# CLIP-DISSECT: AUTOMATIC DESCRIPTION OF NEURON REPRESENTATIONS IN DEEP VISION NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In this paper, we propose **CLIP-Dissect**, a new technique to automatically describe the function of individual hidden neurons inside vision networks. **CLIP-Dissect** leverages recent advances in multimodal vision/language models to label internal neurons with open-ended concepts without the need for any labeled data or human examples, which are required for existing tools to succeed. We show that **CLIP-Dissect** provides more accurate descriptions than existing methods for last layer neurons where the ground-truth is available as well as qualitatively good descriptions for hidden layer neurons. In addition, our method is very flexible: it is model agnostic, can easily handle new concepts and can be extended to take advantage of better multimodal models in the future. Finally **CLIP-Dissect** is computationally efficient and can label all neurons from five layers of ResNet-50 in just four minutes.

## 1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated unprecedented performance in various machine learning tasks spanning computer vision, natural language processing and application domains including healthcare and autonomous driving. However, due to their complex structures, it has been challenging to understand why and how DNNs achieve such great success across numerous tasks and domains. Understanding how the trained DNNs operate is essential to trust their deployment in safety-critical tasks and can help reveal important failure cases or biases of a given model.

One way to achieve these goals is to inspect the functionality of individual neurons in the DNNs, which is the focus of our work. This includes methods based on manual inspection Erhan et al. (2009); Zeiler & Fergus (2014); Zhou et al. (2015); Olah et al. (2017; 2020); Goh et al. (2021), which provide high quality explanations and understanding of the network but require large amounts of manual effort. To address this issue, researchers have developed automated methods to evaluate the functionality of individual neurons, such as Network Dissection Bau et al. (2017) and Compositional Explanations Mu & Andreas (2020). In Bau et al. (2017), the authors first created a new dataset named *Broden* with dense labels associated with a pre-determined set of concepts, and then use *Broden* to find neurons whose activation pattern matches with that of a pre-defined concept. In Mu & Andreas (2020), the authors further extend Network Dissection to detect more complex concepts that are logical compositions of the concepts in *Broden*. Although these methods based on Network Dissection can provide accurate labels in some cases, they have a few major limitations: (1) they require a densely annotated dataset, which is expensive and requires significant amount of human labor to collect; moreover, the dataset may not cover the relevant concepts for all networks; (2) they can only detect concepts from the fixed concept set that is difficult to expand, as new (densely labelled) data is required for each new concept.

To address the above limitations, we propose **CLIP-Dissect**, a novel method to automatically dissect DNNs with *unrestricted* concepts *without* the need of any concept labeled data. Our method is training-free and does not require collecting concept annotated data by leveraging the architecture paradigm of multi-modal model Radford et al. (2021) to efficiently identify the functionality of individual neuron units. We show that **CLIP-Dissect** is $10\times$ more computationally efficient than existing methods and is more accurate at labeling final layer neurons where we know the ground truth. Finally, we show how we can use **CLIP-Dissect** to better understand neural networks and discover that neurons connected by a high weight usually represent similar concepts.

