the class token c^0_{token} has a negligible direct effect on zero-shot accuracy (see mean-ablation in A). Therefore, we focus on the N image tokens.

We use the decomposition into image tokens to generate a heatmap that measures how much the output from each image position contributes to writing in a given text direction. Given a text description t, we obtain this heatmap by computing the score $\langle c^i_{\text{token}}(I), M_{\text{text}}(t)\rangle$ for each position i.

Quantitative segmentation results. We follow a standard protocol for evaluating heatmap-based explainability methods [12]. We first compute image heatmaps given descriptions of the image class (e.g. "An image of a {class}"). We then binarize them (by applying a threshold) to obtain a foreground/background segmentation. We compare the segmentation quality to zero-shot segmentations produced by other explainability methods in the same manner.

We evaluate the methods on ImageNet-segmentation [30], which contains a subset of 4,276 images from the ImageNet validation set with annotated segmentations. Table 2.4 displays the results: our decomposition is more accurate than existing methods across all metrics. See [12] for details about the compared methods and metrics, and additional qualitative comparisons in Section A.

Joint decomposition into per-head image tokens. Finally, we can jointly decompose the output of CLIP across both heads and locations. We use this decomposition to visualize what regions affect each of the basis directions found by TEXTSPAN. Recall that $c_{i,l,h}$ from Eq. 2.6 is the direct contribution of token i at head (h,l) to the representation. For each image token i, we take the inner products between $c_{i,l,h}$ and a basis direction $M_{\text{text}}(t)$ and obtain a *per-head* similarity heatmap. This visualizes the flow of information from input images to the text-labeled basis directions.

In Figure 2.6, we compute heatmaps for the two TEXTSPAN basis elements that have the largest and smallest (most negative) coefficients when producing each head's output. The highlighted regions match the text description for that basis direction—for instance, L22H13 is a geolocation head, its highest-activating direction for the top image is "Photo taken in Paris, France", and the image tokens that contribute to this direction are those matching the Eiffel Tower.

To normalize out bias terms, we subtract from the heatmap an averaged heatmap computed across all class descriptions in ImageNet.

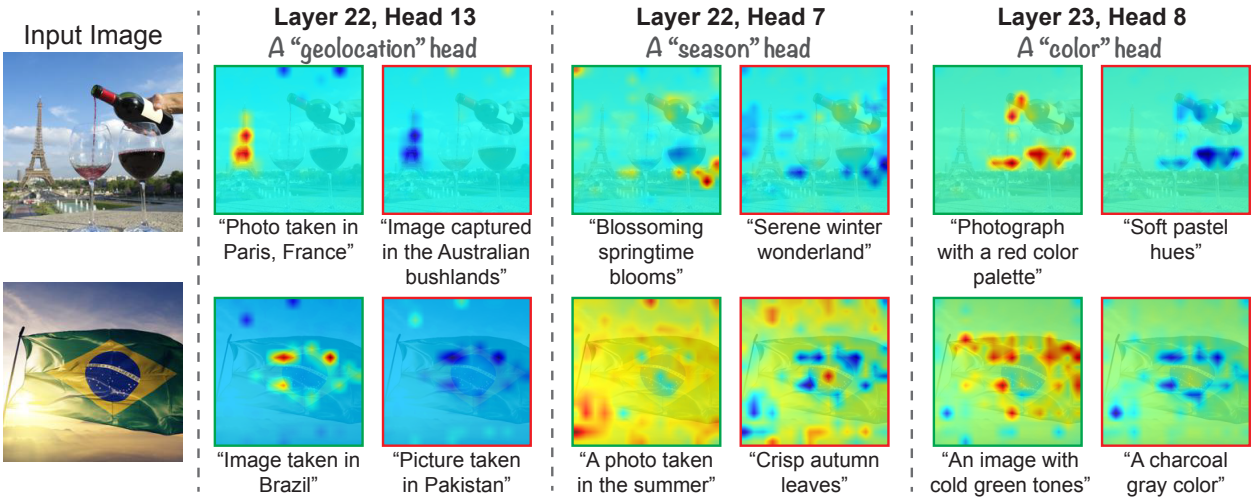

Figure 2.6: **Joint decomposition examples.** For each head (l, h), the left heatmap (green border) corresponds to the description that is most similar to $c^{l,h}_{\text{head}}$ among the TEXTSPAN output set. The right heatmap (red border) corresponds to the least similar text in this set (for $m = 60$). See Figure A.3 for more results.

	base	top random	ours
ViT-B-16	45.6	52.3	**57.5**
ViT-L-14	47.7	57.7	**72.9**
ViT-H-14	37.2	37.0	**43.3**

Table 2.3: **Worst-group accuracy on Waterbirds.** We reduce spurious cues by ablating property-specific heads. See Tables A.7-A.10 for fine-grained results.

	Pixel Acc. ↑	mIoU ↑	mAP ↑
LRP [8]	52.81	33.57	54.37
partial-LRP [86]	61.49	40.71	72.29
rollout [2]	60.63	40.64	74.47
raw attention	65.67	43.83	76.05
GradCAM [77]	70.27	44.50	70.30
Chefer *et al.*[12]	69.21	47.47	78.29
Ours	**75.21**	**54.50**	**81.61**

Table 2.4: **Segmentation performance on ImageNet-segmentation.** The image tokens decomposition results in significantly more accurate zero-shot segmentation than previous methods.

2.6 Limitations and Discussion

We studied CLIP's image representation by analyzing how individual model components affect it. Our findings allowed us to reduce spurious cues in downstream classification and improve zero-shot segmentation. We present two limitations of our investigation and conclude with future directions.

Indirect effects. We analyzed only the direct effects of model components on the representation. Studying indirect effects (e.g. information flow from early layers to deeper ones) can provide additional insights into the internal structure of CLIP and unlock more downstream applications.

Not all attention heads have clear roles. The outputs of TEXTSPAN show that not every head

captures a single image property (see results in Section A). We consider three possible explanations for this: First, some heads may not correspond to coherent properties. Second, the initial descriptions pool does not include descriptions of any image property. Third, some heads may collaborate and have a coherent role only when their outputs are addressed together. Uncovering the roles of more complex structures in CLIP can improve the performance of the described applications.

Future work. We believe that similar analysis for other CLIP architectures (e.g. ResNet) can shed light on the differences between the output representations of different networks. Moreover, our insights may help design better CLIP image encoder architectures and feature extractors for downstream tasks. We plan to explore these directions in future work.

Chapter 3

Interpreting CLIP's Neurons

3.1 Introduction

Automated interpretability of the roles of components in neural networks enables the discovery of model limitations and interventions to overcome them. Recently, such a technique was applied for interpreting the attention heads in CLIP [25], a widely used class of image representation models [64]. However, this approach has only scratched the surface, failing to explain a major set of CLIP's components—neurons. Here we will introduce a new interpretability lens for studying the neurons and use the gained understanding for zero-shot segmentation and mass-production of semantic adversarial examples.

Interpreting the neurons in CLIP is a harder task than interpreting the attention heads. First, there are more neurons than individual heads, which requires a more automated approach. Second, their direct effect on the output—the flow from the neuron, through the residual stream *directly* to the output—is negligible [25]. Third, most information is stored redundantly—many neurons encode the same concept, so just ablating a neuron (i.e. examining indirect effects) does not reveal much since other neurons make up for it.

The limitations presented above mean that we can neither look at the direct effect nor the indirect effect to analyze a single neuron. To address this, we introduce a "second-order lens" for investigating the *second-order effect* of a neuron—its total contribution to the output, flowing via all the consecutive attention heads (see Chapter 3.1).

We start by analyzing the empirical behavior of second-order effects of neurons. We find that these effects have high significance in the late layers. Additionally, each neuron is highly selective: its second-order effect is significant for only a small set (about 2%) of the images. Finally, this effect can be approximated by a single direction in the joint text-image representation space of CLIP (Chapter 3.3).

As each direction that corresponds to a neuron lives in a joint representation space, it can be decomposed as a sparse sum of text representations that describes the neurons' functionality (see

This work was will be presented as *Interpreting the Second-Order Effects of Neurons in CLIP*, Gandelsman et al, at ICLR 2025 [24]

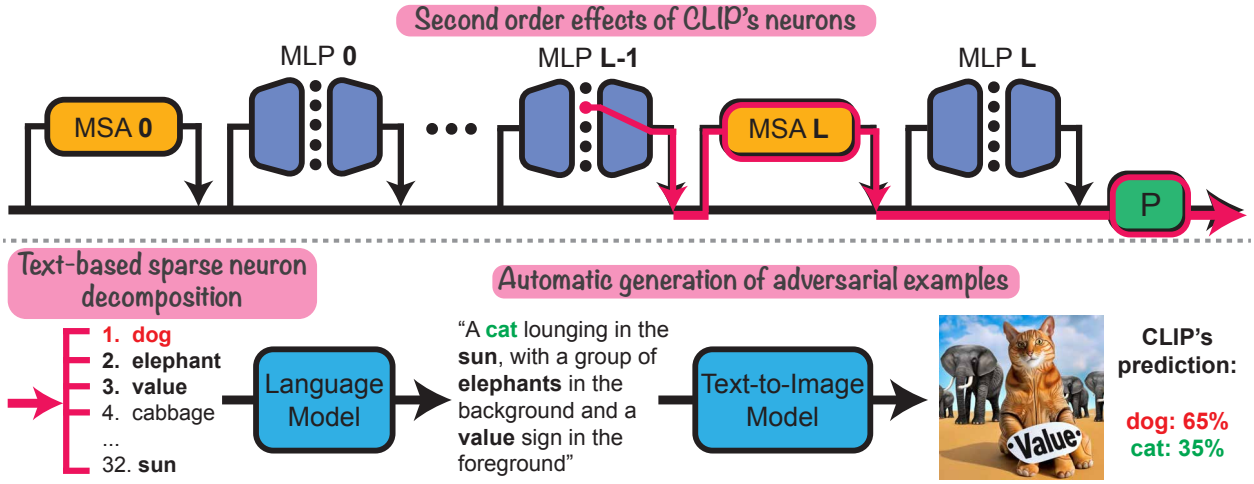

Figure 3.1: **Second order effects of CLIP's neurons.** Top: We analyze the second-order effects of neurons in CLIP-ViT (flow in pink). Bottom-left: Each second-order effect of a neuron can be decomposed to a sparse set of word directions in the joint text-image space. Bottom-right: co-appearing words in these sets can be used for mass-generation of semantic adversarial images.

Chapter 3.1). These text representations show that neurons are polysemantic [21]—each neuron corresponds to *multiple* semantic concepts. To verify that the neuron decompositions are meaningful, we show that these concepts correctly track which inputs activate a given neuron (Chapter 3.4).

The polysemantic behavior of neurons allows us to find concepts that inadvertently overlap in the network, due to being represented by the same neuron. We use these spurious cues for mass production of "semantic" adversarial examples that will fool CLIP (see bottom of Chapter 3.1). We apply this technique to automatically produce adversarial images for a variety of classification tasks. Our qualitative and quantitative analysis shows that incorporating spuriously overlapping concepts in an image deceives CLIP with a significant success rate (Chapter 3.5).

The text representations that describe the neurons' functionality enable an additional application—zero-shot segmentation. Mining for text representations of class names, we can identify class-relevant neurons with the second-order lens. Averaging the activation patterns of such neurons, we generate attribution heatmaps. Binarizing them yields a strong zero-shot image segmenter that outperforms recent work [12, 25].

In summary, we present an automated interpretability approach for CLIP's neurons by modeling their second-order effects and spanning them with text descriptions. We use these descriptions to automatically understand neuron roles and apply this to two applications. This shows that a scalable understanding of internal mechanisms both uncovers errors and elicits new capabilities from models.

3.2 Related work

Contrastive vision-language models. Models like ALIGN [40], CLIP [64], and its variants [93, 50] produce image representations from pre-training on images and their captions. They demonstrated impressive zero-shot capabilities for various downstream tasks, including OCR, geo-localization, and classification [88]. These models' representations are also used for segmentation [51], image generation [**rombach2021highresolution**, 66] and 3D understanding [42]. We aim to reveal the roles of neurons in such models.

Mechanistic interpretability of vision models. Mechanistic interpretability aims to reverse engineer the computation process in neural networks. In computer vision, this approach was applied to model individual network components [78] and to extract intermediate mechanisms like curve detectors [60], object segmenters [3, 5], high-frequency boundary detectors [73], and multimodal concepts detectors [29]. More closely to us, a few works made use of the intrinsic language-image space of CLIP to interpret the direct effect of attention heads and the output representation in CLIP with automatic text descriptions [25, 6]. We go beyond the output and direct effects of individual layers to interpret intermediate neurons in CLIP.

Neurons interpretability. The role of individual neurons (post-non-linearity single channel activations) is broadly studied in computer vision models [3, 5, 29] and language models [63, 26, 56]. [18, 31] demonstrate that neurons can learn universal mechanisms across different models in both domains. [21] show that neurons can be polysemantic (i.e. activated on *multiple* concepts) and exploit this property for generation of L2 adversarial examples. Some work seeks to extract neurons' concepts by learning sparse dictionaries [9, 65]. Other methods use large language models to automatically describe neurons based on which examples they activate on [7, 59, 37, 79]. In contrast, we focus on the contribution of neurons *to the output representation*.

3.3 Second-order effects of neurons

We start by deriving the second-order effect of neurons and presenting their benefits over the first-order and the indirect effects. Finally, we empirically characterize the second-order effects, setting the stage for automatically interpreting them via text in Chapter 3.4.

Analyzing the neuron effects on the output

As mentioned in Chapter 2.3, each MLP layer in CLIP consists of an input linear layer, parametrized by $W^l_{\text{in}} \in \mathbb{R}^{N \times d}$, followed by a GELU non-linearity σ and an output linear layer, parametrized by $W^l_{\text{out}} \in \mathbb{R}^{d \times N}$, where N is the number of neurons in the MLP (the width). Each individual neuron $n \in \{1, ..., N\}$ has different types of contributions to the output—the first-order (direct) effects, second-order effects, and (higher-order) indirect effects. We introduce them and explain the limitations of the direct and indirect effects before continuing to characterize the second-order effects in Chapter 3.3.

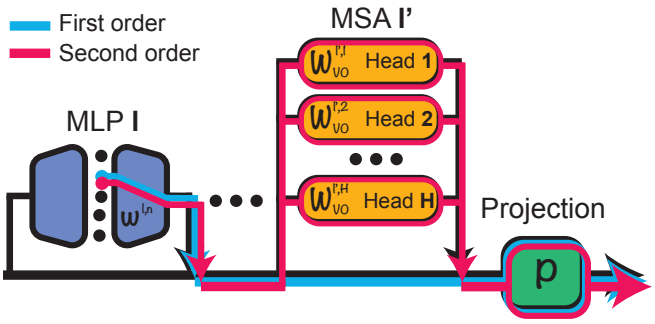

Figure 3.2: **First/Second-order effects.** The first order is the flow coming from a neuron to the projection layer and the output (blue). The second order goes from a single neuron through all the consecutive attention heads, to the projection layer, and to the output (pink).

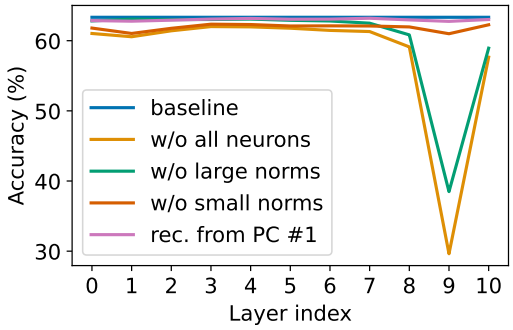

Figure 3.3: **Mean-ablation of second order effects (ViT-B-32).** We evaluate the performance on ImageNet validation set. Second-order effects concentrate in late layers, significant for only a part of the images, and can be approximated by one direction in the output space.

First-order effects (logit lens [58]). The first-order effect is the direct contribution of a component to the residual stream, multiplied by the projection layer (see blue flow in Chapter 3.2). For an individual neuron n in layer l, let $p_i^{l,n}(I) \in \mathbb{R}$ denote its post-GELU activation on the i-th token of the input image I. Then the contribution $e_i^{l,n}$ of the n-th neuron to the i-th token in the residual stream is:

$$e_i^{l,n} = p_i^{l,n}(I) w^{l,n} \tag{3.1}$$

where $w^{l,n} \in \mathbb{R}^d$ is the the n-th column of W_{out}^l. As the output representation is the class token (indexed 0) multiplied by P, the first-order effect for neuron n on the output is $Pe_0^{l,n}$.

As observed by [25], the first-order effects of MLP layers are close to constants in CLIP and most of the first-order contributions are from the late attention layers. We therefore focus on the second-order effects: the flow of information from the neurons through the attention layers.

Second-order effects. The contribution $e_i^{l,n}$ to the residual stream directly affects the input to later layers. We focus on the flow of $e_i^{l,n}$ through subsequent MSAs and then to the output (pink flow in Chapter 3.2). We call this interpretability lens the "second-order lens", in analogy to the "logit lens".

Following [20], the output of an MSA layer MSA^l that corresponds to the class token is a weighted sum of its $K+1$ input tokens $[z_0, ..., z_K]$:

$$\left[\text{MSA}^l([z_0, ..., z_K])\right]_0 = \sum_{h=1}^{H} \sum_{i=0}^{K} a_i^{l,h}(I) W_{VO}^{l,h} z_i \tag{3.2}$$

where $W_{VO}^{l,h} \in \mathbb{R}^{d \times d}$ are transition matrices (the OV matrices) and $a_i^{l,h}(I) \in \mathbb{R}$ are the attention weights from the class token to the i-th token ($\sum_{i=0}^{K} a_i^{l,h} = 1$).

To obtain the second-order effect of a neuron n at layer l, $\phi_n^l(I)$, we compute the additive contribution of the neuron through all the later MSAs and project it to the output space via P.

effect type	accuracy after mean-ablation	variance explained by first PC
indirect	52.3	11.0
second-order	29.6	48.2

Table 3.1: **Comparison to indirect effect.** We compare the second-order effects and the indirect effects by mean-ablating layer 9 in ViT-B-32 on ImageNet validation set.

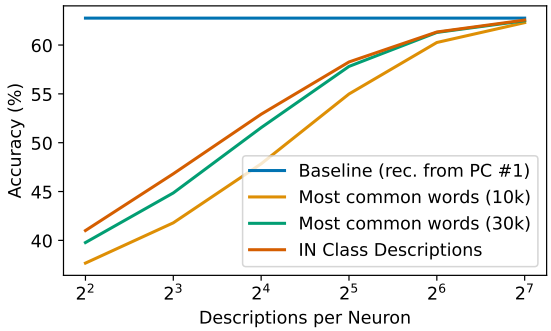

Figure 3.4: **Accuracy for neuron reconstructed from sparse text representations (ViT-B-32, layer 9).** We evaluate the sparse text decompositions for different initial description pools and description set sizes.

Plugging in Chapter 3.1 as the contribution to z_i in Chapter 3.2 and summing over layers, the second order effect of neuron n is then:

$$\phi_n^l(I) = \sum_{l'=l+1}^{L} \sum_{h=1}^{H} \sum_{i=0}^{K} \underbrace{\left(p_i^{l,n}(I) a_i^{l',h}(I)\right)}_{\text{attention-weighted activations}} \underbrace{\left(PW_{VO}^{l',h} w^{l,n}\right)}_{\text{input-independent}} \tag{3.3}$$

Indirect effects. An alternative approach is to analyze the indirect effect of a neuron by measuring the change in output representation when intervening on a neuron's output. Specifically, the intervention is done by replacing the activation $p_i^{l,n}$ of the neuron for each token with a pre-computed per-token mean. However, as was shown by [55], models often learn "self-repair" mechanisms that can obscure the individual roles of neurons. We illustrate these issues in the next section.

Characterizing the second-order effects

We analyze the empirical behavior of the second-order effects of neurons ϕ_n^l derived in the previous section. We find that only neurons from the late MLP layers have a significant second-order effect and that each individual neuron has a significant effect for less than 2% of the images. Finally, we show that ϕ_n^l can be approximated by one linear direction in the output space. These findings will help motivate our algorithm for describing output spaces of neurons with text in Chapter 3.4.