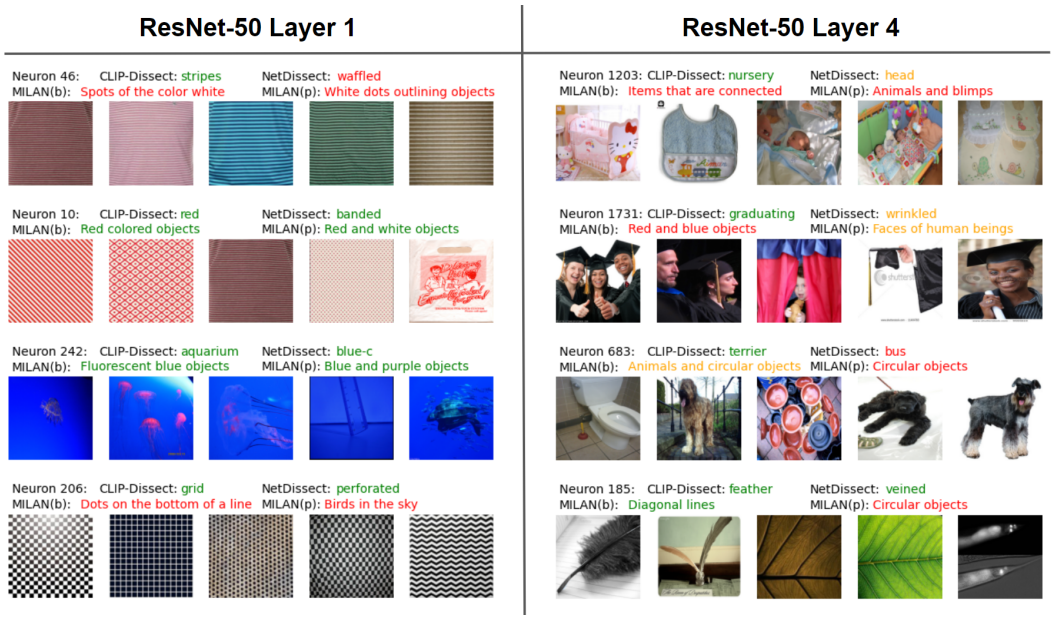



Figure 1: Labels generated by our method **CLIP-Dissect**, Network DissectionBau et al. (2017) and MILANHernandez et al. (2022) for random neurons of ResNet-50 trained on ImageNet. Displayed together with 5 most highly activating images for that neuron. We have subjectively colored the descriptions green if they match these 5 images, yellow if they match but are too generic and red if they do not match. Following *torchvision* Marcel & Rodriguez (2010) naming scheme where layer4 is the second to last layer and layer 1 is the end of first residual block.

## 2 Background and Related Work

**Network dissection.** Network dissection Bau et al. (2017) is the first work on automatically understanding DNNs by inspecting the functionality (described as *concepts*) of each individual neuron[1]. They formulate the problem of identifying concepts of intermediate neurons as a task of matching the pattern of neuron activations to the pattern of a pre-defined concept label mask. In order to define the ground-truth concept label mask, the authors build an auxiliary densely-labeled dataset named *Broden*, which is denoted as $\mathcal{D}_{\text{Broden}}$. The dataset contains a variety of pre-determined concepts $c$ and images $x_i$ with their associated pixel-level labels. Each pixel of images $x_i$ is labeled with a set of relevant concept $c$, which provides a ground-truth binary mask $L_c(x_i)$ for a specific concept $c$. Based on the ground-truth concept mask $L_c(x_i)$, the authors propose to compute the intersection over union score (IoU) between $L_c(x_i)$ and the binarized mask $M_k(x_i)$ from the activations of the concerned neuron unit $k$ over all the images $x_i$ in $\mathcal{D}_{\text{Broden}}$:

$$\text{IoU}_{k,c} = \frac{\sum_{x_i \in \mathcal{D}_{\text{Broden}}} M_k(x_i) \cap L_c(x_i)}{\sum_{x_i \in \mathcal{D}_{\text{Broden}}} M_k(x_i) \cup L_c(x_i)}.$$

If $\text{IoU}_{k,c} > \eta$, then the neuron $k$ is identified to be detecting concept $c$. In Bau et al. (2017), the authors set the threshold $\eta$ to be 0.04. Note that the binary mask $M_k(x_i)$ are computed via thresholding the spatially scaled activation $S_k(x_i) > \xi$, where $\xi$ is the top $0.5\%$ largest activations for the neuron $k$ over all images $x_i \in \mathcal{D}_{\text{Broden}}$ and $S_k(x_i)$ has the same resolution as the pre-defined concept masks by interpolating the original neuron activations $A_k(x_i)$.

**MILAN.** MILAN Hernandez et al. (2022) is a contemporary automated neuron labeling method addressing the issue of being restricted to detect predefined concepts. They can generate unrestricted description of neuron function by training a generative images to text model. The approach of Hernandez et al. (2022) is technically very different from ours as they frame the problem as learning to caption the set of most highly activating images for a given neuron. Their method works by

[1]We follow previous work and use "neuron" to describe a channel in CNNs.

collecting a dataset of human annotations for the set of highly activating images of a neuron, and then training a generative network to predict these human captions. MILAN requires and relies on collecting this curated labeled data set, which limits its capabilities when applied to machine learning tasks outside this dataset. In contrast our method does not require any labeled data and is *training-free*.

**CLIP.** CLIP stands for Contrastive Language-Image Pre-training Radford et al. (2021), an efficient method of learning deep visual representations from natural language supervision. CLIP is designed to address the limitation of static softmax classifiers with a new mechanism to handle *dynamic* output classes. The core idea of CLIP is to enable learning from practically unlimited amounts of raw text, image pairs by training an image feature extractor (encoder) $E_I$ and a text encoder $E_T$ simultaneously. Given a batch of $N$ image $x_i$ and text $t_i$ training example pairs denoted as $\{(x_i, t_i)\}_{i\in[N]}$ with $[N]$ defined as the set $\{1, 2, \dots, N\}$, CLIP aims to increase the similarity of the $(x_i, t_i)$ pair in the embedding space as follows. Let $I_i = E_I(x_i), T_i = E_T(t_i)$, CLIP maximizes the cosine similarity of the $(I_i, T_i)$ in the batch of $N$ pairs while minimizing the cosine similarity of $(I_i, T_j), j \neq i$ using a multi-class N-pair loss Sohn (2016); Radford et al. (2021). Once the image encoder $E_I$ and the text encoder $E_T$ are trained, CLIP can perform zero-shot classification for any set of labels: given a test image $x_1$ we can feed in the natural language names for a set of $M$ labels $\{t_j\}_{j\in[M]}$. The predicted label of $x_1$ is the label $t_k$ that has the largest cosine similarity among the embedding pairs: $(I_1, T_k)$.

## 3 METHOD

In this section, we describe **CLIP-Dissect**, a novel method for automatic, flexible and generalizable neuron label generation for vision networks from popular convolutional neural networks (CNNs) to STOA vision transformers (ViT). An overview of **CLIP-Dissect** is illustrated in Figure 2 and a comprehensive simulations results for CNNs can be found in Section 4, 5 and Appendix A.4 for ViT. We first give an overview of the full pipeline of **CLIP-Dissect** in Sec 3.1 and derive a few theoretically inspired similarity functions based on information theory for **CLIP-Dissect** in Sec 3.2. We discuss and compare **CLIP-Dissect** with other method in Sec 3.3.

### 3.1 CLIP-DISSECT OVERVIEW

**Inputs & Outputs.** There are 3 inputs of the **CLIP-Dissect** algorithm: **(i)** DNN to be dissected/probed, denoted as $f(x)$, **(ii)** dataset of DNN inputs for dissecting the DNN, denoted as $\mathcal{D}_{\text{probe}}$ where $|\mathcal{D}_{\text{probe}}| = N$, **(iii)** concept set, denoted as $\mathcal{S}$, $|\mathcal{S}| = M$.

The output of **CLIP-Dissect** is the neuron labels, which identify the concept associated with each individual neuron. Compared with Network Dissection Bau et al. (2017), our goals are the same – we both want to inspect and detect concepts associated with each neuron. The input **(i)** is also the same, as we both want to dissect the DNN $f(x)$; however, the inputs **(ii)** and **(iii)** have stark differences. Specifically, in **CLIP-Dissect**, our $\mathcal{D}_{\text{probe}}$ does not require any concept labels and thus can be any publicly available dataset such as CIFAR-100, ImageNet, a combination of datasets or unlabeled images collected from the internet. On the other hand, Network Dissection can only use a $\mathcal{D}_{\text{probe}}$ that has been densely labeled with the concepts from the concept set $\mathcal{S}$. As a result, users of Network Dissection are limited to $\mathcal{D}_{\text{probe}}$ and the fixed concept set $\mathcal{S}$ in Broden unless they are willing to create their own densely labeled dataset. This is a major limitation of Network Dissection Bau et al. (2017) and its follow-up works Mu & Andreas (2020). In contrast, the concept set $\mathcal{S}$ and probing dataset $\mathcal{D}_{\text{probe}}$ in our framework are *decoupled* – we can use any text corpus to form the concept set $\mathcal{S}$ and any image dataset independent of $\mathcal{S}$ in **CLIP-Dissect**.