Experimental setting. To evaluate the second-order effects and their contributions to the output representation, we measure the downstream performance on the ImageNet classification task [15] after ablating these effects for each neuron. Specifically, we apply mean-ablation [57], replacing the additive contributions of individual $\phi_n^l(I)$'s to the representation with the mean computed across a dataset D. In our experiments, we mean-ablate all the neurons in a layer simultaneously and evaluate the downstream classification performance before and after ablation. Components with larger effects should result in larger accuracy drops.

We take D to be ~ 5000 images from the ImageNet training set. We report zero-shot classification accuracy on the ImageNet validation set. Our model is OpenAI's ViT-B-32 CLIP, which has 12 layers. We present additional results for ViT-L-14 and for ImageNet-R [36] in Chapter B and Chapter B.3.

Second-order effects concentrate in moderately late layers. We evaluate the contributions of all the ϕ_n^l across different layers and observe that the neurons with the most significant second-order effects appear relatively late in the model. The results for different layers in ViT-B-32 CLIP model are presented in Chapter 3.3 ("w/o all neurons"). As shown, mean-ablating layers 8-10 leads to the largest drop in performance. These layers appear right before the MSA layers with the most significant direct effects, as shown in [25] (layers 9-11; see Chapter B). The same trend is preserved for a larger model size as well (see Chapter B).

The second-order effect is sparse. We find that the second-order effect of each individual neuron is significant only for less than 2% of the images across the validation set. We repeat the same experiment as before, but this time we only mean-ablate $\phi_n^l(I)$ for a subset of images, while keeping the original effects for other images. For most of the images, except the subset of images in which $\phi_n^l(I)$ has a large norm, we can mean-ablate $\phi_n^l(I)$ without changing the accuracy significantly, as shown in Chapter 3.3 ("w/o small norm"). Differently, mean-ablating the contributions for the 100 images with the largest $\phi_n^l(I)$ norms results in a significant drop in performance ("w/o large norm"). The same trend is shown for images from ImageNet-R in Chapter B.3.

The second-order effect is approximately rank 1. While the second-order effect for a given neuron can write to different directions in the joint representation space for each image, we find that $\phi_n^l(I)$ can be approximated by one direction $r_n^l \in \mathbb{R}^{d'}$ in this space, multiplied by a coefficient $x_n^l(I)$ that depends on the image. We use the set S_n^l, which contains the largest second-order effects in norm from D, and set r_n^l to be the first principle component computed from S_n^l. We approximate $\phi_n^l(I)$ with $x_n^l(I)r_n^l + b_n^l$, where $b_n^l \in \mathbb{R}^{d'}$ is the bias computed by averaging $\phi_n^l(I)$ across D, and $x_n^l(I) \in \mathbb{R}$ is the norm of the projection of $\phi_n^l(I)$ onto r_n^l.

To verify that this approximation recovers $\phi_n^l(I)$ we replace each $\phi_n^l(I)$ for each neuron and image in the validation set with the approximation. We then evaluate the downstream classification performance. As shown in Chapter 3.3 ("reconstruction from PC #1"), for each layer l, this replacement results in a negligible drop in performance from the baseline, that uses the full representation. The same behavior is observed for ViT-L model and for a different initial set of images in the Appendix.

Comparison to indirect effect. We compare the second-order effect to the indirect effect and present the variance explained by the first principle component for each of them and the drop in performance when simultaneously mean-ablating all the effects from one layer. As shown in Chapter 3.1, Mean-ablating the indirect effects results in a smaller drop in performance due to self-repair behavior. Moreover, the first principle component explains significantly less of the variance in the indirect effect, than in the second-order effect. This demonstrates two advantages of the second-order effects—uncovering neuron functionality that is obfuscated by self-repair, and one-dimensional behavior that can be easily modeled and decomposed, as we will show in the next section.

Neuron	ImageNet class descriptions	Common words (30k)
#4	+"Picture with falling snowflakes" +"Picture portraying a person [...] in extreme weather conditions" -"Picture with a bucket in a construction site" +"Photograph taken during a holiday service"	+"snowy" +"frost" +"closings" +"advent"
#391	+"Image with a traditional wooden sled" +"Image with a wooden cutting board" +"Picture showcasing beach accessories" -"Photograph with a syringe and a surgical mask"	+"woodworking" -"swelling" +"cedar" +"heirloom"
#2137	+"Photo with a lime garnish" +"Image with candies in glass containers" -"Picture featuring lifeboat equipment" +"Close-up photo of a melting popsicle"	+"refreshments" +"gelatin" +"sour" +"cosmopolitan"
#2914	+"Photo that features a stretch limousine" +"Image capturing a suit with pinstripes" +"Caricature with a celebrity endorsing the brand" +"Image showcasing a Bullmastiff's prominent neck folds"	+"motorhome" +"yacht" +"cirrus" +"cabriolet"

Table 3.2: **Examples of sparse decompositions (ViT-B-32, layer 9).** We present the top-4 texts corresponding to the sparse decomposition of each neuron and the signs of the decomposition coefficients, for two initial pools ($m = 128$). See Chapter B.1 for more neurons.

3.4 Sparse decomposition of neurons

We aim to interpret each neuron by associating its second-order effect with text. We build on the previous observation that each second-order effect of a neuron ϕ_n^l is associated with a vector direction r_n^l. Since r_n^l lies in a shared image-text space, we can decompose it to a sparse set of text directions. We use a sparse coding method [61] to mine for a small set of texts for each neuron, out of a large pool of descriptions. We evaluate the found texts across different initial pools with different set sizes.

Decomposing a neuron into a sparse set of descriptions. Given the first principal component of the second-order effect of each neuron, r_n^l, we will decompose it as a sparse sum of text directions t_j: $r_n^l \approx \hat{r}_n^l = \sum_{j=1}^{M} \gamma_j^{l,n} M_{\text{text}}(t_j)$. To do this, we start from a large pool T of M text descriptions (e.g. the most common words in English). We apply a sparse coding algorithm to approximate r_n^l as the sum above, where only m of the $\gamma_j^{l,n}$'s are non-zero, for some $m \ll M$.

Experimental settings. We verify that the reconstructed $\hat{r}_n^l$ from the text representations captures the variation in the image representation, as measured by zero-shot accuracy on ImageNet. We simultaneously replace the neurons' second-order contributions in a single layer with the approximation $x_n^l(I)\hat{r}_n^l + b_n^l$.

To obtain sparse decomposition for each neuron, we use scikit-learn's implementation of orthogonal matching pursuit [61]. We consider two strategies for constructing the pool of text

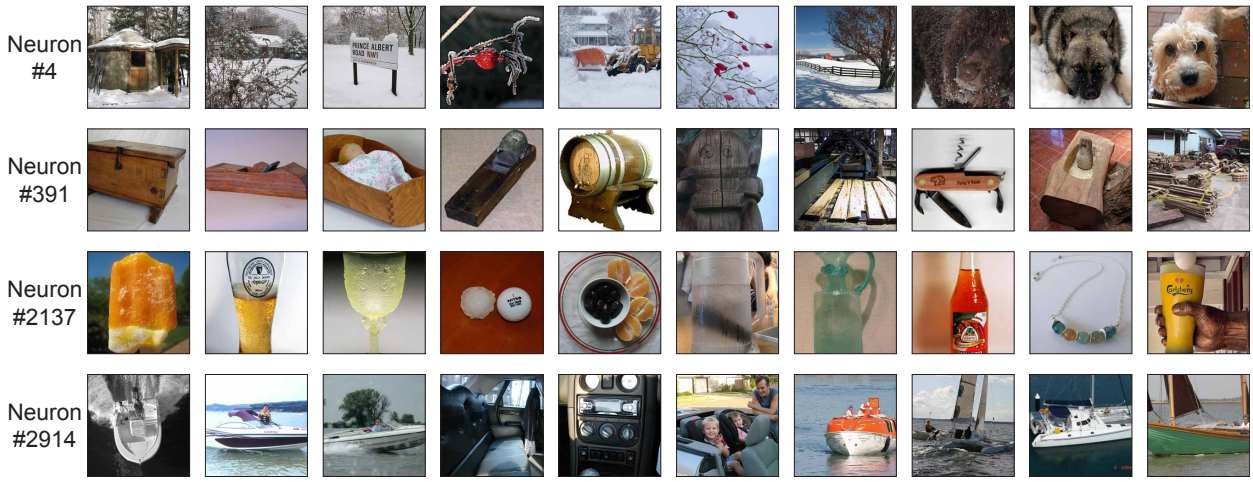

Figure 3.5: **Images with largest second-order effect norm per neuron.** We present the top images from 10% of ImageNet validation set for the neurons in Chapter 3.2. Note that neurons are polysemantic - they have large second-order effects on images that show multiple concepts (e.g. cars and boats). See top-50 images in Chapter B.6.

descriptions T. The first type is single words - the 10k and 30k most common words in English. The second type is image descriptions - we prompt ChatGPT-3.5 to produce descriptions of images that include an object of a specific class. Repeating this process for all the ImageNet (IN) classes results in $\sim$28k unique image descriptions. We then evaluate the reconstruction of r_n^l for different m's and pools.

Effect of sparse set size m and different pools. We experiment with $m \in \{4, 8, 16, 32, 64, 128\}$ and the three text pools, and present the accuracy on 10% of ImageNet validation set in Chapter 3.4. We approach the original classification accuracy with 128 text descriptions per neuron reconstruction $\hat{r}_n^l$. Using full descriptions outperforms using single words for the text pool, but the gap vanishes for larger m.

Qualitative results. We present the images with the largest second-order norms in Chapter 3.5, and the corresponding top-4 text descriptions in Chapter 3.2. As shown, the found descriptions match the objects in the top 10 images. Moreover, some individual neurons correspond to multiple concepts (e.g. writing both toward "yacht" and a type of a car - "cabriolet"). This property is even more apparent if more nearest neighbors are presented (see Chapter B.6 for the top 50 nearest neighbors). This corroborates with previous literature on neurons' polysemantic behavior [21] - single neurons behave as a superposition of multiple interpretable features. This property will allow us to generate adversarial images in Chapter 3.5.

3.5 Applications

Automatic generation of adversarial examples

The sparse decomposition of r_n^l's allows us to find overlapping concepts that neurons are writing to. We use these spurious cues to generate semantic adversarial images. Our pipeline, shown in Chapter 3.1, mines for spurious words that correlate with the incorrect class in CLIP (e.g. "elephant", that correlates with "dog"), combines them into image descriptions that include the correct class name ("cat"), and generates adversarial images by providing these descriptions to a text-to-image model. We explain the steps in the pipeline and provide quantitative and qualitative results.

Finding relevant spurious cues in neurons. Given two classes c_1 and c_2, we first select neurons that contribute the most toward the classification direction $v = M_{\text{text}}(c_2) - M_{\text{text}}(c_1)$, then mine their sparse decompositions for spurious cues. Specifically, we extract the set of neurons $\mathcal{N}$ whose directions are most similar to v: $\mathcal{N} = \text{top-k}_{n \in N} |\langle v, r_n^l \rangle|$. Utilizing the sparse decomposition from before, we compute a *contribution score* w_j for each phrase j in the pool T:

$$w_j^v = \sum_{n \in \mathcal{N}} \gamma_j^{l,n} \langle v, r_n^l \rangle. \tag{3.4}$$

This looks at the weight that each neuron in $\mathcal{N}$ assigns to j in its sparse decomposition, weighted by how important that neuron is for classification. A phrase with a high contribution score has significant weight in one or more important neurons, and so is a potential spurious cue. The top phrases, sorted by the contribution score are collected into a set of phrase candidates W_v.

Generating "semantic" adversarial examples. We use text and image generative models to create examples with the object c_2 that are classified as c_1. First, we generate image descriptions with a large language model (LLM) by providing it phrases from the set W^v and the class name c_1 and prompting it to generate image descriptions that include elements from both. We prompt the model to exclude anything related to c_2 from the descriptions and use visually distinctive words from W_v.

The resulting descriptions are fed into a text-to-image model to generate the adversarial images. Note that the adversarial images lie on the manifold of generated images, differently from non-semantic adversarial attacks that modify individual pixels.

Experimental settings. We generate adversarial images for classifying between pairs of classes from CIFAR-10 [47]. We use the 30k most common words as our pool T. We choose the top 100 neurons from layers 8-10 for $\mathcal{N}$, and the top 25 words according to their contribution scores for prompting the LLM. We prompt LLaMA3 [85] to generate 50 descriptions for each classification task (see prompt in Chapter B). We then filter out descriptions that include the class name and choose 10 random descriptions. We generate 10 images for each description with DeepFloyd IF text-to-image model [83]. This results in 100 images per experiment. We repeat the experiment 3 times and manually remove images that include c_2 objects or do not include c_1 objects.

We report three additional baselines. First, we repeat the same process with 100 random neurons instead of the set $\mathcal{N}$. Second, we repeat the same generation process with sparse text decompositions computed from the first principle components of the indirect effects instead of the second-order

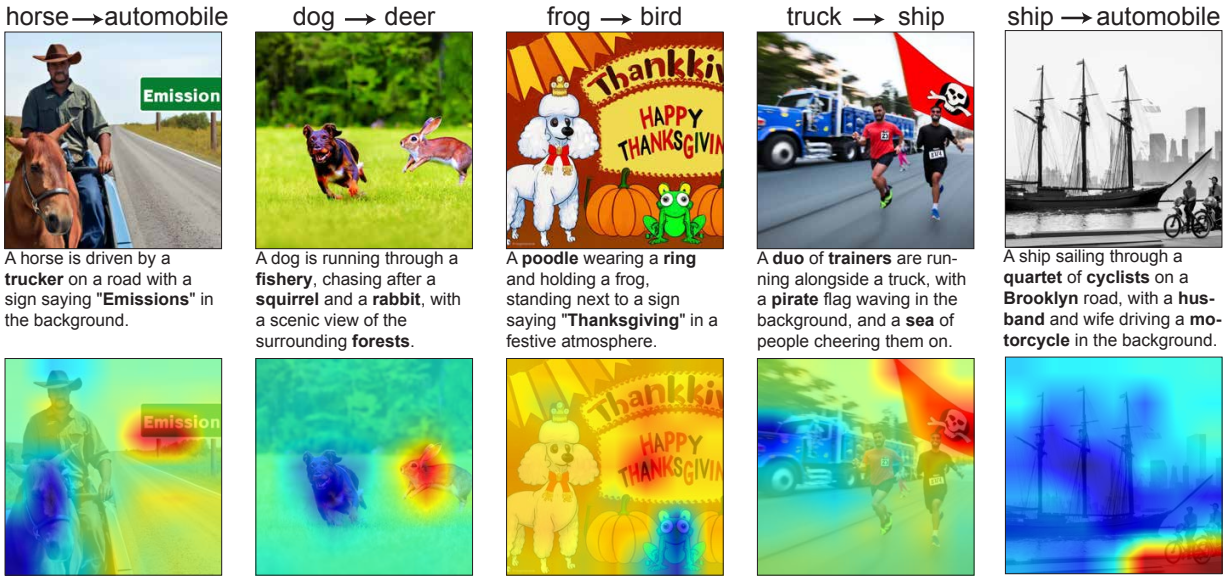

Figure 3.6: **Adversarial images generated by our method.** For each binary classification task, we present the generated images, the input text to the text-to-image model (words from W^v are bold), and an attribution map [25] for the classification (areas that contribute to the incorrect class score are red). See additional results in Chapter B.8.

effect. Third, we do not rely on the neuron decompositions, and instead prompt the language model with the words from M for which their text representations are the most similar to v. Both for our pipeline and the baselines, we automatically filter out synonyms of c_2 from the phrases provided to the language model according to their sentence similarity to c_2 [68].

Quantitative results. The classification accuracy results for the adversarial images are presented in Chapter 3.3. The success rate of our adversarial images is significantly higher than the indirect effect baseline, the similar words baseline, and the random baseline, which succeeds only accidentally. For the task of generating "ship" images the will be missclassified as "truck", no other baseline manged to generate *any* adversarial images, while ours generated 5.7 images on average.

Qualitative results. Chapter 3.6 includes generated adversarial examples and the descriptions that were used in their generation. The presented attribution heatmaps [25] show that the found spurious objects from W_v contribute the most to the misclassification, while the object from the correct class (e.g. a horse in the left-most image) contributes the least. We provide more results for additional classification tasks (e.g. "stop-sign v.s. yield") in Chapter B.8.

We show that understanding internal components in models can be grounded by exploiting them for adversarial attacks. Our attack is optimization-free and is not compute-intensive. Hence, it can be used for measuring interpretability techniques, with better understanding leading to improved attacks.

Task	Random	Indirect effect	Similar words	Second order
horse → automobile	1.0 (±1.4)	2.8 (±3.7)	1.0 (±1.4)	**5.3** (±1.9)
dog → deer	0.3 (±0.5)	6.3 (±4.8)	3.3 (±0.9)	**22.7** (±0.5)
bird → frog	0.3 (±0.5)	1.0 (±1.4)	5.0 (±2.9)	**8.0** (±4.5)
ship → truck	0.0 (±0.0)	0.0 (±0.0)	0.0 (±0.0)	**5.7** (±0.9)
ship → automobile	1.3 (±1.9)	0.0 (±0.0)	1.3 (±0.9)	**7.0** (±4.5)

Table 3.3: **Accuracy of adversarial images.** We report how many generated images out of 100, fooled the binary classifier (standard deviation in parentheses).

	Pix. Acc. ↑	mIoU ↑	mAP ↑
Partial-LRP [86]	55.0	35.5	66.9
Rollout [2]	61.8	42.6	74.0
LRP [8]	62.9	35.8	68.5
GradCAM [77]	67.3	39.3	61.9
Chefer *et al.*[12]	68.9	49.1	79.7
Raw-attention	69.6	49.8	80.0
TextSpan [25]	76.5	58.1	84.1
Ours	**78.1**	**59.0**	**84.9**

Table 3.4: **Segmentation performance on ImageNet-segmentation.** Our zero-shot segmentation is more accurate than previous methods across all metrics.

Zero-shot segmentation

Finally, we use our understanding of neurons for zero-shot segmentation. Each neuron corresponds to an attribution map, by looking at its activations $p_i^{l,n}(I)$ on each image patch. Ensembling all the neurons that contribute towards a concept results in an aggregated attribution map that can be binarized to generate reliable segmentations.

Specifically, to generate a segmentation map for an image I, we find a set of neurons with the largest absolute value of the dot product with the encoded class name c_i we aim to segment: $|\langle r_n^l, M_{\text{text}}(c_i)\rangle|$. We then average their spatial activation maps $p_i^{l,n}(I)$, standardize the average activations into $[0, 1]$, and binarize the values into foreground/background segments by applying a threshold of 0.5.

Segmentation results. We provide results on ImageNet-Segmentation [30], which includes foreground/background segmentation maps of ImageNet objects. We use activation maps from the top 200 neurons of layers 8-10. Chapter 3.4 presents a quantitative comparison to previous explainability methods. Our method outperforms other zero-shot segmentation methods across all standard evaluation metrics. We provide qualitative results before thresholding in Chapter 3.7. While the first-order effects ("TextSpan") highlight individual discriminative object parts, our

Figure 3.7: **Qualitative results on ImageNet-Segmentation (ViT-B-32).** Our heatmaps capture more object parts than the first-order token decomposition of [25].

heatmaps capture more parts of the full object.

3.6 Limitations and discussion

We analyzed the second-order effects of neurons on the CLIP representation and used our understanding to perform zero-shot segmentation and generate adversarial images. We present mechanisms that we did not analyze in our investigation and conclude with a discussion of broader impact and future directions.

Neuron-attention maps mechanisms. We investigated how the neurons flow through individual consecutive attention *values*, and ignored the effect of neurons on consecutive queries and keys in the attention mechanism. Investigating these effects will allow us to find neurons that modify the attention map patterns. We leave it for future work.