**Algorithm.** There are 3 key steps in **CLIP-Dissect**:

1. *Compute the concept-activation matrix $P$.* Using the image encoder $E_I$ and text encoder $E_T$ of a CLIP model, we first compute the text embedding $T_i$ of the concepts $t_i$ in the concept set $\mathcal{S}$ and the image embedding $I_i$ of the images $x_i$ in the probing dataset $\mathcal{D}_{\text{probe}}$. Next, we calculate the concept-activation matrix $P$ whose $(i, j)$-th element is the inner product $I_i \cdot T_j$, i.e. $P_{i,j} = I_i \cdot T_j$.

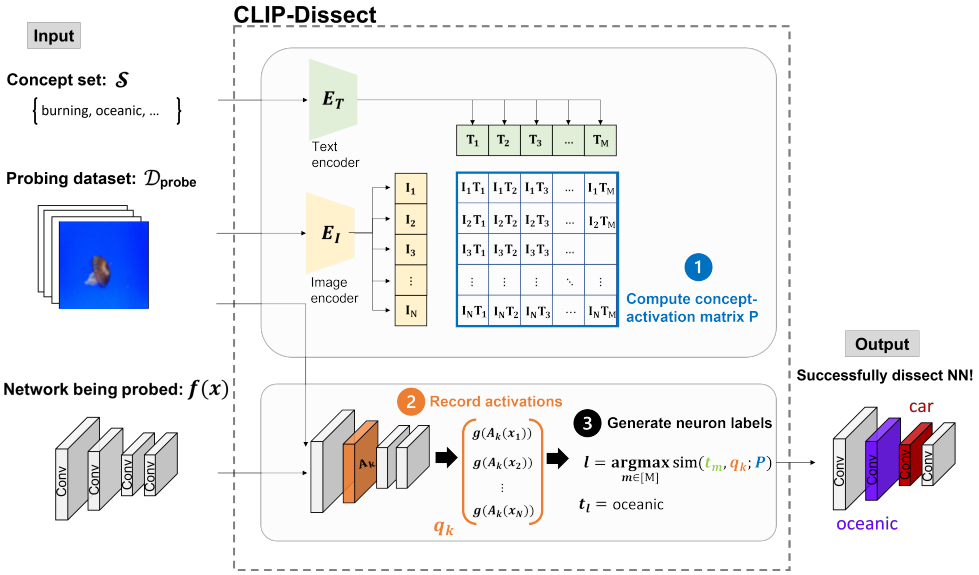



Figure 2: Overview of **CLIP-Dissect**: a 3-step algorithm to dissect neural network of interest.

2. *Record activations of target neurons.* Given the neuron unit $k$, compute the activation $A_k(x_i)$ of the $k$-th neuron for every image $x_i \in \mathcal{D}_{\text{probe}}$. Define a summary function $g$, which takes the activation map $A_k(x_i)$ as input and returns a real number. Here we let $g$ be the mean function that computes the mean of the activation map over spatial dimensions. We record $g(A_k(x_i))$, for all $i, k$.
3. *Generate the neuron labels.* Given a neuron unit $k$, the concept label for $k$ is determined by calculating the most similar concept $t_m$ with respect to its activation vector $\mathbf{q}_k = [g(A_k(x_1)), \ldots, g(A_k(x_N))]^\top$ and the similarity function $\texttt{sim}$ is defined as $\texttt{sim}(t_m, q_k; P)$. In other words, the label of neuron $k$ is $t_l$, where $l = \arg\max_m \texttt{sim}(t_m, q_k; P)$. Below we discuss different ways to define $\texttt{sim}$.

### 3.2 Similarity function

There are many ways to design the similarity function $\texttt{sim}$, and this choice has a large effect on the performance of our method. In particular, simple functions like cosine similarity perform poorly, likely because they place too much weight on the inputs that don't activate the neuron highly. We focus on the following 4 similarity functions and compare their results in the Table 3:

- **Cos**: Cosine similarity between activation vectors of the target neuron $q_k$ and CLIP with corresponding the label $P_{:,m}$.

$$\texttt{sim}(t_m, q_k; P) \triangleq \frac{P_{:,m}^\top q_k}{||P_{:,m}|| \cdot ||q_k||} \tag{1}$$