Neuron-neuron mechanisms. We did not analyze the mutual effects between neurons in the same layer or across different layers. Returning to our adversarial "frog/bird" attack example, a neuron that writes toward "dog" may not be activated if a different neuron writes simultaneously toward "frog", thus reducing our attack efficiency. While we can still generate multiple adversarial images, we believe that understanding dependencies between neurons can improve it further.

Future work and broader impact. The mass production of adversarial images can be harmful to systems that rely on neural networks (e.g., the adversarial attack that causes misclassification between "yield" and "stop sign" in Chapter B.8). Automatic extraction of such cases allows the defender to be prepared for them and, possibly, fine-tune the model on the generated images to

avoid such attacks. We plan to investigate this approach to improve CLIP's robustness in future work.

Currently, our attack pipeline relies on a few independent components, each of which has failure modes. For example, the language model can fail to generate a coherent sentence that includes *many* phrases from W_v, and can omit the class name c_2 or accidentally include the class name c_1. Additionally, the text-to-image model can fail to generate an image that follows the exact description and can drop crucial elements from the description. We believe that future improvements in the language and vision models will increase the success rate of our attack, and plan to continue to develop and improve it in the future.

Chapter 4

Rosetta Neurons: Mining the Common Units in a Model Zoo

4.1 Introduction

One of the key realizations of modern machine learning is that models trained on one task end up being useful for many other, often unrelated, tasks. This is evidenced by the success of backbone pretrained networks and self-supervised training regimes. In computer vision, the prevailing theory is that neural network models trained for various vision tasks tend to share the same concepts and structures because they are inherently present in the visual world. However, the precise nature of these shared elements and the technical mechanisms that enable their transfer remain unclear.

In this chapter, we seek to identify and match units that express similar concepts across different models. We call them *Rosetta Neurons* (see fig. 3.1). How do we find them, considering it is likely that each model would express them differently? Additionally, neural networks are usually over-parameterized, which suggests that multiple neurons can express the same concept (synonyms). The layer and channel that express the concept would also differ between models. Finally, the value of the activation is calibrated differently in each. To address these challenges, we carefully choose the matching method we use. We found that post ReLU/GeLU values tend to produce

This work was originally published as *Rosetta Neurons: Mining the Common Units in a Model Zoo* , Dravid et al, at ICCV 2023 [18]

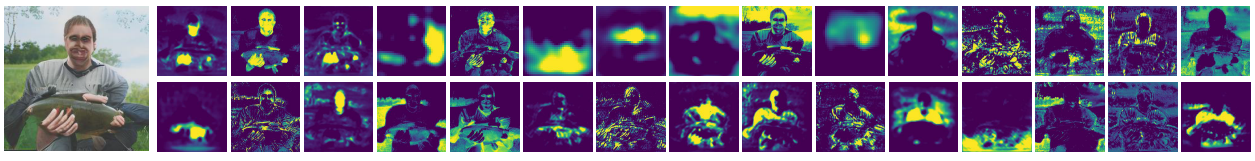

Figure 4.1: **Visualization of all the concepts for one class.** An example of the set of all concepts emerging for ImageNet "Tench" class by matching the five discriminative models from Table 4.2 and clustering within StyleGAN-XL. GAN heatmaps are visualized over one generated image.

distinct activation maps, thus these are the values we match. We compare units from different layers between the models while carefully normalizing the activation maps to overcome these differences. To address synonym neurons, we also apply our matching method on a model with itself and cluster units together according to the matches.

We search for Rosetta Neurons across eight different models: Class Supervised-ResNet50 [33], DINO-ResNet50, DINO-ViT [11], MAE [35], CLIP-ResNet50 [64], BigGAN [10], StyleGAN-2 [41], StyleGAN-XL [72]. We apply the models to the same dataset and correlate different units of different models. We mine the Rosetta neurons by clustering the highest correlations. This results in the emergence of model-free global representations, dictated by the data.

Fig. 4.1 shows an example image and all the activation maps from the discovered Rosetta Neurons. The activation maps include semantic concepts such as the person's head, hand, shirt, and fish as well as non-semantic concepts like contour, shading, and skin tone. In contrast to the celebrated work of Bau *et al*. on Network Dissection [3, 4], our method does not rely on human annotations or semantic segmentation maps. Therefore, we allow for the emergence of non-semantic concepts.

The Rosetta Neurons allow us to translate from one model's "language" to another. One particularly useful type of model-to-model translation is from discriminative models to generative models as it allows us to easily visualize the Rosetta Neurons. By applying simple transformations to the activation maps of the desired Rosetta Neurons and optimizing the generator's latent code, we demonstrate realistic edits. Additionally, we demonstrate how GAN inversion from real image to latent code improves when the optimization is guided by the Rosetta Neurons. This can be further used for out-of-distribution inversion, which performs image-to-image translation using a regular latent-to-image GAN. All of these edits usually require specialized training (e.g. [22, 39, 97]), but we leverage the Rosetta Neurons to perform them with a fixed pre-trained model.

The contributions here are as follows:

- We show the existence of Rosetta Neurons that share the same concepts across different models and training regimes.
- We develop a method for matching, normalizing, and clustering activations across models. We use this method to curate a dictionary of visual concepts.
- The Rosetta Neurons enables model-to-model translation that bridges the gap between representations in generative and discriminative models.
- We visualize the Rosetta Neurons and exploit them as handles to demonstrate manipulations to generated images that otherwise require specialized training.

The Rosetta Stone is an ancient Egyptian artifact, a large stone inscribed with the same text in three different languages. It was the key to deciphering Egyptian hieroglyphic script. The original stone is on public display at the British Museum in London.

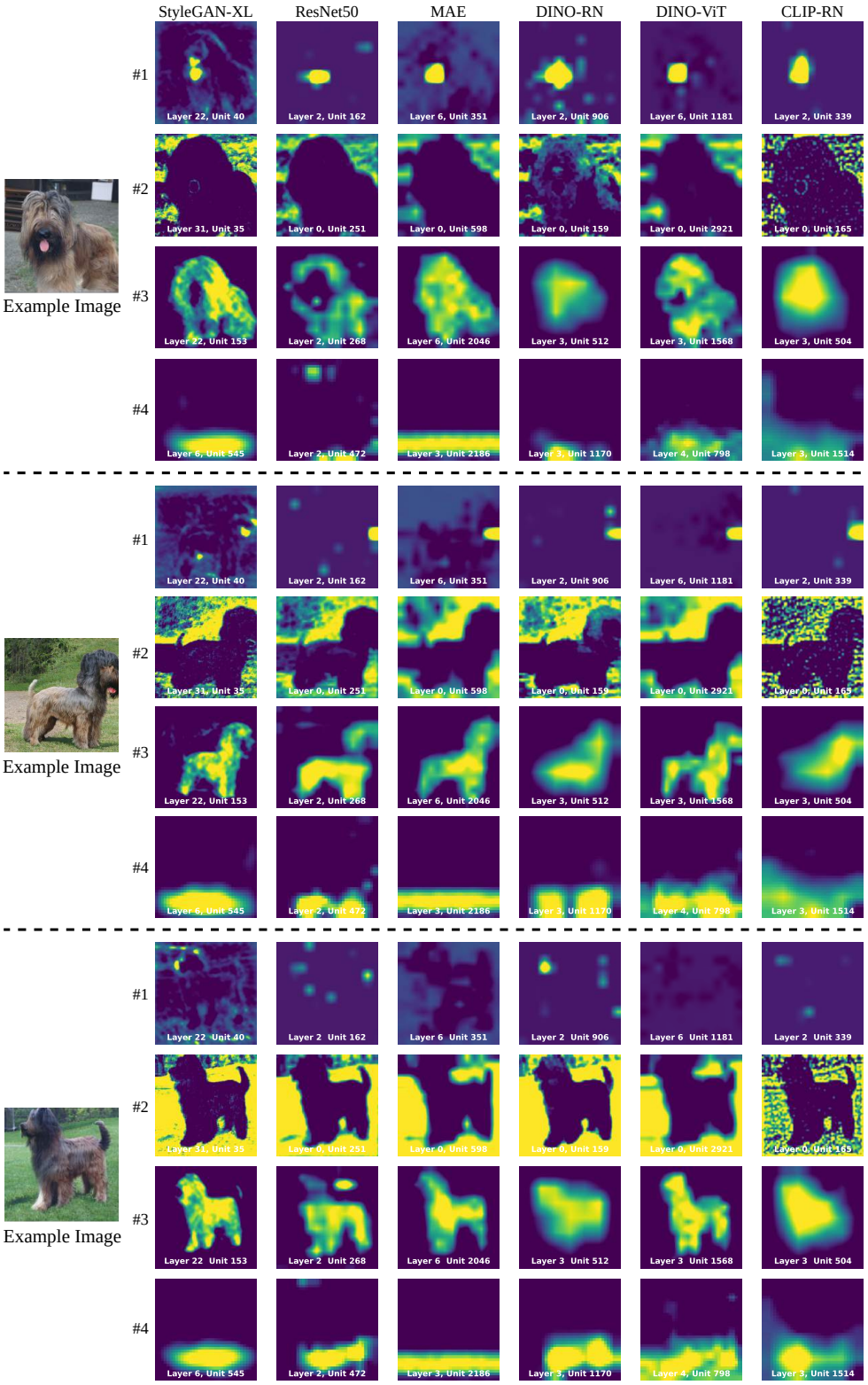

Figure 4.2: **Rosetta Neuron Dictionary.** A sample from the dictionary curated for the ImageNet class "Briard". The full dictionary can be found in the supplementary material. The figure presents 4 emergent concepts demonstrated in 3 example images. For each model, we present the normalized activation maps of the Rosetta Neuron matching the shared concept.

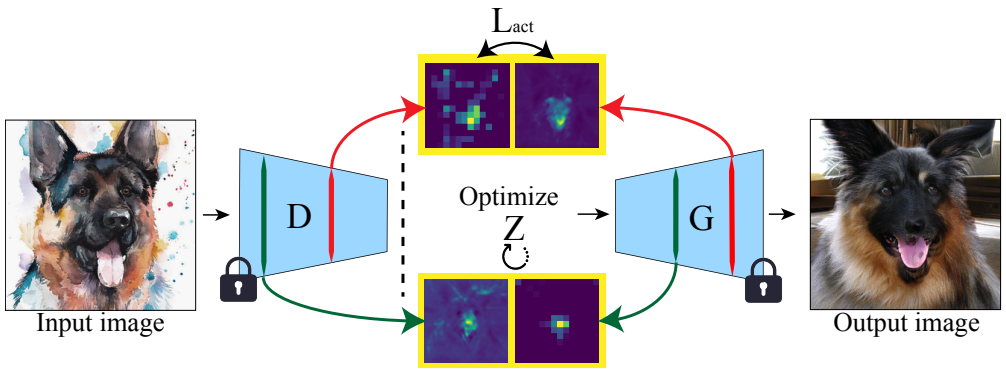

Figure 4.3: **Rosetta Neurons guided image inversion.** An input image is passed through a discriminative model D (i.e.: DINO) to obtain the Rosetta Neurons' activation maps. Then, the latent code Z of the generator is optimized to match those activation maps, according to the extracted pairs.

4.2 Related Work

Visualizing deep representations. The field of interpreting deep models has been steadily growing, and includes optimizing an image to maximize the activations of particular neurons [92, 82, 60], gradient weighted activation maps [81, 62, 67, 76], nearest neighbors of deep feature representations [48], etc. The seminal work of Bau *et al.*[4, 3] took a different approach by identifying units that have activation maps highly correlated with semantic segments in corresponding images, thereby reducing the search space of meaningful units. However, this method necessitates annotations provided by a pre-trained segmentation network or a human annotator and is confined to discovering explainable units from a predefined set of classes and in a single model. Whereas all previous works focused on analyzing a single, specific neural network model, the focus of our work is in capturing commonalities across many different networks. Furthermore, unlike [3, 4], our method does not require semantic annotation.

Explaining discriminative models with generative models. GANAlyze [28] optimized the latent code of a pre-trained GAN to find directions that affect a classifier decision. Semantic Pyramid [80] explored the subspaces of generated images to which the activations of a classifier are invariant. Lang *et al.* [49] trained a GAN to explain attributes that underlie classifier decisions. In all of these cases, the point where the generative and discriminative models communicate is in the one "language" they both speak - pixels; which is the output of the former and an input of the latter. Our method for bridging this gap takes a more straightforward approach: we directly match neurons from pre-trained networks and identify correspondences between their internal activations. Moreover, as opposed to [49] and [80], our method does not require GAN training and can be applied to any off-the-shelf GAN and discriminative model.

Analyzing representation similarities in neural networks. Our work is inspired by the neuroscience literature on representational similarity analysis [45, 19] that aims to extract correspondences between different brain areas [32], species [46], individual subjects [13], and between neural networks and brain neural activities [89]. On the computational side, Kornblith *et al.* [44]

aimed to quantify the similarities between different layers of discriminative convolutional neural networks, focusing on identifying and preserving invariances. Esser, Rombach, and Ommer [23, 69] trained an invertible network to translate non-local concepts, expressed by a latent variable, across models. In contrast, our findings reveal that individual neurons hold shared concepts across a range of models and training regimes without the need to train a specialized network for translation. This leads to another important difference: the concepts we discover are local and have different responses for different spatial locations in an image. We can visualize these responses and gain insights into how these concepts are represented in the network.

4.3 Method

Our goal is to find Rosetta Neurons across a variety of models. We define Rosetta Neurons as two (or more) neurons in different models whose activations (outputs) are positively correlated over a set of many inputs. Below we explain how to find Rosetta Neurons across a variety of models and describe how to merge similar Rosetta Neurons into clusters that represent the same concepts.

Mining common units in two models

Preliminaries. Given two models $F^{(1)}, F^{(2)}$, we run n inputs through both models. For discriminative models, this means a set of images $\{I_i\}_{i=1}^n$. If one of the models is generative, we first sample n random input noises $\{Z_i\}_{i=1}^n$ and generate images $I_i = F^{(1)}(z_i)$ that will be the set of inputs to the discriminative model $F^{(2)}$. We denote the set of extracted activation maps of F by F^{act}. The size $|F^{act}|$ is the total number of channels in all the layers. The j-th intermediate activation map of F when applied to the i-th input is then F_i^j. That is $F_i^j = F^j(I_i)$ for a discriminative model and $F_i^j = F^j(z_i)$ for a generative one.

Comparing activation maps. To compare units $F^{(1)j}$ and $F^{(2)k}$, namely, the j-th unit from the first model with the k-th unit from the second one, we first bilinearly interpolate the feature maps to have the same spatial dimensions according to the maximum of the two map sizes. Our approach to perform matching is based on correlation, similar to [45], but taken across both data instances and spatial dimensions. We then take the mean and variance across the n images and across the spatial dimensions of the images, where x combines both spatial dimensions of the images.

$$\begin{aligned} \overline{F^j} &= \frac{1}{nm^2} \sum_{i,x} F_{i,x}^j \\ var(F^j) &= \frac{1}{nm^2 - 1} \sum_{i,x} \left(F_{i,x}^j - \overline{F^j} \right)^2 \end{aligned} \tag{4.1}$$

Next, the measure of distance between two units is calculated by Pearson correlation:

$$d(F^{(1)j}, F^{(2)k}) = \frac{\sum_{i,x} \left(F_{i,x}^{(1)j} - \overline{F^{(1)j}} \right) \left(F_{i,x}^{(2)k} - \overline{F^{(2)k}} \right)}{\sqrt{var(F^{(1)j}) \cdot var(F^{(2)k})}} \tag{4.2}$$

In our experiments, this matching is computed between a generative model G and a discriminative model D. The images used for D are generated by G applied to n sampled noises.

Filtering "best buddies" pairs. To detect reliable matches between activation maps, we keep the pairs that are mutual nearest neighbors (named "best-buddies" pairs by [14]) according to our distance metric and filter out any other pair. Formally, our set of "best buddies" pairs is:

$$\begin{aligned} BB(F^{(1)}, F^{(2)}; K) = \{(j,k)| \\ F^{(1)k} \in KNN(F^{(2)j}, F^{(1)act}; K) \\ \wedge \ F^{(2)j} \in KNN(F^{(1)k}, F^{(2)act}; K)\} \end{aligned} \tag{4.3}$$

Where $KNN(F^{(a)j}, F^{(b)act})$ is the set of the K-nearest neighbors of the unit j from model $F^{(a)}$ among all the units in model $F^{(b)}$: beginequation*

As shown in [14], the probability of being mutual nearest neighbors is maximized when the neighbors are drawn from the same distribution. Thus, keeping the "best buddies" discards noisy matches.

Extracting common units in m models

Merging units between different models. To find similar activation maps across many different discriminative models $D_i, i \in [m]$, we merge the "best buddies" pairs calculated between D_i and a generator G for all the i's. Formally, our Rosetta units are:

$$R(G, D_1 ... D_m) = \{(j, k_1, ..., k_m) | \forall i : (j, k_i) \in BB(G, D_i)\} \tag{4.4}$$

This set of tuples includes the "translations" between similar neurons across all the models. Note that when $m = 1$, $R(G, D_1) = BB(G, D_1)$.

Clustering similar units into concepts. Empirically, the set of Rosetta units includes a few units that have similar activation maps for the n images. For instance, multiple units may be responsible for edges or concepts such as "face." We cluster them according to the self "best-buddies" of the generative model, defined by $BB(G, G; K)$. We set two Rosetta Neurons in R to belong to the same cluster if their corresponding units in G are in $BB(G, G; K)$.

Curating a dictionary. After extracting matching units for a dataset across a model zoo, we enumerate the sets of matching Rosetta Neurons in the clustered R. Fig. 4.2 is a sample from such a dictionary. Fig. 4.1 shows a list of all the concepts for a single image. Since the concepts emerge and are not related to human annotated labels, we simply enumerate them and present each concept on several example images to visually identify it. Using 1600 instances generated by the GAN, Distances are taken between all possible bipartite pairs of units, the $K = 5$ nearest neighbors are extracted, from which Best-Buddies are filtered. Typically for the datasets and models we experimented with, around 50 concepts emerge. The exact list of models used in our experiments and the datasets they were trained on can be found in Table. 4.2. See supplementary material for the dictionaries.

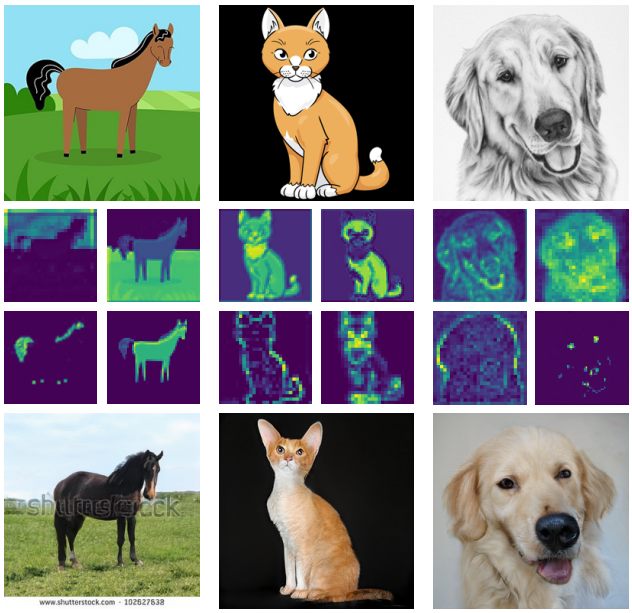

Figure 4.4: **Out-of-distribution inversions**. By incorporating the Rosetta Neurons in the image inversion process, we can invert sketches and cartoons (first row), and generate similar in-distribution images (last row). A subset of the Rosetta Neurons from the input images that were matched during the inversion process is shown in the middle rows.

4.4 Visualizing the Rosetta Neurons

As we involve a generative model in the Rosetta Neurons mining procedure, we can utilize it for visualizing the discovered neurons as well. In this section, we present how to visualize the neurons via a lightweight matches-guided inversion technique. We then present how direct edits of the activation maps of the neurons can translate into a variety of generative edits in the image space, without any generator modification or re-training.