- **Rank reorder**: This function calculates the similarity between $q_k$ and $P_{:,m}$ by creating a vector $q'_k$, which is a reordered version of $q_k$. This is done by replacing the $i$-th largest element of $P_{:,m}$ by the $i$-th largest element of $q_k$ for all $i$. This essentially reorders the elements of $q_k$ according to the ranks of the elements in $P_{:,m}$. Similarity function is then defined as:

$$\texttt{sim}(t_m, q_k; P) \triangleq -\|q'_k - q_k\|_p \tag{2}$$

- **WPMI** (**W**eighted **P**ointwise **M**utual **I**nformation): We propose a mathematically grounded idea to derive $\texttt{sim}$ based on mutual information, where the label of a neuron is defined as the concept that maximizes the mutual information between the set (denoted as $B_k$) of most highly activated images on neuron $k$ and the label $t_m$. Specifically:

$$\texttt{sim}(t_m, q_k; P) \triangleq \text{wpmi}(t_m, q_k) = \log p(t_m | B_k) - \lambda \log p(t_m), \tag{3}$$

where $p(t_m|B_k) = \Pi_{x_i \in B_k} p(t_m|x_i)$ and $\lambda$ is a hyperparameter.

- **SoftWPMI**: Finally, we propose a generalization of WPMI where we use the probability $p(x \in B_k)$ to denote the chance an image $x$ belongs to the example set $B_k$. Standard WPMI corresponds to the case where $p(x \in B_k)$ is either 0 or 1 for all $x \in D_{probe}$ while SoftWPMI relaxes the binary setting of $p(x \in B_k)$ to real values between 0 and 1. This gives us the following function:

$$\mathtt{sim}(t_m, q_k; P) \triangleq \text{soft_wpmi}(t_m, q_k) = \log \mathbb{E}[p(t_m|B_k)] - \lambda \log p(t_m) \tag{4}$$

where we compute $\log \mathbb{E}[p(t_m|B_k)] = \log(\Pi_{x \in D_{probe}}[1 + p(x \in B_k)(p(t_m|x) - 1)])$. As shown in our experiments (Table 3), we found SoftWPMI give the best results among the three and thus we use it for most of our experiments.

Due to page constraint, we leave the derivation and details on how to calculate WPMI and SoftWPMI using only CLIP products of the matrix $P$ as well as our hyperparameter choices to Appendix A.1.

### 3.3 Compability with future models

The current version of our algorithm relies on the CLIP Radford et al. (2021) multimodal model. However, this doesn't have to be the case, and developing improved CLIP-like models has received a lot of attention recently, with many recent work reporting better results with an architecture similar to CLIP Yu et al. (2022); Yuan et al. (2021); Zhai et al. (2022); Pham et al. (2021). If these models are eventually released publicly we can directly replace CLIP with a better model without any changes to our algorithm. As a result our method will improve over time as general ML models get more powerful, while existing work Bau et al. (2017) Hernandez et al. (2022) can't really be improved without collecting a new dataset specifically for that purpose.

## 4 Experiments

In this section we evaluate our method through analyzing two pre-trained networks: ResNet-50 He et al. (2016) trained on ImageNet Deng et al. (2009), and ResNet-18 trained on Places-365 Zhou et al. (2017). Unless otherwise mentioned we use 20,000 most common English words[2] as the concept set $\mathcal{S}$.

### 4.1 Qualitative results

Figure 1 shows examples of neuron labels generated by **CLIP-Dissect** for randomly chosen hidden neurons in different layers compared against the label assigned to those same neurons by Network

[2]Source: https://github.com/first20hours/google-10000-english/blob/master/20k.txt

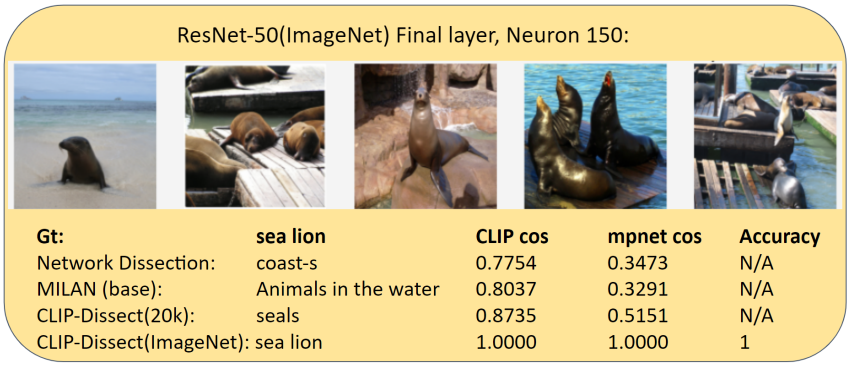



Figure 3: Example of a final layer neuron: we compare the descriptions generated by different methods and our metrics. Accuracy only evaluated for CLIP-Dissect with ImageNet labels as concept set since it is the only method where exact correct answer is a possible choice and therefore accuracy makes sense.