Rosetta Neurons-Guided Inversion

To visualize the extracted Rosetta Neurons, we take inspiration from [80], and use the generative model G to produce images for which the generator activation maps of the Rosetta Neurons best match to the paired activation maps extracted from $D(I_v)$, as shown in figure 4.3. As opposed to [80], we do not train the generative model to be conditioned on the activation maps. Instead, we invert images through the fixed generator into some latent code z, while maximizing the similarity between the activation maps of the paired Rosetta Neurons. Our objective is:

$$\arg\min_{z}(-L_{act}(z, I_v) + \alpha L_{reg}(z)) \tag{4.5}$$

Where α is a loss coefficient, L_{reg} is a regularization term (L_2 or L_1), and $L_{act}(z, I_v)$ is the mean of normalized similarities between the paired activations:

$$L_{act}(z, I_v) = \frac{1}{|BB(G,D)|} \sum_{\substack{(j,k)\in \\ BB(G,D)}} \frac{\sum_x \left(G_x^j - \overline{G^j}\right)\left(D_x^k - \overline{D^k}\right)}{\sqrt{var(G^j) \cdot var(D^k)}} \tag{4.6}$$

Where G^j is the j-th activation map of $G(z)$ and D^k is the k-th activation map of $D(I_v)$. For obtaining this loss, we use the mean and variance precomputed by Eq. 4.1 over the entire dataset during the earlier mining phase. However, we calculate the correlation over the spatial dimensions of a single data instance.

The Rosetta neurons guided inversion has two typical modes. The first mode is when both the initial activation map and the target one have some intensity somewhere in the map (e.g. two activation maps that are corresponding to "nose" are activated in different spacial locations). In this case, the visual effect is an alignment between the two activation maps. As many of the Rosetta neurons capture object parts, it results in image-to-image alignment (e.g., fig. 4.5). The second mode is when either the target or the initial activation map is not activated. In this case, a concept will appear or disappear (e.g., fig. 4.8).

Visualizing a single Rosetta Neuron. We can visualize a single Rosetta Neuron by modifying the loss in our inversion process (eq. 4.6). Rather than calculating the sum over the entire set of Rosetta Neurons, we do it for a single pair that corresponds to the specific Rosetta neuron. When this optimization procedure is applied a few times on the same input neuron pair starting from a few different randomly initialized latent codes, we get a diverse set of images that are matching to the same activation map of the wanted Rosetta Neuron. This allows a user to disentangle and detect what is the concept that is specifically represented by the given neuron. Figure 3.1 present two optimized images for each of the presented Rosetta Neurons. This visualization allows the viewer to see that Concept #1 corresponds to the concept "red color," rather than to the concept "hat."

Inverting out-of-distribution images. The inversion process presented above does not use the generated image in the optimization, as opposed to common inversion techniques that calculate the pixel loss or perceptual loss between the generated image the input image. Our optimization process does not compare the image pixel values, and as many of the Rosetta Neurons capture high-level semantic concepts and coarse structure of the image, this allows us to invert images outside of the training distribution of the generative model. Figure 4.5 presents a cross-class image-to-image translation that is achieved by Rosetta Neurons guided inversion. As shown, the pose of the input images of dogs is transferred to the poses of the optimized cat images, as the Rosetta Neurons include concepts such as "nose," "ears," and "contour" (please refer to Figure 3.1 for a subset of the Rosetta Neurons for this set of models).

Figure 4.4 presents the inversion results for sketches and cartoons, and a subset of the Rosetta Neurons that were used for optimization. As shown, the matches-guided inversion allows us to "translate" between the two domains via the shared Rosetta Neurons and preserve the scene layout

Figure 4.5: **Cross-class image-to-image translation.** Rosetta Neurons guided inversion of input images (top row) into a StyleGAN2 trained on LSUN cats [90], allows us to preserve the pose of the animal while changing it from dog to cat (bottom row). See supplementary material for more examples.

and object pose. Our lightweight method does not require dedicated models or model training, as opposed to [97, 39].

Inverting in-distribution images. We found that adding the loss term in eq. 4.5 to the simple reconstruction loss objective improves the inversion quality. Specifically, we optimize:

$$\arg\min_{z}(L_{rec}(G(z), I_v) + \alpha L_{reg}(z) - \beta L_{act}(z, I_v)) \tag{4.7}$$

Where L_{rec} is the reconstruction loss between the generated image and the input image, and β is a loss coefficient. The reconstruction loss can be pixel loss, such as L_1 or L_2 between the two images, or a perceptual loss.

We compare the inversion quality with and without the Rosetta Neurons guidance and present the PSNR, SSIM, and LPIPS [94] for StyleGAN-XL inversion. We use solely a perceptual loss as a baseline, similarly to [72]. We add our loss term to the optimization, where the Rosetta Neurons are calculated from 3 sets of matches with StyleGAN-XL: matching to DINO-RN, matching to CLIP-RN, and matching across all the discriminative models in Table 4.2. We use the same hyperparameters as in [72], and set $\alpha = 0.1$ and $\beta = 1$.

Table 4.1 presents the quantitative inversion results for 5000 randomly sampled images from the ImageNet validation set (10% of the validation set, 5 images per class), as done in [72]. Figure 4.6 presents the inversion results for the baseline and for the additional Rosetta Neurons guidance using the matches between all the models. As shown qualitatively and quantitatively, the inversion quality improves when the Rosetta Neurons guiding is added. We hypothesize this is due to the optimization objective that directly guides the early layers of the generator and adds layout constraints. These soft constraints reduce the optimization search space and avoid convergence to local minima with low similarity to the input image.

	PSNR ↑	SSIM ↑	LPIPS ↓
Perceptual loss	13.99	0.340	0.48
+DINO matches	15.06	0.360	0.45
+CLIP matches	15.20	0.362	**0.44**
+All matches	**15.42**	**0.365**	0.46

Table 4.1: **Inversion quality on ImageNet.** We compare the inversion quality for StyleGAN-XL when Rosetta Neurons guidance is added, for 3 sets of matches - StyleGAN-XL & DINO-RN, StyleGAN-XL & CLIP-RN and all the models from figure 4.2.

Model	Training dataset	Resolution
StyleGAN-XL	ImageNet	256
StyleGAN2	LSUN(cat)	256
StyleGAN2	LSUN(horse)	512
BigGAN	ImageNet	256
ResNet50	ImageNet	224
DINO-ResNet50	ImageNet	224
DINO-VIT-base	ImageNet	224
MAE-base	ImageNet	224
CLIP	WebImageText	224

Table 4.2: **Models used in the paper.**

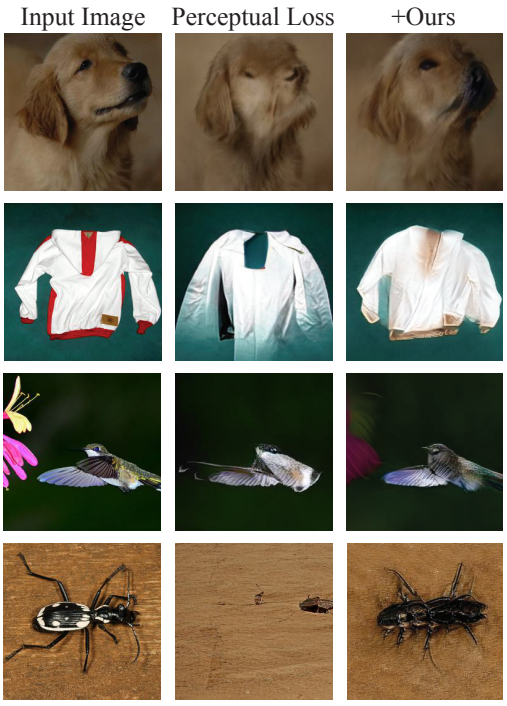

Figure 4.6: **Image inversions for StyleGAN-XL.** We compare inversions by optimizing perceptual loss only (second column), to additional Rosetta Neurons guidance loss, with matches calculated across all the models presented in Figure 4.2 (third column). See supplementary material for more examples.

Rosetta Neurons Guided Editing

The set of Rosetta Neurons allows us to apply controlled edits on a generated image $I_{src} = G(z)$ and thus provides a counterfactual explanation to the neurons. Specifically, we modify the activation maps corresponding to the Rosetta Neurons, extracted from $G(z)$, and re-optimize the latent code to match the edited activation maps according to the same optimization objective presented in eq. 4.5. As opposed to previous methods like [22], which trained a specifically designed generator to allow disentangled manipulation of objects at test-time, we use a fixed generator and only optimize the latent representation. Next, we describe the different manipulations that can be done on the activation maps, before re-optimizing the latent code:

Zoom-in. We double the size of each activation map that corresponds to a Rosetta Neurons with bilinear interpolation and crop the central crop to return to the original activation map size. We start our re-optimization from the same latent code that generated the original image.

Shift. To shift the image, we shift the activation maps directly and pad them with zeros. The shift stride is relative to the activation map size (e.g. we shift a 4×4 activation map by 1, while shifting 8×8 activation maps by 2).

Copy & paste. We shift the activation maps into two directions (e.g. left and right), creating two sets of activation maps - left map, and right map. We merge them by copying and pasting the left half of the left activation map and the right half of the right activation map. We found that starting from random z rather than z that generated the original image obtains better results.

Figure 4.7 shows the different image edits that are done via latent optimization to match the manipulated Rosetta Neurons. We apply the edits for two different generative models (BigGAN and StyleGAN2) to show the robustness of the method to different architectures.

Fine-grained Rosetta Neurons edit. Our optimization procedure allows us to manipulate a subset of the Rosetta Neurons, instead of editing all of the neurons together. Specifically, we can manually find among the Rosetta Neurons a few that correspond to elements in the image that we wish to modify. We create "ground truth" activations by modifying them manually and re-optimizing the latent code to match them. For example - to remove concepts specified by Rosetta Neurons, we set their values to the minimal value in their activation maps. We start our optimization from the latent that corresponds to the input image and optimize until the picked activation maps converge to the manually edited activation maps. Figure 4.8 presents examples of removed Rosetta Neurons. Modifying only a few activation maps (1 or 2 in the presented images) that correspond to the objects we aimed to remove, allows us to apply realistic manipulations in the image space. As opposed to [3], we do not rewrite the units in the GAN directly and apply optimization instead, as we found that direct edits create artifacts in the generated image for large and diverse GANs.

Implementation details. For the re-optimization step, we train z for 500 steps, with Adam optimizer [43] and a learning rate of 0.1 for StyleGAN2 and 0.01 for BigGAN. Following [72], the learning rate is ramped up from zero linearly during the first 5% of the iterations and ramped down to zero using a cosine schedule during the last 25% of the iterations. We use $K = 5$ for calculating the nearest neighbors. The inversion and inversion-based editing take less than 5 minutes per image on one A100 GPU.

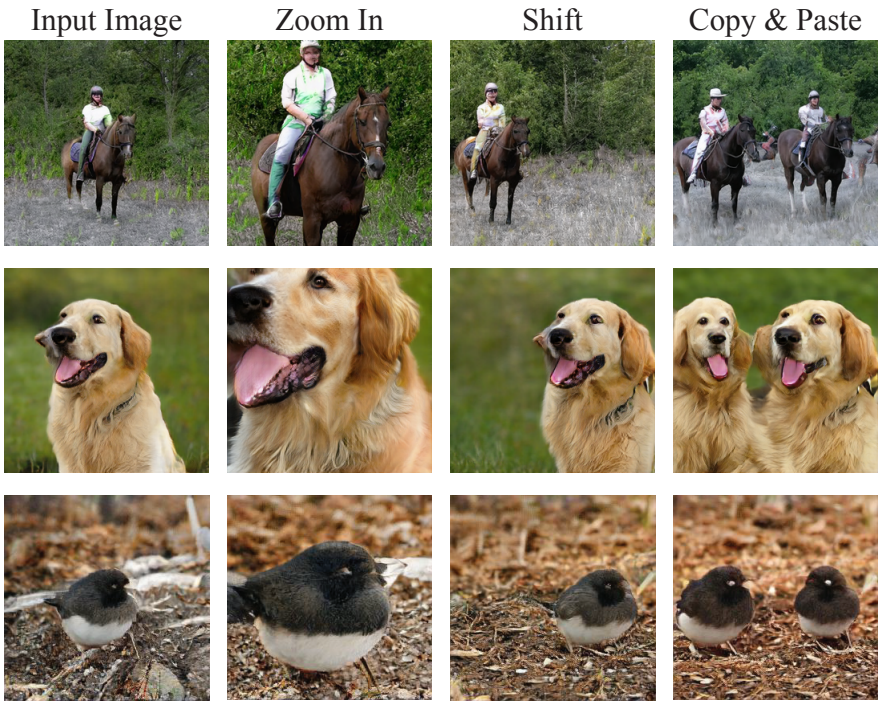

Figure 4.7: **Rosetta Neurons guided editing**. Direct manipulations on the activation maps corresponding to the Rosetta neurons are translated to manipulations in the image space. We use two models (top row - StyleGAN2, bottom two rows - BigGAN) and utilize the matches between each of them to DINO-RN.

Figure 4.8: **Single Rosetta Neurons Edits.** We optimize the latent input s.t. the value of a desired Rosetta activation reduces. This allows removing elements from the image (e.g. emptying the beer in the glass, reducing the water stream in the fountain, and removing food from a plate). See appendix for more examples.

4.5 Limitations

Our method can not calculate GAN-GAN matches directly, only through a discriminative model. Unlike discriminative models that can receive the same input image, making two GANs generate the same image is not straightforward. Consequently, we only match GANs with discriminative models.

Secondly, we were unsuccessful when applying our approach to diffusion models, such as [70]. We speculate that this is due to the autoregressive nature of diffusion models, where each step is a conditional generative model from image to image. We hypothesize that as a result, the noisy image input is a stronger signal in determining the outcome of each step, rather than a specific unit. Thus, the units in diffusion models have more of an enhancing or editing role, rather than a generating role, which makes it less likely to identify a designated perceptual neuron.

Lastly, our method relies on correlations, and therefore there is a risk of mining spurious correlations. As shown in Figure 4.2, the dog in the third example does not have its tongue visible, yet both StyleGAN-XL and DINO-RN activated for Concept #1 in a location where the tongue would typically be found. This may be due to the correlation between the presence of a tongue and the contextual information where it usually occurs.

4.6 Conclusion

We introduced a new method for mining and visualizing common representations that emerge in different visual models. Our results demonstrate the existence of specific units that represent the same concepts in a diverse set of deep neural networks, and how they can be utilized for various generative tasks via a lightweight latent optimization process. We believe that the found common neurons can be used in a variety of additional tasks, including image retrieval tasks and more advanced generative tasks. Additionally, we hope that the extracted representations will shed light on the similarities and dissimilarities between models that are trained for different tasks and with different architectures. We plan to explore this direction in future work.

Chapter 5

Discussion and Future Work

This thesis presented the first steps in understanding the computation inside pre-trained deep vision models. First, I showed how different components in one model, CLIP, can be described automatically. It was achieved by analyzing specific information flows from these components to the output space and using the fact that the output space can be interpreted with text representations. Then, I showed that some components in other models, that were trained for different tasks and data, are sharing the same functionality. The fact that we can automatically interpret CLIP components and some of them are similar to components in other models paves a way for automatically interpreting other models as well.

Next, I present my future research plans, ranging from the near future – automating and extending the presented approaches for interpretability, to longer horizon goals – accelerating scientific discovery.

Interpreting and reverse-engineering deep neural networks still require manual analysis of different model components and their interactions. Given a neural network, my ultimate goal is to automate this reverse-engineering process and to extract a description of a minimal human-interpretable circuit for specific sub-tasks. Combining the interpretation approach for specific flows in the model, and iterative hypothesis generation (that becomes more plausible with large generative language and vision models), together with testing and refinement, can result in a scalable understanding of model circuits.

I believe that the presented approach for doing *AI science* (e.g., scalable understanding of model circuits) will be useful in designing future AI systems. Reasoning about the limitations of different circuits and the shortcomings of the existing deep learning frameworks can lead to improved training recipes, architectures, and model safety.

A scalable understanding of model behavior can be useful not only for designing better models but also for automated scientific discovery. Current deep neural networks that are trained on large amounts of data already manage to outperform humans on some tasks. In the future, these models will continue to improve and solve new tasks that humans do not know how to solve. Extracting the underlying mechanisms that these models learn for solving new problems, and abstracting them to a human language, can increase the rate of new scientific discoveries. The approach for automated hypotheses generation, presented earlier, can be applied to explain the steps for solving a task that

humans don't know how to solve. Looking forward, I will aim to automatically lift the computation in models into an algorithmic level of abstraction that humans can understand, to explain previously unknown phenomena.

Bibliography

[1] Kfir Aberman et al. “Deep saliency prior for reducing visual distraction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 19851–19860.

[2] Samira Abnar and Willem Zuidema. “Quantifying Attention Flow in Transformers”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4190–4197. DOI: 10.18653/v1/2020.acl-main.385. URL: https://aclanthology.org/2020.acl-main.385.

[3] David Bau et al. “GAN Dissection: Visualizing and Understanding Generative Adversarial Networks”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[4] David Bau et al. “Network Dissection: Quantifying Interpretability of Deep Visual Representations”. In: *Computer Vision and Pattern Recognition*. 2017.

[5] David Bau et al. “Understanding the role of individual units in a deep neural network”. In: *Proceedings of the National Academy of Sciences* (2020). ISSN: 0027-8424. DOI: 10.1073/pnas.1907375117. URL: https://www.pnas.org/content/early/2020/08/31/1907375117.

[6] Usha Bhalla et al. *Interpreting CLIP with Sparse Linear Concept Embeddings (SpLiCE)*. 2024. arXiv: 2402.10376 [cs.LG].

[7] Steven Bills et al. *Language models can explain neurons in language models*. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html. 2023.

[8] Alexander Binder et al. “Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers”. In: vol. 9887. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2016, pp. 63–71. DOI: 10.1007/978-3-319-44781-0_8.

[9] Trenton Bricken et al. “Towards Monosemanticity: Decomposing Language Models With Dictionary Learning”. In: *Transformer Circuits Thread* (2023). URL: https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[10] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=B1xsqj09Fm.

[11] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021.

[12] Hila Chefer, Shir Gur, and Lior Wolf. "Transformer Interpretability Beyond Attention Visualization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 782–791.

[13] Andrew C. Connolly et al. "The Representation of Biological Classes in the Human Brain". In: *The Journal of Neuroscience* 32 (2012), pp. 2608–2618.

[14] Tali Dekel et al. "Best-Buddies Similarity for Roboust Template Matching". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2021–2029.

[15] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[16] Alexey Dosovitskiy and Thomas Brox. "Inverting Convolutional Networks with Convolutional Networks". In: *CoRR* abs/1506.02753 (2015). arXiv: 1506.02753. URL: http://arxiv.org/abs/1506.02753.

[17] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[18] Amil Dravid et al. "Rosetta Neurons: Mining the Common Units in a Model Zoo". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 1934–1943.

[19] Shimon Edelman. "Representation is representation of similarities". In: *Behavioral and Brain Sciences* 21.4 (1998), pp. 449–467. DOI: 10.1017/S0140525X98001253.

[20] Nelson Elhage et al. "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread* (2021). https://transformer-circuits.pub/2021/framework/index.html.

[21] Nelson Elhage et al. "Toy Models of Superposition". In: *Transformer Circuits Thread* (2022). URL: https://transformer-circuits.pub/2022/toy_model/index.html.

[22] Dave Epstein et al. "BlobGAN: Spatially Disentangled Scene Representations". In: *European Conference on Computer Vision (ECCV)* (2022).

[23] Patrick Esser, Robin Rombach, and Björn Ommer. "A Disentangling Invertible Interpretation Network for Explaining Latent Representations". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 9220–9229.

[24] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. "Interpreting the Second-Order Effects of Neurons in CLIP". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=GPDcvoFGOL.

[25] Yossi Gandelsman, Alexei A. Efros, and Jacob Steinhardt. "Interpreting CLIP's Image Representation via Text-Based Decomposition". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=5Ca9sSzuDp.

[26] Mor Geva et al. *Transformer Feed-Forward Layers Are Key-Value Memories*. 2021. arXiv: 2012.14913 [cs.CL].