| Method | $D_{probe}$ | Concept set $\mathcal{S}$ | CLIP cos | mpnet cos |
|---|---|---|---|---|
| Network Dissection (baseline) | Broden | Broden | 0.6929 | 0.2952 |
| MILAN base (baseline) | ImageNet val | - | 0.7080 | 0.2788 |
| **CLIP-Dissect** (Ours) | ImageNet val | Broden | 0.7393 | 0.4201 |
| **CLIP-Dissect** (Ours) | ImageNet val | 3k | 0.7456 | 0.4161 |
| **CLIP-Dissect** (Ours) | ImageNet val | 10k | 0.7661 | 0.4696 |
| **CLIP-Dissect** (Ours) | ImageNet val | 20k | 0.7900 | 0.5257 |
| **CLIP-Dissect** (Ours) | ImageNet val | ImageNet | 0.9766 | 0.9458 |
| **CLIP-Dissect** (Ours) | CIFAR100 train | 20k | 0.7300 | 0.3664 |
| **CLIP-Dissect** (Ours) | Broden | 20k | 0.7407 | 0.3945 |
| **CLIP-Dissect** (Ours) | ImageNet val | 20k | 0.7900 | 0.5257 |
| **CLIP-Dissect**(Ours) | ImageNet val + Broden | 20k | 0.7900 | 0.5233 |

Table 1: The cosine similarity of predicted labels compared to ground truth labels on final layer neurons of ResNet-50 trained on ImageNet. The higher similarity the better. We can see that our method performs better when $D_{probe}$ and concept set are larger and/or more similar to training data.

Dissection Bau et al. (2017) and MILAN Hernandez et al. (2022). We do not compare against Compositional Explanations Mu & Andreas (2020) as it is much more computationally expensive and complementary to our approach as their composition could also be applied to our explanations. We observe that not every neuron corresponds to a clear concept and our method can detect low-level concepts on early layers and provide more descriptive labels than existing methods Bau et al. (2017); Hernandez et al. (2022) in later layers, such as the 'graduating' and 'nursery' neurons. These results use the union of ImageNet validation set and Broden as $D_{probe}$. More quantitavive examples are shown in the Appendix. In general we observe that MILAN sometimes gives very accurate descriptions but often produces descriptions that are too generic or even semantically incorrect (highlighted as red labels), while Network Dissection is good at detecting low level concepts but fails on concepts missing from its dataset. We compared against two versions of MILAN: MILAN base(b) that was trained to describe neurons of networks trained on both ImageNet and Places365, as well as MILAN places365(p) that was only trained on Places365 neurons to test its generalization ability.

### 4.2 Quantitative results

We also quantitatively compare our method's performance against Network Dissection Bau et al. (2017) and MILAN Hernandez et al. (2022) when possible. The key idea of these experiments is to generate labels for neurons where we have access to *ground truth* descriptions, i.e. neurons in the final layer of a network, where the ground truth concept is the name of the corresponding class (class label). This avoids the need for human evaluation and uses real function of the target neurons while human evaluations are usually limited to describing a few most highly activating images. Example evaluation of a final layer neuron can be seen in Figure 3. We mostly focus on below two metrics for measuring the quality of explanations:

a) **Cos similarity:** We measure the cosine similarity in a sentence embedding space between the ground truth class name for the neuron, i.e. sea lion in Figure 3 and the explanation generated by the method. For embeddings we use the CLIP ViT-B/16 text encoder (denoted as CLIP cos) as well as the all-mpnet-base-v2 sentence encoder (denoted as mpnet cos). Example values for the similarities can be seen in Figure 3.

b) **Accuracy:** The second metric we measure is accuracy, which computes the percentage of neurons the method assigns the exact correct label i.e. the class name to. This metric cannot be used for all methods/situations, for example MILAN generates explanations without a concept set so it is unlikely to match the exact wording of the class name. Because of this we only measure accuracy in situations where the method chooses from a concept set that includes the exact correct label, such as Network Dissection for models trained on Places365 (not possible for ImageNet since ImageNet labels are missing from Broden).