[27] Lore Goetschalckx et al. *GANalyze: Toward Visual Definitions of Cognitive Image Properties*. 2019. arXiv: 1906.10112 [cs.CV].

[28] Lore Goetschalckx et al. "GANalyze: Toward Visual Definitions of Cognitive Image Properties". In: *arXiv preprint arXiv:1906.10112* (2019).

[29] Gabriel Goh et al. "Multimodal Neurons in Artificial Neural Networks". In: *Distill* (2021). https://distill.pub/2021/multimodal-neurons. DOI: 10.23915/distill.00030.

[30] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. "ImageNet Auto-Annotation with Segmentation Propagation". In: *Int. J. Comput. Vision* 110.3 (Dec. 2014), pp. 328–348. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0713-9. URL: https://doi.org/10.1007/s11263-014-0713-9.

[31] Wes Gurnee et al. *Universal Neurons in GPT2 Language Models*. 2024. arXiv: 2401.12181 [cs.LG].

[32] James Haxby et al. "Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex". In: *Science (New York, N.Y.)* 293 (Oct. 2001), pp. 2425–30. DOI: 10.1126/science.1063736.

[33] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778.

[34] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[35] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *arXiv:2111.06377* (2021).

[36] Dan Hendrycks et al. "The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization". In: *ICCV* (2021).

[37] Evan Hernandez et al. "Natural Language Descriptions of Deep Visual Features". In: *CoRR* abs/2201.11114 (2022). arXiv: 2201.11114. URL: https://arxiv.org/abs/2201.11114.

[38] Gabriel Ilharco et al. *OpenCLIP*. Version 0.1. 2021. DOI: 10.5281/zenodo.5143773. URL: https://doi.org/10.5281/zenodo.5143773.

[39] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. 2017.

[40] Chao Jia et al. “Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision”. In: *International Conference on Machine Learning*. 2021. URL: https://api.semanticscholar.org/CorpusID:231879586.

[41] Tero Karras et al. “Analyzing and Improving the Image Quality of StyleGAN”. In: *Proc. CVPR*. 2020.

[42] Justin Kerr et al. *LERF: Language Embedded Radiance Fields*. 2023. arXiv: 2303.09553 [cs.CV].

[43] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[44] Simon Kornblith et al. “Similarity of Neural Network Representations Revisited”. In: *ArXiv* abs/1905.00414 (2019).

[45] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. “Representational similarity analysis - connecting the branches of systems neuroscience”. In: *Frontiers in Systems Neuroscience* 2 (2008). ISSN: 1662-5137. DOI: 10.3389/neuro.06.004.2008. URL: https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008.

[46] Nikolaus Kriegeskorte et al. “Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey”. In: *Neuron* 60 (2008), pp. 1126–1141.

[47] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. 2009.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60 (2012), pp. 84–90.

[49] Oran Lang et al. “Explaining in Style: Training a GAN to explain a classifier in StyleSpace”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 673–682.

[50] Yanghao Li et al. *Scaling Language-Image Pre-training via Masking*. 2023. arXiv: 2212.00794 [cs.CV].

[51] Timo Lüddecke and Alexander Ecker. “Image Segmentation Using Text and Image Prompts”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 7086–7096.

[52] Scott M. Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *CoRR* abs/1705.07874 (2017). arXiv: 1705.07874. URL: http://arxiv.org/abs/1705.07874.

[53] Aravindh Mahendran and Andrea Vedaldi. “Understanding Deep Image Representations by Inverting Them”. In: *CoRR* abs/1412.0035 (2014). arXiv: 1412.0035. URL: http://arxiv.org/abs/1412.0035.

[54] Joanna Materzynska, Antonio Torralba, and David Bau. *Disentangling visual and written concepts in CLIP*. 2022. arXiv: 2206.07835 [cs.CV].

[55] Thomas McGrath et al. *The Hydra Effect: Emergent Self-repair in Language Model Computations*. 2023. arXiv: 2307.15771 [cs.LG].

[56] Kevin Meng et al. "Locating and Editing Factual Associations in GPT". In: *Advances in Neural Information Processing Systems* 36 (2022). arXiv:2202.05262.

[57] Neel Nanda et al. *Progress measures for grokking via mechanistic interpretability*. 2023. arXiv: 2301.05217 [cs.LG].

[58] nostalgebraist. *interpreting GPT: the logit lens*. 2020. URL: https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens (visited on 08/30/2020).

[59] Tuomas Oikarinen and Tsui-Wei Weng. *CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks*. 2023. arXiv: 2204.10965 [cs.CV].

[60] Chris Olah et al. "Zoom In: An Introduction to Circuits". In: *Distill* (2020). DOI: 10.23915/distill.00024.001.

[61] Yagyensh C. Pati, Ramin Rezaiifar, and Perinkulam S. Krishnaprasad. "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition". In: *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers* (1993), 40–44 vol.1. URL: https://api.semanticscholar.org/CorpusID:16513805.

[62] Vitali Petsiuk, Abir Das, and Kate Saenko. "RISE: Randomized Input Sampling for Explanation of Black-box Models". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2018.

[63] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. *Learning to Generate Reviews and Discovering Sentiment*. 2017. arXiv: 1704.01444 [cs.LG].

[64] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html.

[65] Senthooran Rajamanoharan et al. *Improving Dictionary Learning with Gated Sparse Autoencoders*. 2024. arXiv: 2404.16014 [cs.LG].

[66] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *CoRR* abs/2102.12092 (2021). arXiv: 2102.12092. URL: https://arxiv.org/abs/2102.12092.

[67] Sylvestre-Alvise Rebuffi et al. "There and Back Again: Revisiting Backpropagation Saliency Methods". In: *CoRR* abs/2004.02866 (2020). arXiv: 2004.02866. URL: https://arxiv.org/abs/2004.02866.

[68] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084.

[69] Robin Rombach, Patrick Esser, and Björn Ommer. "Network-to-Network Translation with Conditional Invertible Neural Networks". In: *arXiv: Computer Vision and Pattern Recognition* (2020).

[70] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[71] Shiori Sagawa et al. "Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization". In: *CoRR* abs/1911.08731 (2019). arXiv: 1911.08731. URL: http://arxiv.org/abs/1911.08731.

[72] Axel Sauer, Katja Schwarz, and Andreas Geiger. "StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets". In: *ACM SIGGRAPH 2022 Conference Proceedings*. SIGGRAPH '22. Vancouver, BC, Canada: Association for Computing Machinery, 2022. ISBN: 9781450393379. DOI: 10.1145/3528233.3530738. URL: https://doi.org/10.1145/3528233.3530738.

[73] Ludwig Schubert et al. "High-Low Frequency Detectors". In: *Distill* (2021). DOI: 10.23915/distill.00024.005.

[74] Christoph Schuhmann et al. "LAION-5B: An open large-scale dataset for training next generation image-text models". In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022. URL: https://openreview.net/forum?id=M3Y74vmsMcY.

[75] Toby Segaran and Jeff Hammerbacher, eds. *Beautiful Data: The Stories Behind Elegant Data Solutions*. Beijing: O'Reilly, 2009. ISBN: 978-0-596-15711-1.

[76] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128 (2016), pp. 336–359.

[77] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization." In: *ICCV*. IEEE Computer Society, 2017, pp. 618–626. ISBN: 978-1-5386-1032-9. URL: http://dblp.uni-trier.de/db/conf/iccv/iccv2017.html#SelvarajuCDVPB17.

[78] Harshay Shah, Andrew Ilyas, and Aleksander Madry. "Decomposing and Editing Predictions by Modeling Model Computation". In: *arXiv preprint arXiv:2404.11534* (2024).

[79] Tamar Rott Shaham et al. *A Multimodal Automated Interpretability Agent*. 2024. arXiv: 2404.14394 [cs.AI].

[80] Assaf Shocher et al. “Semantic Pyramid for Image Generation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 7455–7464.

[81] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” In: *CoRR* abs/1312.6034 (2013). URL: http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SimonyanVZ13.

[82] Jost Tobias Springenberg et al. “Striving for simplicity: The all convolutional net”. In: *arXiv preprint arXiv:1412.6806* (2014).

[83] StabilityAI. *DeepFloyd-IF*. https://github.com/deep-floyd/IF. 2023.

[84] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *CoRR* abs/1703.01365 (2017). arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

[85] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[86] Elena Voita et al. “Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5797–5808. DOI: 10.18653/v1/P19-1580. URL: https://aclanthology.org/P19-1580.

[87] Peter Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-201. Caltech, Jan. 1, 2010. URL: /se3/wp-content/uploads/2014/09/WelinderEtal10_CUB-200.pdf,%20http://www.vision.caltech.edu/visipedia/CUB-200.html.

[88] Mitchell Wortsman. *Reaching 80% Zero-Shot Accuracy with OpenCLIP: ViT-G/14 Trained on LAION-2B*. 2023. URL: https://laion.ai/blog/giant-openclip/.

[89] Daniel Yamins et al. “Performance-optimized hierarchical models predict neural responses in higher visual cortex”. In: *Proceedings of the National Academy of Sciences of the United States of America* 111 (May 2014). DOI: 10.1073/pnas.1403112111.

[90] Fisher Yu et al. “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop”. In: *arXiv preprint arXiv:1506.03365* (2015).

[91] Mert Yuksekgonul, Maggie Wang, and James Zou. *Post-hoc Concept Bottleneck Models*. 2023. arXiv: 2205.15480 [cs.LG].

[92] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *European Conference on Computer Vision*. 2013.

[93] Xiaohua Zhai et al. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV].

[94] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *CVPR*. 2018.

[95] Bolei Zhou et al. *Places: An Image Database for Deep Scene Understanding*. 2016. arXiv: 1610.02055 [cs.CV].

[96] Kaiyang Zhou et al. *Conditional Prompt Learning for Vision-Language Models*. 2022. arXiv: 2203.05557 [cs.CV].

[97] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

Appendix A

Chapter 2 Supplementary Material

Layer Normalization

We describe here the modifications that are needed to be incorporated in our method to take into account layer-normalizations. There are two places where layer-normalizations are used - before the projection layer (to the output of the ViT), and before each layer in the ViT (to the layer input). We present how the individual contributions of $c_{i,l,h}$, $c^{l,h}_{\text{head}}$ and c^{i}_{token} should be changed.

Pre-projection layer normalization. As mentioned in the Section 2.3, in many implementations of CLIP, a layer-normalization LN is applied to the output of the ViT before the projection layer. Formally, the image representation of image I is then:

$$M_{\text{image}}(I) = P\mathsf{LN}(\mathsf{ViT}(I)) \tag{A.1}$$

The normalization layer can be rewritten as:

$$\mathsf{LN}(x) = \gamma * \frac{x - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} + \beta = \left[\frac{\gamma}{\sqrt{\sigma_l^2 + \epsilon}}\right] * x - \left[\frac{\mu_l \gamma}{\sqrt{\sigma_l^2 + \epsilon}} - \beta\right] \tag{A.2}$$

where $x \in \mathbb{R}^d$ is the input token, $\mu_l, \sigma_l \in \mathbb{R}$ are the mean and standard deviation, and $\gamma, \beta \in \mathbb{R}^d$ are learned vectors. To incorporate the layer normalization in our decomposition, we compute the mean and the standard deviation during the forward pass of the model. The multiplicative term, $\frac{\gamma}{\sqrt{\sigma_l^2+\epsilon}}$ is absorbed into the projection matrix P. The contribution of $\frac{\mu_l \gamma}{\sqrt{\sigma_l^2+\epsilon}} - \beta$ is split equally between all the $c_{i,l,h}$ terms in the Eq. 2.6. We apply these modifications when we decompose OpenCLIP-based models.

MLPs and MSAs input layer normalizations. In the main paper, we do not describe the normalization layers that are applied to each input of MLP and MSA in the model. More accurately, the residual updates of the ViT are:

$$\hat{Z}^l = \mathsf{MSA}^l(\mathsf{LN}^l(Z^{l-1})) + Z^{l-1}, \quad Z^l = \mathsf{MLP}^l(\hat{\mathsf{LN}}^l(\hat{Z}^l)) + \hat{Z}^l \tag{A.3}$$

Where $\hat{\mathsf{LN}}^l$ and LN^l are the layer normalizations applied to each token in the input matrix of the MLP layers and MSA layers. This modification does not affect our corollaries about the direct contributions of the MLP layers and MSA layers, as we only address the outputs of these layers. The only other equation in which this modification takes place is in Eq. 3.2:

$$\left[\mathsf{MSA}^l(Z^{l-1})\right]_{cls} = \sum_{h=1}^{H}\sum_{i=0}^{N} x_i^{l,h}, \quad x_i^{l,h} = \alpha_i^{l,h}\mathsf{LN}^l(z_i^{l-1})W_{VO}^{l,h} \tag{A.4}$$

Mean-Ablation of the Class-Token Attended from Itself

We show that we can ignore the direct effect of the class token in the MSAs term when we decompose it into tokens (see section 2.5). We mean-ablate the direct contribution of the class token to the MSAs term in Eq. 2.6. We simultaneously ablate both the class token and the MLPs. The ImageNet zero-shot classification performances of the three ViT models are shown in Table A.1. As shown, the direct contributions of all the MLP layers *and* the direct contributions of the class token in the decomposed MSAs term results in a negligible drop in performance for all the models.

	Base accuracy	+ class token ablation	+ MLPs ablation
ViT-B-16	70.22	69.37	67.32
ViT-L-14	75.25	74.38	73.87
ViT-H-14	77.95	76.89	76.29

Table A.1: **Mean-ablation of the class token contribution to the MSAs term.** The overall drop in accuracy is relatively small, even when the MLPs are replaced by their mean across ImageNet validation set.

Text Descriptions

General text descriptions. To generate the set of text descriptions that are used by our algorithm, we prompted ChatGPT (GPT-3.5) to produce image descriptions. We used the prompt provided in Table A.2, and manually prompted the language model to generate more examples for specific patterns we found in the initial result (e.g. more colors, more letters). This process resulted in 3498 sentences.

Most common words. For the set of most common words, we used the same number of examples, and took the 3498 most common English words, as determined by n-gram frequency analysis of Google's Trillion Word Corpus ([75]).

Class-specific text descriptions. We generate additional class-specific text descriptions, by prompting ChatGPT with the prompt template provided in table A.2. We queried to model for each of the ImageNet class names. This process resulted in 28767 unique sentences.

Random vectors. As a baseline we created a random set of 3498 vectors sampled from a unit Gaussian.

General text descriptions initial prompt
Imagine you are trying to explain a photograph by providing a complete set of image characteristics. Provide generic image characteristics. Be as general as possible and give short descriptions presenting one characteristic at a time that can describe almost all the possible images of a wide range of categories. Try to cover as many categories as possible, and don't repeat yourself. Here are some possible phrases: "An image capturing an interaction between subjects", "Wildlife in their natural habitat", "A photo with a texture of mammals", "An image with cold green tones", "Warm indoor scene", "A photo that presents anger". Just give the short titles, don't explain why, and don't combine two different concepts (with "or" or "and"). Make each item in the list short but descriptive. Don't be too specific.
Class-specific text descriptions prompt
Provide 40 image characteristics that are true for almost all the images of $\{class\}$. Be as general as possible and give short descriptions presenting one characteristic at a time that can describe almost all the possible images of this category. Don't mention the category name itself (which is "$\{class\}$"). Here are some possible phrases: "Image with texture of ...", "Picture taken in the geographical location of...", "Photo that is taken outdoors", "Caricature with text", "Image with the artistic style of...", "Image with one/two/three objects", "Illustration with the color palette ...", "Photo taken from above/below", "Photograph taken during ... season". Just give the short titles, don't explain why, and don't combine two different concepts (with "or" or "and").

Table A.2: **ChatGPT prompts for image descriptions generation.**

Additional Initial Description Pool Ablation

We present additional ablation of the initial set of text descriptions provided to TEXTSPAN. The text description generation processes for each of the pools are described in Section A.

As shown in Figure A.1, using the class-specific descriptions pool that includes around $\times 8$ more examples than the general descriptions pool, allows us to obtain higher accuracy with fewer descriptions per head (smaller m). Nevertheless, using each of the two pools results in relatively similar accuracy with $m = 60$.

TEXTSPAN outputs for CLIP-ViT-L

We apply TEXTSPAN to the attention heads of the last 4 layers of CLIP ViT-L. Tables A.3-A.6 present the first 5 descriptions per head.

Qualitative results for image token decomposition

Figure A.2(a) shows the similarity heatmaps for text descriptions. As presented our heatmaps highlight the objects that are described in the text. Figure A.2(b) presents the relative similarity heatmaps given two descriptions (by subtracting between the two heatmaps). The areas in the

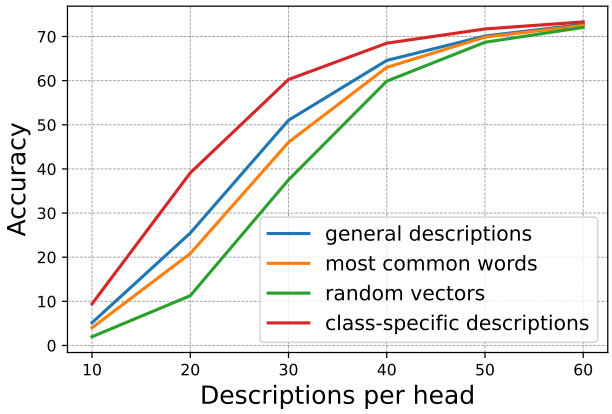

Figure A.1: ImageNet classification accuracy for the image representation projected to TEXTSPAN bases (additional results). We evaluate our algorithm for different initial description pools, and with different output sizes.

Figure A.2: **Heatmaps produced by the image token decomposition**. We visualize (a) what areas in the image directly contribute to the similarity score between the image representation and a text representation and (b) what areas make an image representation more similar to one text representation rather than another.

images that make the image representations more similar to one of the text representations rather than the other, correspond to the areas that are mentioned by it and ignored by the other text.

Most similar images to TEXTSPAN results

We randomly choose 3 attention heads from the last 4 layers of CLIP ViT-L. For each head (l, h), we retrieve the 3 images with the highest similarity score between their $c^{l,h}_{\text{head}}$ and the top 10 text representations found by our algorithm. The retrieval is done from ImageNet validation set. The results are presented in Figure A.5. As shown, in most cases, the top text representation corresponds to the attributes of the images.

Layer 20, Head 0	**Layer 20, Head 1**
Picture taken in Hungary	Picture taken in Seychelles
Image taken in New England	Picture taken in Saudi Arabia
Futuristic technological concept	Muted urban tones
Playful siblings	Man-made pattern
Picture taken in the English countryside	an image of glasgow
Layer 20, Head 2	**Layer 20, Head 3**
Image of a police car	Intrica wood carvingte
Picture taken in Laos	Image snapped in Spain
Remote alpine chalet	Photo taken in Bora Bora, French Polynesia
A photograph of a small object	An image of a Preschool Teacher
Desert sandstorm	A breeze
Layer 20, Head 4	**Layer 20, Head 5**
Image with a pair of subjects	an image of samoa
Image with five subjects	Urban nostalgia
Image with a trio of friends	A photo with the letter K
A photo of an adult	Image snapped in the Colorado Rockies
Image with a seven people	Serendipitous discovery
Layer 20, Head 6	**Layer 20, Head 7**
Bustling city square	Energetic children
Peaceful village alleyway	Grumpy facial expression
ornate cathedral	Intricate ceramic patterns
Image taken in the Alaskan wilderness	Photo taken in Bangkok, Thailand
Modern airport terminal	Subdued moments
Layer 20, Head 8	**Layer 20, Head 9**
Photo taken in Rioja, Spain	Tranquil Asian temple
Photo taken in Borneo	Vibrant city nightlife
Vibrant urban energy	A photo with the letter R
Picture captured in the Icelandic glaciers	intricate mosaic artwork
serene oceanside scene	Photo taken in the Rub' al Khali (Empty Quarter)
Layer 20, Head 10	**Layer 20, Head 11**
A bowl	Photo taken in Beijing, China
A bottle	Photo with retro color filters
Nostalgic pathways	Image with holographic cyberpunk aesthetics
A laptop	Urban street fashion
Reflective ocean view	Photograph with the artistic style of tilt-shift
Layer 20, Head 12	**Layer 20, Head 13**
Photo with grainy, old film effect	Image taken from a distance
Detailed illustration	Photograph with the artistic style of split toning
Serene beach sunset	Photo taken in Beijing, China
An image of the number 10	A close-up shot
An image of the number 5	An image of a Novelist
Layer 20, Head 14	**Layer 20, Head 15**
Quirky street performer	Remote hilltop hut
Antique sculptural element	Photo taken in Barcelona, Spain
Celebratory atmosphere	Dynamic movement
Overwhelmed facial expression	Caricature of an influential leader
Serene winter wonderland	A picture of Samoa

Table A.3: **Top-5 results of TEXTSPAN.** Applied to the heads at layer 20 of CLIP-ViT-L.

Layer 21, Head 0	**Layer 21, Head 1**
Timeless black and white	Picture taken in the southeastern United States
Vintage sepia tones	Picture taken in the Netherlands
Image with a red color	Image taken in Brazil
A charcoal gray color	Image captured in the Australian bushlands
Soft pastel hues	Picture taken in the English countryside
Layer 21, Head 2	**Layer 21, Head 3**
A photo of a woman	Precise timekeeping mechanism
A photo of a man	Image snapped in the Canadian lakes
Energetic children	An image of Andorra
An image with dogs	thrilling sports challenge
A picture of a baby	Photo taken in Namib Desert
Layer 21, Head 4	**Layer 21, Head 5**
An image with dogs	Inquisitive facial expression
A picture of a bridge	Artwork featuring typographic patterns
A photo with the letter R	A photograph of a big object
Dramatic skies	Reflective landscape
Ancient castle walls	Burst of motion
Layer 21, Head 6	**Layer 21, Head 7**
Photo taken in the Italian pizzerias	A pin
thrilling motorsport race	A thimble
Urban street fashion	A bookmark
An image of a Animal Trainer	Picture taken in Rwanda
Serene countryside sunrise	A pen
Layer 21, Head 8	**Layer 21, Head 9**
Inviting coffee shop	Photograph with a blue color palette
Photograph taken in a music store	Image with a purple color
An image of a News Anchor	Image with a pink color
Joyful family picnic scene	Image with a orange color
cozy home library	Timeless black and white
Layer 21, Head 10	**Layer 21, Head 11**
Playful winking facial expression	Photo captured in the Arizona desert
Joyful toddlers	Picture taken in Alberta, Canada
Close-up of a textured plastic	Photo taken in Rio de Janeiro, Brazil
An image of a Teacher	Picture taken in Cyprus
Image with a seven people	Photo taken in Seoul, South Korea
Layer 21, Head 12	**Layer 21, Head 13**
Photo with grainy, old film effect	Quiet rural farmhouse
Macro botanical photography	Lively coastal fishing port
A laptop	an image of liechtenstein
Vintage nostalgia	Image taken in the Florida Everglades
serene mountain retreat	thrilling motorsport race
Layer 21, Head 14	**Layer 21, Head 15**
Photo taken in Beijing, China	Submerged underwater scene
Cheerful adolescents	Artwork featuring overlapping scribbles
Picture taken in Ecuador	Surrealist artwork with dreamlike elements
Dreamy haze	Serene winter wonderland
Image captured in the Greek islands	Wildlife in their natural habitat

Table A.4: **Top-5 results of TEXTSPAN.** Applied to the heads at layer 21 of CLIP-ViT-L.

Layer 22, Head 0	**Layer 22, Head 1**
Artwork with pointillism technique	A semicircular arch
Artwork with woven basket design	An isosceles triangle
Artwork featuring barcode arrangement	An oval
Image with houndstooth patterns	Rectangular object
Image with quilted fabric patterns	A sphere
Layer 22, Head 2	**Layer 22, Head 3**
Urban park greenery	An image of legs
cozy home interior	A jacket
Urban subway station	A helmet
Energetic street scene	A scarf
Tranquil boating on a lake	A table
Layer 22, Head 4	**Layer 22, Head 5**
An image with dogs	Harmonious color scheme
Joyful toddlers	An image of cheeks
Serene waterfront scene	Vibrant vitality
thrilling sports action	Captivating scenes
A picture of a baby	Dramatic chiaroscuro photography
Layer 22, Head 6	**Layer 22, Head 7**
Curious wildlife	Serene winter wonderland
Majestic soaring birds	Blossoming springtime blooms
An image with dogs	Crisp autumn leaves
Image with a dragonfly	A photo taken in the summer
An image with cats	Posed shot
Layer 22, Head 8	**Layer 22, Head 9**
A photo with the letter V	A photo of food
A photo with the letter F	delicate soap bubble play
A photo with the letter D	Dynamic and high-energy music performance
A photo with the letter T	Hands in an embrace
A photo with the letter X	Futuristic technology display
Layer 22, Head 10	**Layer 22, Head 11**
Image with a yellow color	A charcoal gray color
Image with a orange color	Sepia-toned photograph
An image with cold green tones	Minimalist white backdrop
Image with a pink color	High-contrast black and white
Sepia-toned photograph	Image with a red color
Layer 22, Head 12	**Layer 22, Head 13**
Photo taken in Namib Desert	Image taken in Thailand
Ocean sunset silhouette	Picture taken in the Netherlands
Photo taken in the Brazilian rainforest	Picture taken in the southeastern United States
Serene countryside sunrise	Image captured in the Australian bushlands
Bustling cityscape at night	Picture taken in the geographical location of Spain
Layer 22, Head 14	**Layer 22, Head 15**
A silver color	contemplative urban view
Play of light and shadow	Photograph revealing frustration
Image with a white color	Celebratory atmosphere
A charcoal gray color	Captivating authenticity
Cloudy sky	Intense athletic competition

Table A.5: **Top-5 results of TEXTSPAN.** Applied to the heads at layer 22 of CLIP-ViT-L.

Layer 23, Head 0	**Layer 23, Head 1**
Intrica wood carvingte	Photograph taken in a retro diner
Nighttime illumination	Intense athlete
Image with woven fabric design	Detailed illustration of a futuristic bioreactor
Image with shattered glass reflections	Image with holographic retro gaming aesthetics
A photo of food	Antique historical artifact
Layer 23, Head 2	**Layer 23, Head 3**
Image showing prairie grouse	Bustling city square
Image with a penguin	Serene park setting
A magnolia	Warm and cozy indoor scene
An image with dogs	Modern airport terminal
An image with cats	Remote hilltop hut
Layer 23, Head 4	**Layer 23, Head 5**
Playful siblings	Intertwined tree branches
A photo of a young person	Flowing water bodies
Image with three people	A meadow
A photo of a woman	A smoky plume
A photo of a man	Blossoming springtime blooms
Layer 23, Head 6	**Layer 23, Head 7**
Picture taken in Sumatra	A paddle
Picture taken in Alberta, Canada	A ladder
Picture taken in the geographical location of Spain	Intriguing and enigmatic passageway
Image taken in New England	A bowl
Photo captured in the Arizona desert	A table
Layer 23, Head 8	**Layer 23, Head 9**
Photograph with a red color palette	ornate cathedral
An image with cold green tones	detailed reptile close-up
Timeless black and white	Image with a seagull
Image with a yellow color	A clover
Photograph with a blue color palette	Futuristic space exploration
Layer 23, Head 10	**Layer 23, Head 11**
Image with six subjects	A photo with the letter N
Image with a four people	A photo with the letter J
An image of the number 3	Serendipitous discovery
An image of the number 10	A fin
The number fifteen	Unusual angle
Layer 23, Head 12	**Layer 23, Head 13**
Image with polka dot patterns	Photo taken in a museum
Striped design	Surreal digital collage
Checkered design	Cinematic portrait with dramatic lighting
Artwork with pointillism technique	Collage of vintage magazine clippings
Photo taken in Galápagos Islands	Candid documentary photography
Layer 23, Head 14	**Layer 23, Head 15**
An image with dogs	Resonant harmony
Majestic soaring birds	Subtle nuance
Graceful swimming fish	An image of cheeks
An image with bikes	emotional candid gaze
Picture with boats	Whimsicachildren's scenel

Table A.6: **Top-5 results of TEXTSPAN.** Applied to the heads at layer 23 of CLIP-ViT-L.

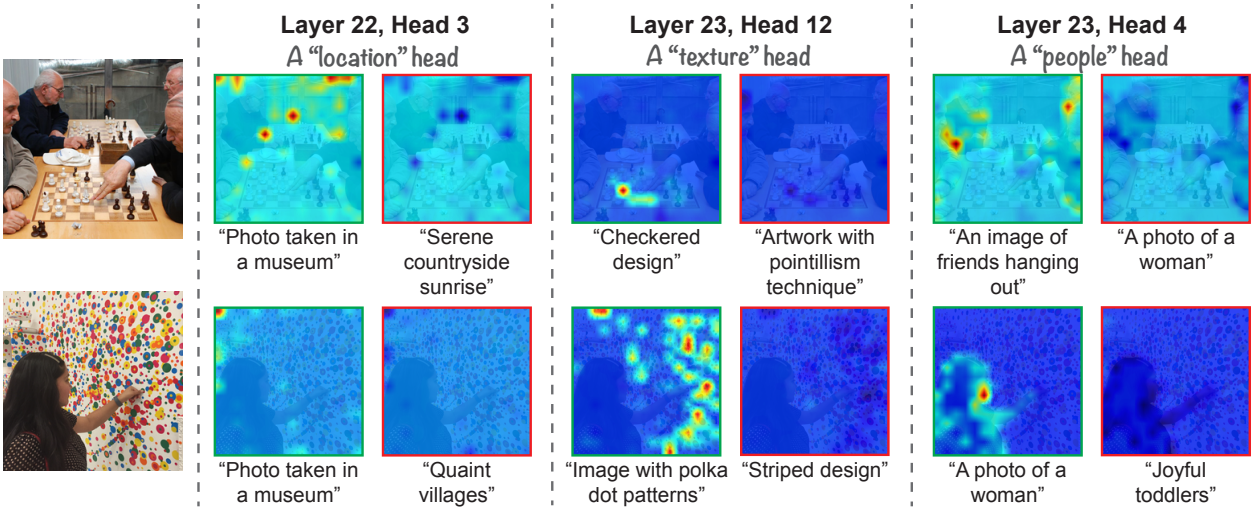

Figure A.3: **Additional joint decomposition examples.**

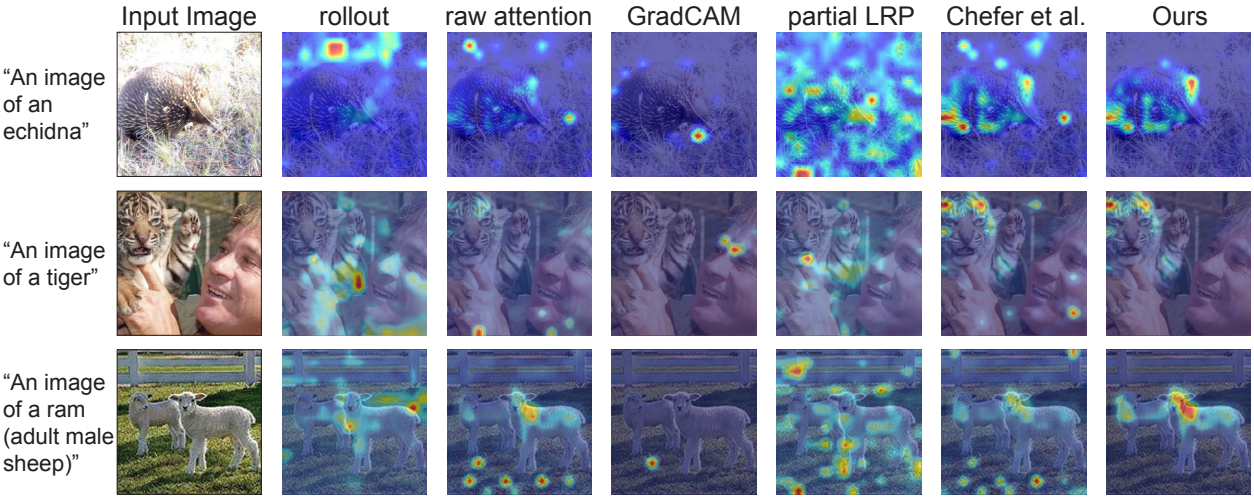

Figure A.4: **Comparison to other explainability methods**. The highlighted regions produced by our decomposition are more aligned with the areas of the image that are mentioned in the text.

	base	ours
ViT-B-16	76.7	**83.8**
ViT-L-14	73.1	**84.2**
ViT-H-14	77.0	**84.1**

Table A.7: **Overall classification accuracy on Waterbirds dataset.** We reduce spurious cues by zeroing the direct effects of property-specific heads.

	water background	land background
waterbird class	92.1 (**93.1**)	**77.8** (66.2)
landbird class	**72.9** (47.7)	**94.9** (94.8)

Table A.8: **Zero-shot classification accuracy on Waterbirds dataset, per class and background (ViT-L)**. The accuracy for the baseline CLIP model is in parentheses. As shown, we reduce the spurious correlation between the background and the object class.

	water background	land background
waterbird class	62.3 (**69.8**)	**43.3** (37.2)
landbird class	**87.9** (71.0)	**98.0** (96.4)

Table A.9: **Zero-shot classification accuracy on Waterbirds dataset, per class and background (ViT-H)**. The accuracy for the baseline CLIP model is in parentheses.

	water background	land background
waterbird class	80.5 (**86.1**)	**81.6** (63.5)
landbird class	**57.5** (45.6)	94.3 (**96.1**)

Table A.10: **Zero-shot classification accuracy on Waterbirds dataset, per class and background (ViT-B)**. The accuracy for the baseline CLIP model is in parentheses.

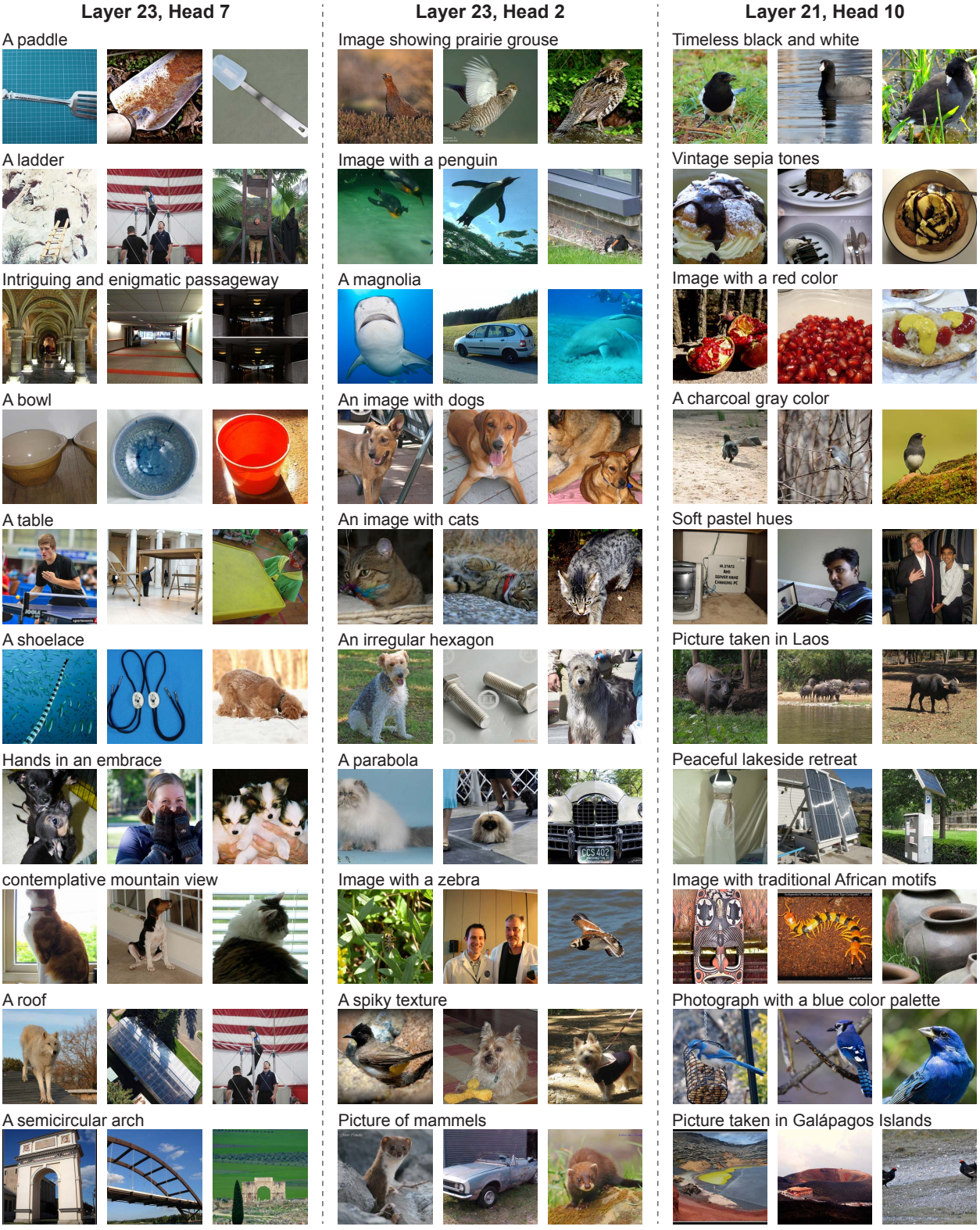

Figure A.5: **Top 3 images with highest similarities to TEXTSPAN outputs.** For 3 randomly selected attention heads, we retrieve the images with the highest similarity score between their head contributions $c^{l,h}_{\text{head}}$ and the top 10 text representations found by our algorithm.

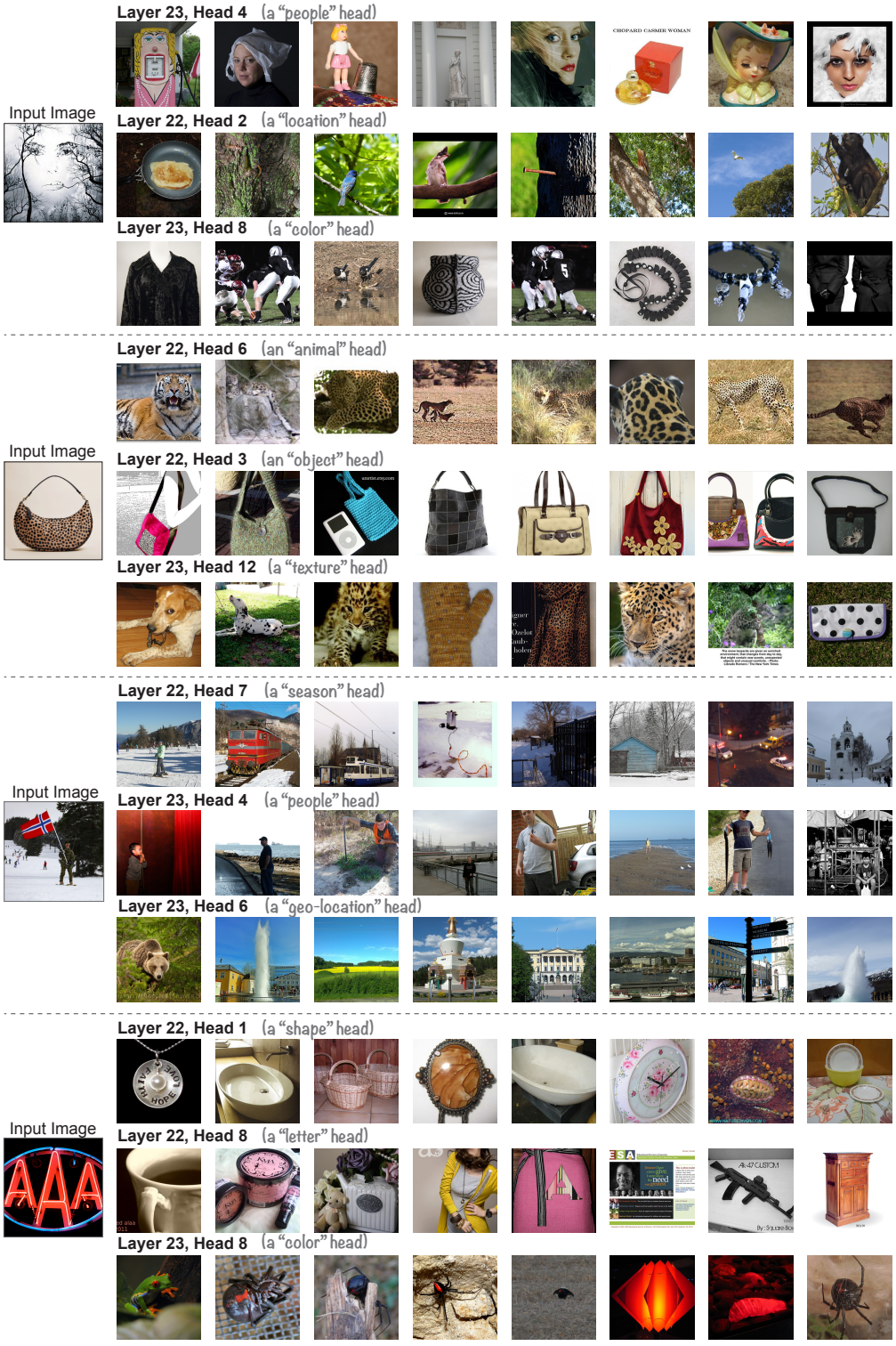

Figure A.6: **Additional results for image retrieval based on head-specific similarity.**

Appendix B

Chapter 3 Supplementary Material

Second order ablations for ViT-L

We repeat the same experiments from Chapter 3.3 for ViT-L-14, trained on LAION dataset [74]. For this model, we only use 10% of ImageNet validation set. Here, the maximal drop in performance when ablating the second order is relatively smaller and is spread across more layers. Nevertheless, the same properties presented and discussed in Chapter 3.3 hold for this model.

First order ablations

For the two models discussed above, ViT-B-32 and ViT-L-14, we provide the mean-ablation results for the first-order effects of MSA layers, as computed in [25]. For each model, we present the performance before and after accumulative mean-ablation of all the first-order effects of MSA layers. As shown in Chapter B.4 and Chapter B.5, the neurons with the significant second-order effects appear right before the layers with the significant first-order effects.

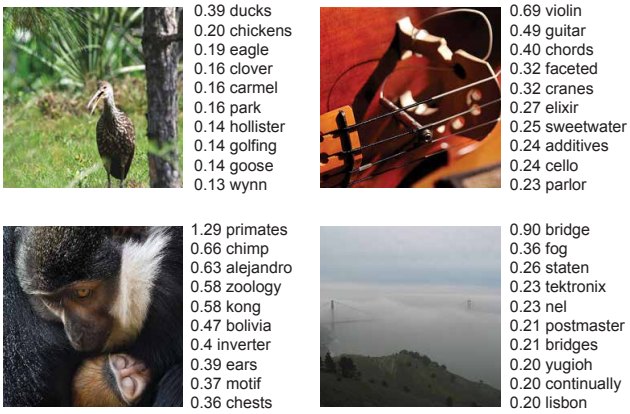

Figure B.1: **Concept discovery in images (ViT-B-32).** We include top-10 words discovered by aggregating words in sparse decompositions of activated neurons.

Additional adversarial images

We present additional semantic adversarial results, generated by our method for ViT-B-32, in Chapter B.8. We demonstrate a wide variety of tasks, including additional pairs from CIFAR-10 dataset, and adversarial attacks related to traffic signs (e.g. misclassification between a stop sign and a yield sign or a crossroad). For each image, we provide the text used for generating it, and highlight the spurious cues words from the sparse decompositions.

Additional sparse decomposition results

We provide additional examples of sparse decompositions of neurons in Chapter B.1 and the images with the top norms for the second-order effects of the same neurons in Chapter B.7. As shown, the found descriptions match the objects in the top 10 images.

Concept discovery in images

We present an additional application - concept discovery in images. We aim to discover concepts in image I, by aggregating phrases that correspond to the neurons that are activated on I. Here, we start from the set of *activated* neurons $\mathcal{N}$ (for which $||\phi_n^l(I)||_2$ is above the 98th percentile of norms computed across ImageNet images). Similarly to the contribution score described in Chapter 3.5, we compute an *image-contribution score* w_j^I for each phrase j according to its combined weight in the decompositions of neurons in $\mathcal{N}$. Formally, w_j^I is the overall sum of weights that each neuron in $\mathcal{N}$ assigns to j in its decomposition, weighted by the neuron second-order norms: $w_j^I = \sum_{n \in \mathcal{N}} \gamma_j^{l,n} ||\phi_n^l(I)||_2$. The phrases with the highest image-contribution score are picked to describe the image concepts.

Qualitative results. We present qualitative results for neurons and the top-10 discovered concepts from layer 9 of ViT-B-32 in Chapter B.1, when using the most common words as the pool. The number of neurons activated on these images, $|\mathcal{N}|$, is between 29 and 59, less than 2% of the neurons in the layer. Nevertheless, the top words extracted from these neurons relate semantically to the objects in the image and their locations. Surprisingly, the top word for each of the images appears only in one or two of the neuron sparse decompositions and is not spread across many activated neurons.

We acknowledge that while this application discovers meaningful concepts that correspond to the input images, there are other approaches for extracting these concepts (e.g. sparsely decomposing the image representation, as shown in [6]).

Derivations with Layer Normalization

In many implementations of CLIP, there is a layer normalization between the Vision Transformer and the projection layer P. In this case, the representation is:

$$M_{\text{image}}(I) = P(LN(\mathsf{ViT}(I))) \tag{B.1}$$

where the LN is the layer normalization. Specifically, LN can be written as:

$$LN(x) = \gamma * \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta = \underbrace{\left[\frac{\gamma}{\sqrt{\sigma^2+\epsilon}}\right]}_{=A} * x - \underbrace{\left[\frac{\mu\gamma}{\sqrt{\sigma^2+\epsilon}} - \beta\right]}_{=B}, \tag{B.2}$$

where $x \in \mathbb{R}^d$ is the input token, $\mu_l, \sigma_l \in \mathbb{R}$ are the mean and standard deviation, and $\gamma, \beta \in \mathbb{R}^d$ are learned vectors. To include A and B in the second-order effect of a neuron flow, we replace the input-independent component in Chapter 3.3, $PW_{VO}^{l',h}w^{l',n}$, with:

$$P(A * W_{VO}^{l',h}w^{l,n} + \frac{B}{c}) \tag{B.3}$$

Where c is a normalization constant that splits B equally across all the neurons that can additively contribute to it.

Except for the layer normalization before the projection layer, the input to the MSA layers that comes from the residual stream also flows through a layer normalization. Thus, if the input to the MSA layer in layer l is the list of tokens $[z_0^l, ...z_K^l]$, the output that corresponds to the class token is:

$$\left[\mathsf{MSA}^l([z_0, ..., z_K])\right]_0 = \sum_{h=1}^{H}\sum_{i=0}^{K} a_i^{l,h}(I)W_{VO}^{l,h}LN^l(z_i), \tag{B.4}$$

where LN^l is the normalization layer at layer l, that can be parameterized similarly to Chapter B.2 by $A^l, B^l \in \mathbb{R}^d$. We modify the definition of the second-order effect accordingly:

$$\phi_n^l(I) = \sum_{l'=l+1}^{L}\sum_{h=1}^{H}\sum_{i=0}^{K}\left(p_i^{l,n}(I)a_i^{l',h}(I)\right)\left(P\left(A * W_{VO}^{l',h}(A^l * w^{l,n} + \frac{B^{l'}}{c^{l'}}) + \frac{B}{c}\right)\right), \tag{B.5}$$

where $c^{l'}$ is is a normalization coefficient that splits $B^{l'}$ equally across all the neurons before layer l'.

In all of our experiments, we use this modification. Most of the elements in the modification add constant biases. Therefore, they can be ignored in our experiments as in many of the experiments constant biases do not change the results. For example, in our mean-ablation experiment, we subtract the mean, computed across a dataset.

Prompts

We provide the prompt that was used for generating sentences given the set of words W_v, as presented in Chapter 3.5, in Chapter B.2. This prompt is given to LLAMA3 model [85].

Additionally, we provide the prompt that was used for generating the pool of ImageNet class descriptions, presented in Chapter 3.4. We prompt ChatGPT (GPT 3.5) with the prompt template provided in Chapter B.3.

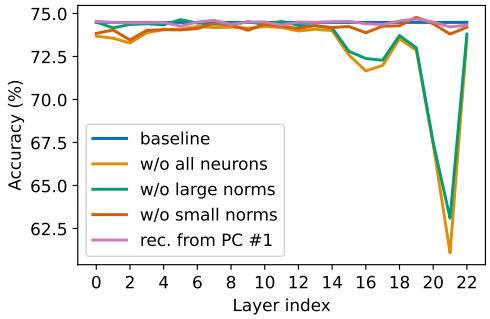

Figure B.2: **ViT-L-14 second-order ablations.**

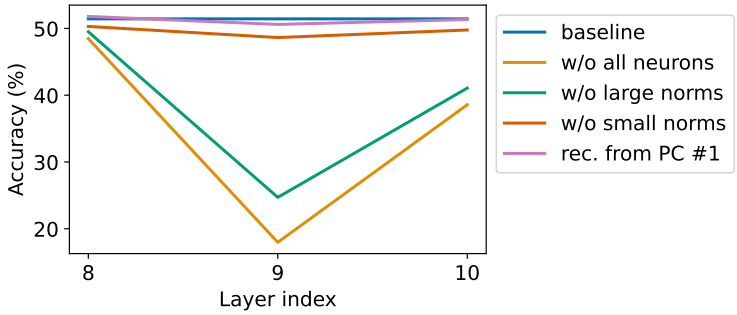

Figure B.3: **Mean-ablation of second order effects on ImageNet-R (ViT-B-32, layers 8-10).** We repeat the evaluation in Chapter 3.3 on ImageNet-R. The performance of different ablations follows the same trends as that of ImageNet.

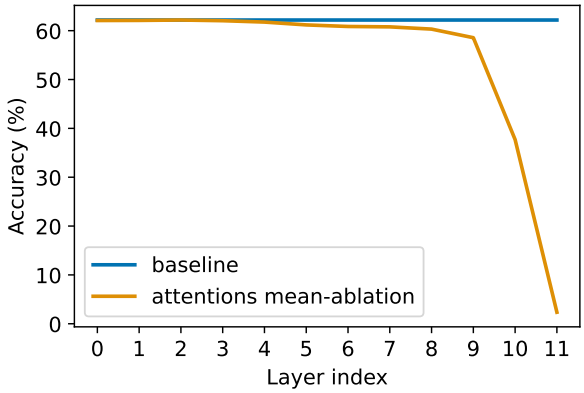

Figure B.4: **ViT-B-32 first-order MSAs ablation.**

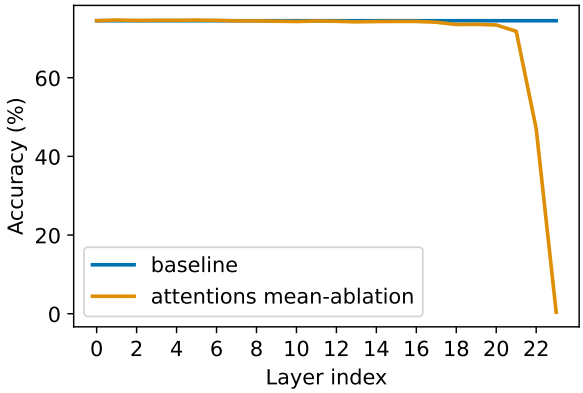

Figure B.5: **ViT-L-14 first-order MSAs ablation.**

Compute

As our method does not require additional training, the time of our experiments depends linearly on the inference time of CLIP (and other generative models that were used for the adversarial images generation), and on the number of images we use for the experiments (∼5000 in our case). All our experiments were run on one A100 GPU. The most time-consuming experiment—computing the per-layer mean-ablation results for ViT-L-14—took 5 days.

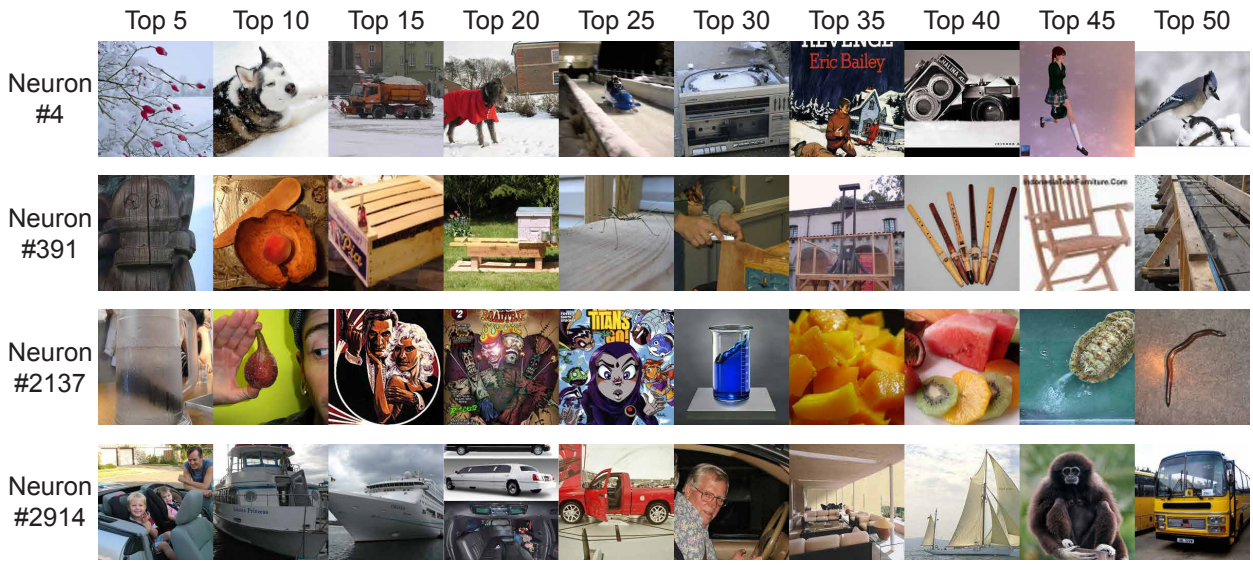

Figure B.6: **Images with largest second-order effect norm per neuron.** We present the top images from 10% of ImageNet validation set, for the neurons in Chapter 3.2. Notice that additional concepts that are not captured by the top-4 descriptions in Chapter 3.2 are starting to appear.

Neuron	ImageNet class descriptions	Common words (30k)
#600	+“Image with a wiry, weather-resistant coat” +“Image showcasing a compact and lightweight sleeping bag” +“Picture of a camper towing bicycles” +“Image with a Border Terrier jumping”	+“tents” +“svalbard” +“miles” -“mountainous”
#974	-“Photograph taken during a race” -“Silhouette of a running dog” -“Picture taken in a fishing competition” +“Silhouette of hammerhead shark with other ocean creatures”	+“runners” +“races” -“dolphin” +“expiration”
#1517	+“Chair with a foot pedal control” -“Picture that captures the breed’s intelligence” -“Image with snow-capped mountains as scenery” +“Image with graffiti on a train”	+“bus” -“filings” -“percussion” +“wheelchairs”
#2002	+“Image depicting a sustainable living option” +“Photo taken in a train yard” -“Image featuring snow-covered rooftops” +“Rescue equipment”	-“genres” +“governance” +“‘gravel” +“conserve”

Table B.1: **Additional examples of sparse decomposition results.** For each neuron, we present the top-4 texts corresponding to the sparse decomposition with $m = 128$ and the signs of the coefficients in the decomposition.

You are a capable instruction-following AI agent. I want to generate an image by providing image descriptions as input to a text-to-image model. The image descriptions should be short. Each of them must include the word "{*class_1*}". They must not include the word "{*class_2*}", any synonym of it, or a plural version! The image descriptions should include as many words as possible from the next list and almost no other words: {*list*} Do not use names of people or places from the list unless they are famous and there is something visually distinctive about them. In each of the image descriptions mention as many objects and animals as possible from the list above. If you want to mention the place in which the image is taken or a name of a person, describe it with visually distinctive words. For example, if "Paris" is in the list, instead of saying "... in Paris", say "... with the Eiffel Tower in the background" or "... next to a sign saying 'Paris'". Don't mention words that are too similar to "{*class_2*}", even if they are in the list above. For example, if the word was "tree" you should not mention "trees", "bush" or "eucalyptus". Only use words that you know what they mean. Generate a list of 50 image descriptions.

Table B.2: **The language model prompt for generating image descriptions.**

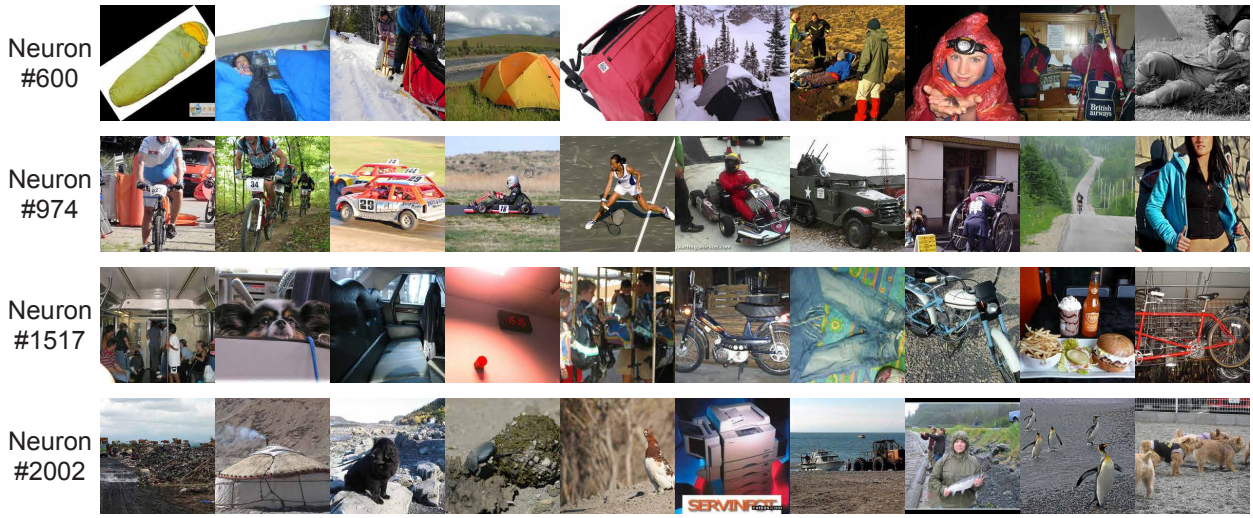

Figure B.7: **Images with largest second-order effect norm per neuron.** We present the top images from 10% of ImageNet validation set, for the neurons in Chapter B.1.

Provide 40 image characteristics that are true for almost all the images of {*class*}. Be as general as possible and give short descriptions presenting one characteristic at a time that can describe almost all the possible images of this category. Don't mention the category name itself (which is "{*class*}"). Here are some possible phrases: "Image with texture of ...", "Picture taken in the geographical location of...", "Photo that is taken outdoors", "Caricature with text", "Image with the artistic style of...", "Image with one/two/three objects", "Illustration with the color palette ...", "Photo taken from above/below", "Photograph taken during ... season". Just give the short titles, don't explain why, and don't combine two different concepts (with "or" or "and").

Table B.3: **The prompt for generating the pool of class descriptions.** We prompt the model with all the ImageNet classes.

Figure B.8: **Additional adversarial examples generated by our method.** We provide the sentence that was given to the text-to-image model to generate it. Words from W^v are highlighted in bold.

Appendix C

Chapter 4 Supplementary Material

We provide extended examples of Rosetta dictionaries as well as additional edits and visualizations.

Figure C.1: **Additional out-of-distribution and cross-class inversions**. We show out-of-distribution image inversions done by Rosetta Neurons guidance for StyleGAN2 model, trained on LSUN cats (left 3 images) and LSUN horses (right 3 images).

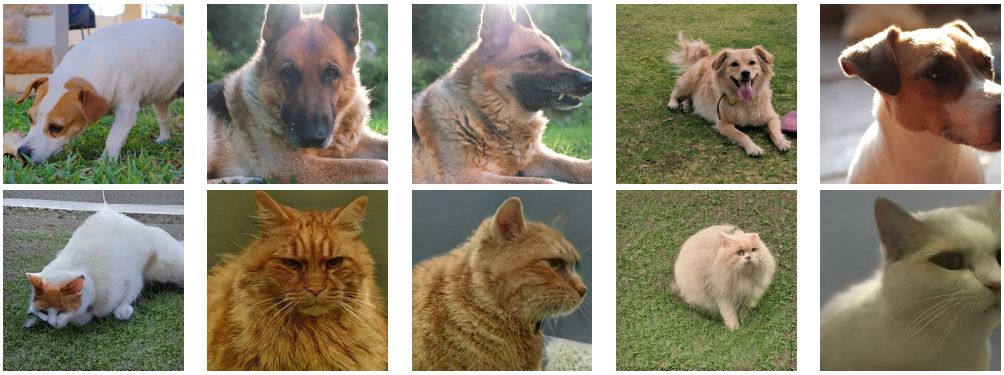

Figure C.2: **Dog-to-cat cross-class inversions**. Using Rosetta Neurons guidance for StyleGAN2 model, trained on LSUN cats.

Figure C.3: **Additional examples of Rosetta Neurons guided editing.** We show examples using BigGAN and its matches to CLIP-RN.

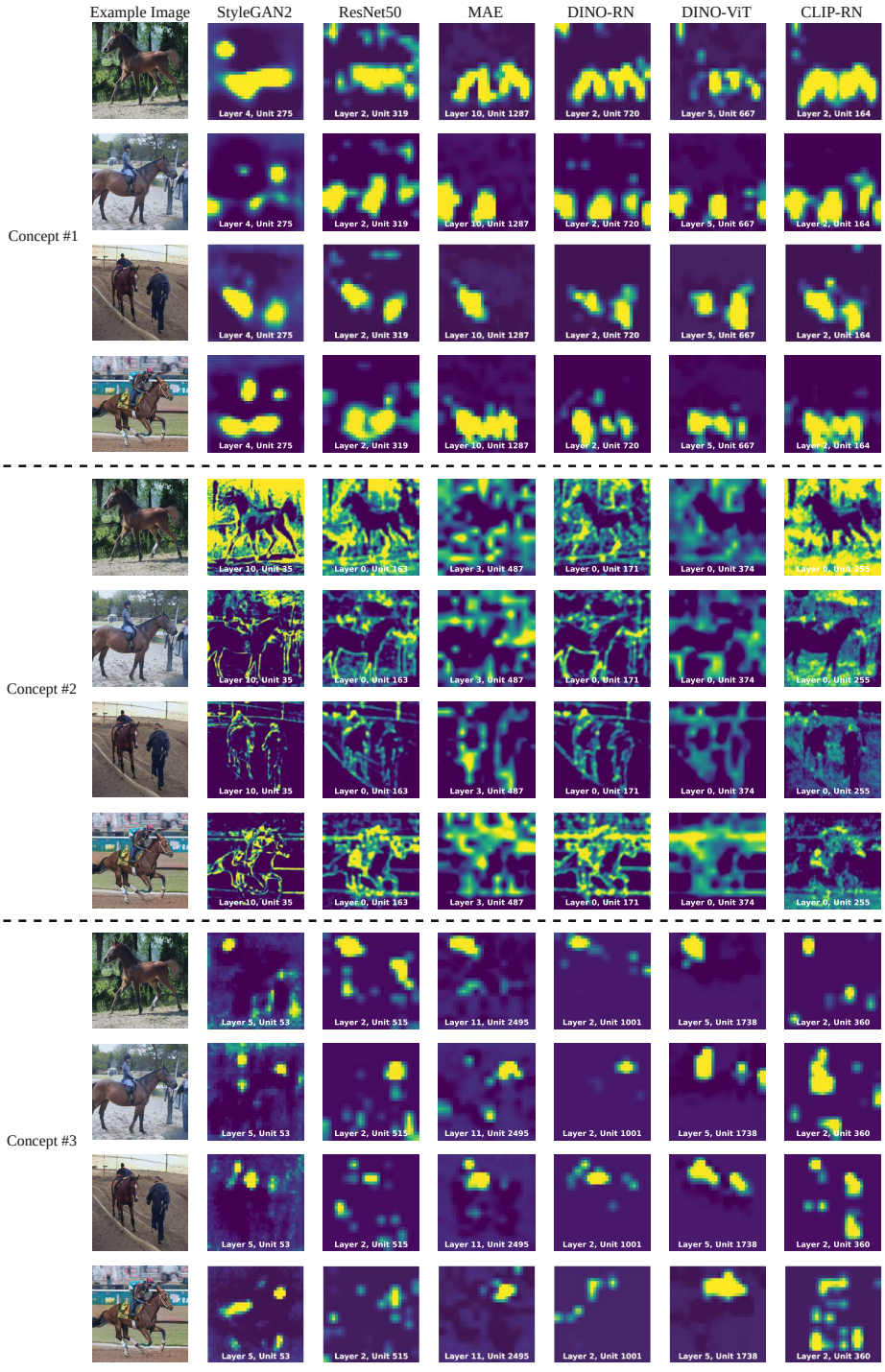

Figure C.4: **Rosetta Neuron Dictionary for LSUN-horses.** A sample from the dictionary curated for the LSUN-horses dataset. The figure presents 6 emergent concepts demonstrated in 4 example images.

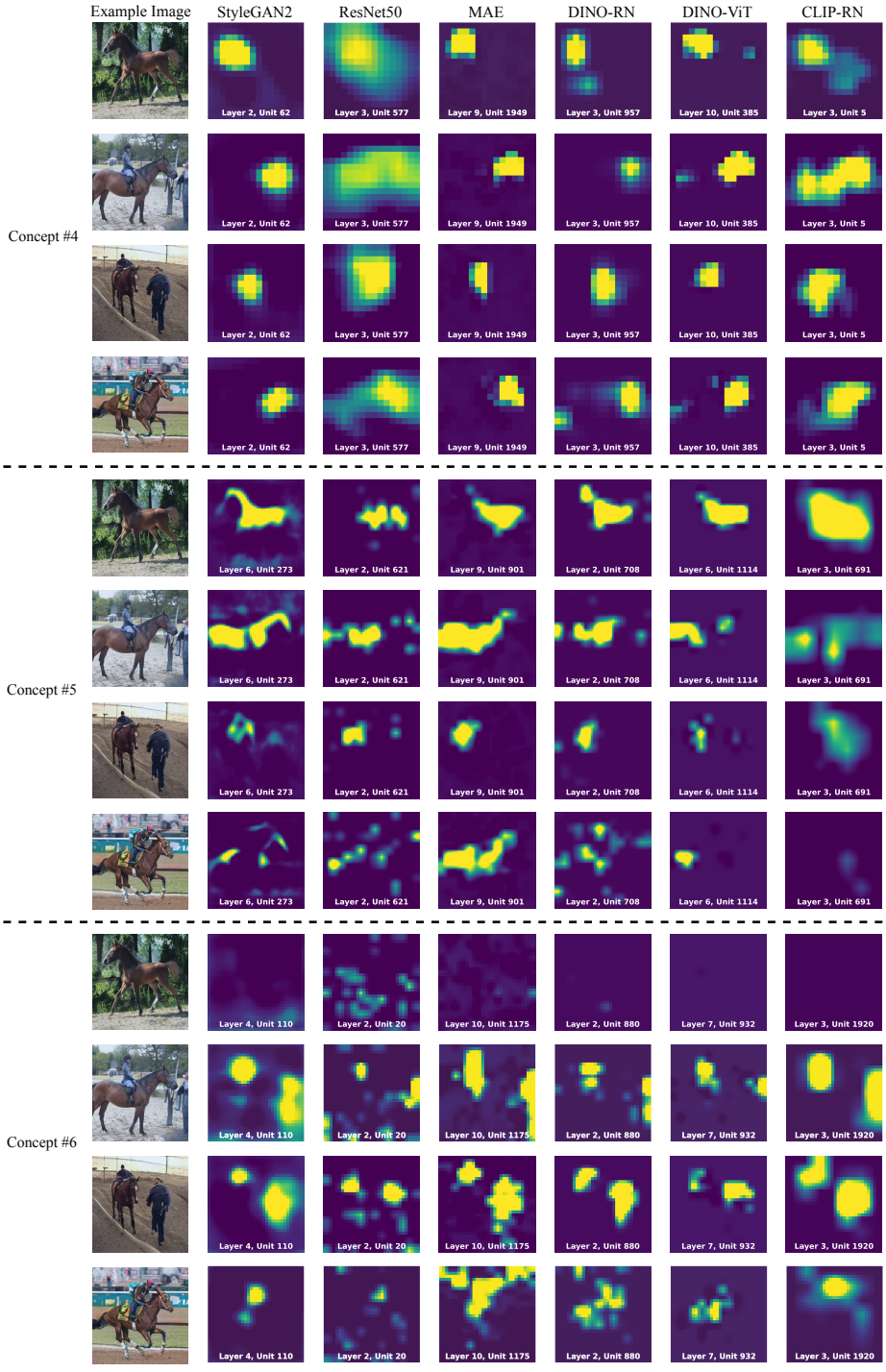

Figure C.5: **Rosetta Neuron Dictionary for LSUN-horses (cont.)**

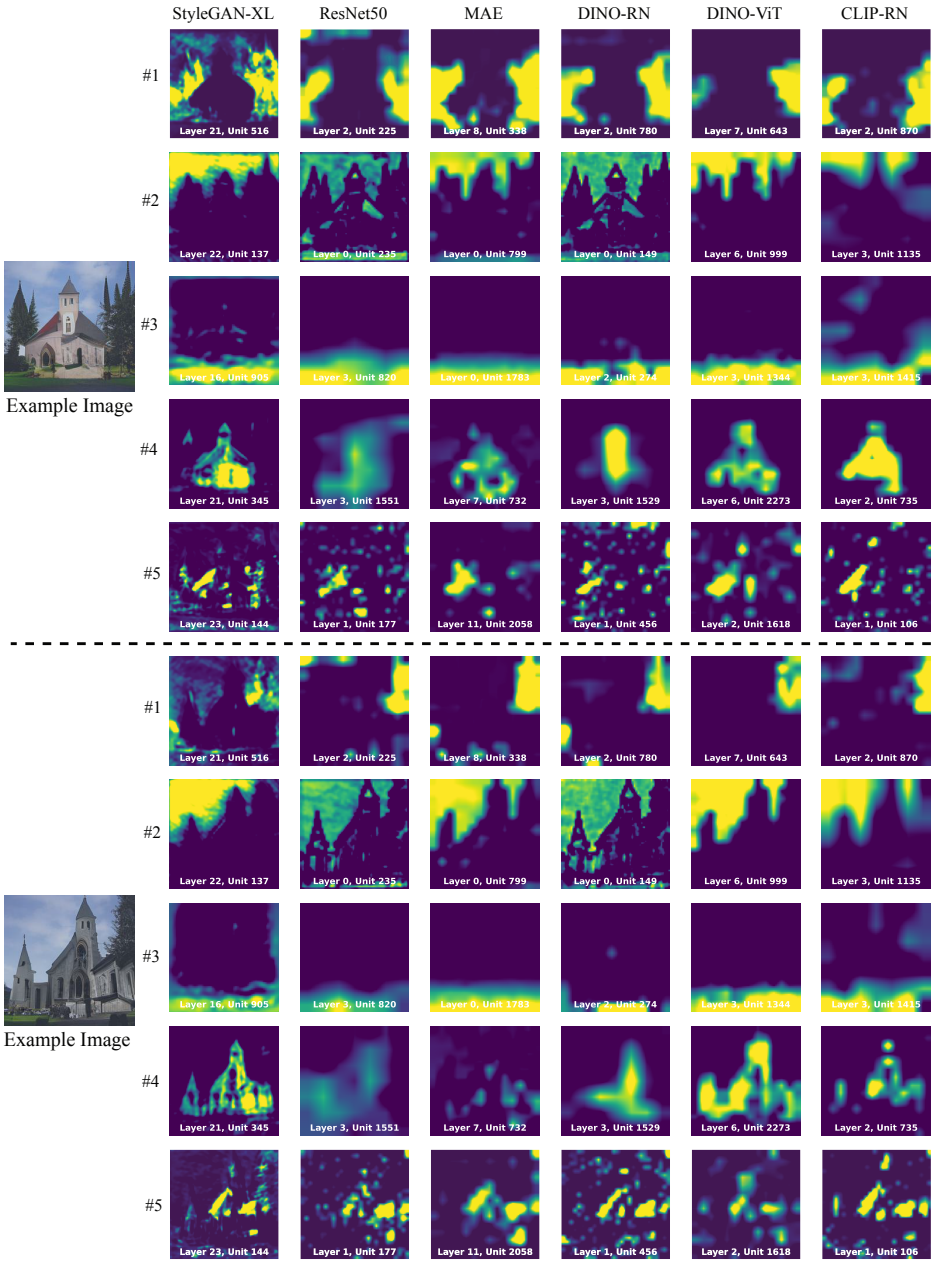

Figure C.6: **Rosetta Neuron Dictionary.** A sample from the dictionary curated for the ImageNet class "Church". The figure presents 5 emergent concepts demonstrated in 2 example images.

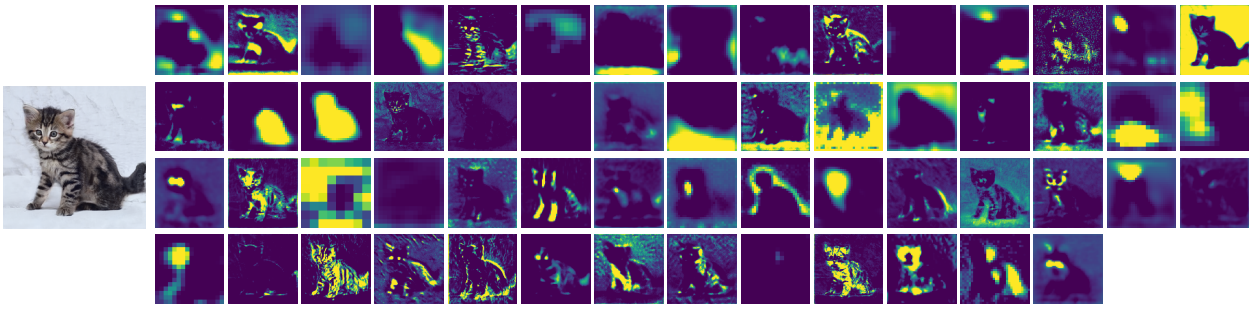

Figure C.7: **All the concepts for LSUN-cats.** Shown for one StyleGAN2 generated image.

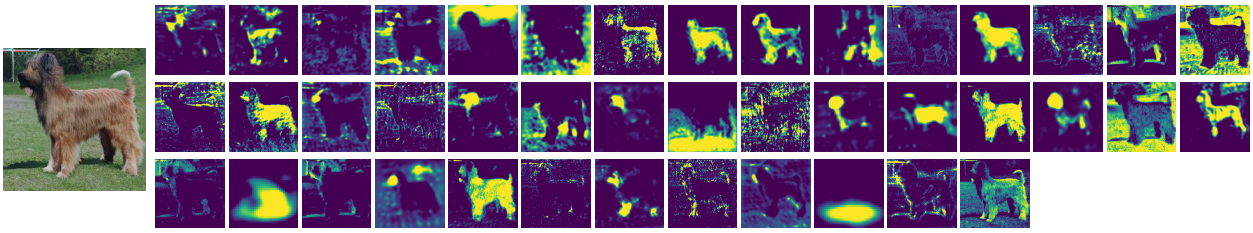

Figure C.8: **All the concepts for ImageNet class "Briard".** Shown on one StyleGAN-XL generated image.

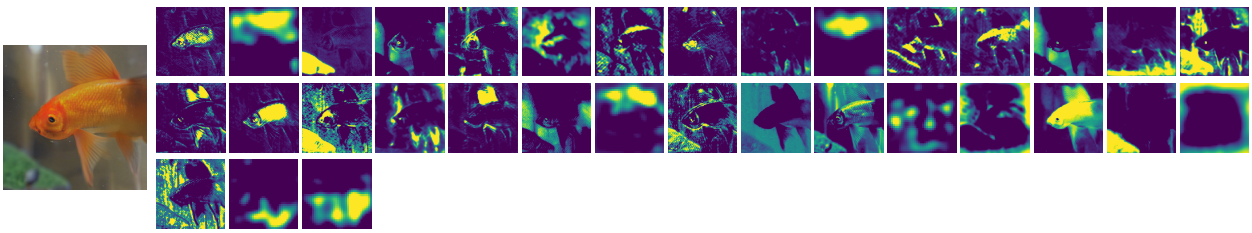

Figure C.9: **All the concepts for ImageNet class "Goldfish".** Shown on one StyleGAN-XL generated image.

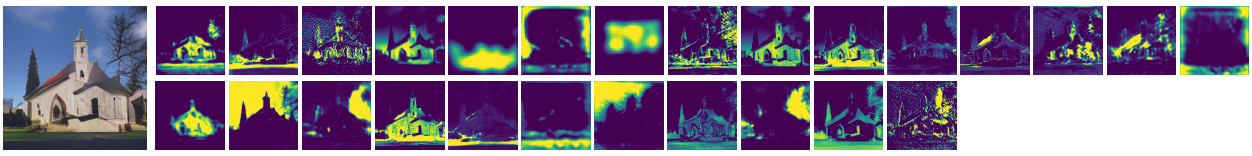

Figure C.10: **All the concepts for ImageNet class "Church".** Shown on one StyleGAN-XL generated image.

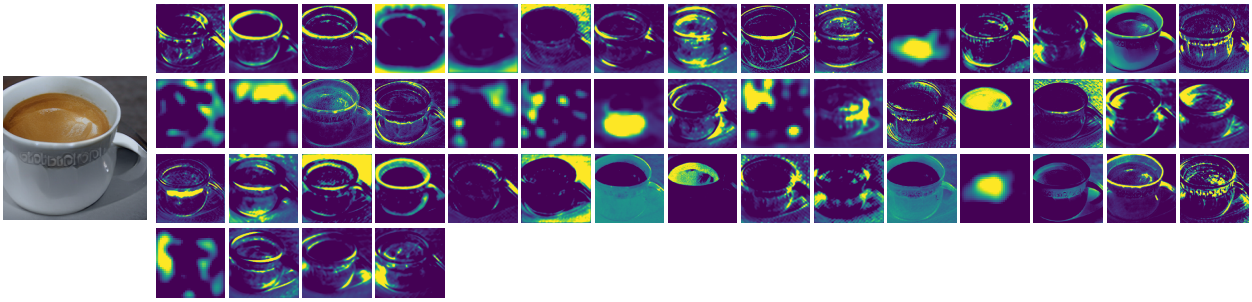

Figure C.11: **All the concepts for ImageNet class "Espresso".** Shown on one StyleGAN-XL generated image.

Figure C.12: **Additional Single Rosetta Neurons Edits**. By decreasing (two left image pairs) or increasing (two right image pairs) the values of specific manually chosen Rosetta Neurons before the latent optimization process, we can remove or add elements to the image. In this figure, we demonstrate (left to right): Removing lava eruptions, removing trees, adding Crema to an Espresso, and adding a dog's tongue. For the leftmost example, we also provide the complete list of Rosetta Neurons visualizations. The chosen concept is marked with a red frame.

Figure C.13: **Additional image inversions for StyleGAN-XL.** We compare using perceptual loss (second row) to perceptual loss with additional guidance from the Rosetta Neurons (third row).