| Method | $D_{probe}$ | Uses gt labels | Text set | Top1 Acc | CLIP cos | mpnet cos |
|---|---|---|---|---|---|---|
| NetworkDissection (Baseline) | Broden | Yes | Broden | 43.82% | 0.8887 | 0.6697 |
| **CLIP-Dissect** (ours) | Broden | No | Broden | **58.05%** | **0.9106** | **0.7024** |

Table 2: Performance when labeling final layer neurons of a ResNet18 trained on Places365. Accuracy measured on 267/365 neurons whose label is a directly included in Broden labels.

| | | $D_{probe}$ | | | | |
|---|---|---|---|---|---|---|
| Metric | Similarity fn | CIFAR100 train | Broden | ImageNet val | ImageNet val + Broden | Average |
| mpnet cos similarity | cos | 0.2761 | 0.215 | 0.2823 | 0.2584 | 0.2580 |
| | Rank reorder | 0.3250 | 0.3857 | 0.4901 | 0.5040 | 0.4262 |
| | WPMI | 0.3460 | 0.3878 | **0.5302** | **0.5267** | 0.4477 |
| | SoftWPMI | **0.3664** | **0.3945** | 0.5257 | 0.5233 | **0.4525** |
| Top1 accuracy | cos | 8.50% | 5.70% | 15.9% | 11.4% | 10.38% |
| | Rank reorder | 36.30% | 57.50% | 89.80% | 89.90% | 68.38% |
| | WPMI | 23.80% | 47.10% | 87.00% | 86.90% | 61.20% |
| | SoftWPMI | **46.20%** | **70.50%** | **95.00%** | **95.40%** | **76.78%** |

Table 3: Comparison of the performance between similarity functions. We look at the final layer of ResNet-50 trained on ImageNet (same as Tab 1). We use $\mathcal{S} = 20$k for cosine similarity and $\mathcal{S} =$ ImageNet classes for top1 accuracy. We can see SoftPMI performs best overall.

In Table 1 we can see that the labels generated by our method are closer to ground truth in a sentence embedding space than those of Network Dissection or MILAN regardless of our choice of $D_{probe}$ or concept set $\mathcal{S}$. We can also see using a larger concept set (e.g. 3k v.s. 20k) improves the performance of our method. Table 2 shows that our method outperforms Network Dissection even on a task that is favorable to their method as the Places365 dataset has large overlaps with Broden. We want to highlight that we can reach higher accuracy even though Network Dissection has access to and relies on the ground truth labels in Broden while ours does not.

### 4.3 Choice of similarity function

Table 3 compares the performance of different similarity functions used in **CLIP-Dissect**. We use accuracy and cos similarity in embedding space as defined in Sec 4.2 to measure the quality of descriptions. We observed that SoftPMI performs the best and thus it is used in all other experiments unless otherwise mentioned. Table 3 also showcases how **CLIP-Dissect** can give final layer neurons the correct label with a very impressive 95% accuracy.

### 4.4 Computational efficiency

Table 4 shows the runtime of different automated neuron labeling methods when tasked to label all the neurons of five layers in ResNet-50. We can see our method runs in just 4 minutes, more than 10 times faster than the second most efficient method MILAN Hernandez et al. (2022) which takes 55.5 min.

### 4.5 Detecting concepts missing from $D_{probe}$

One surprising ability we found is that our method is able to assign the correct label to a neuron even if $D_{probe}$ does not have any images corresponding to that concept. For example, **CLIP-Dissect** was able to assign the correct dog breed to 46 out of 118 neurons detecting dog breeds, and correct bird

| Method | CLIP-Dissect | Network Dissection | Compositional Explanations | MILAN |
|---|---|---|---|---|
| Runtime | **3min50s** | >4 hrs | >>14 hours | 55min 30s |

Table 4: The time it takes to describe the layers ['conv1', 'layer1', 'layer2', 'layer3', 'layer4'] of ResNet-50 via different methods using our hardware(Tesla P100 GPU).We can see CLIP-Dissect is much more computationally efficient than existing methods.

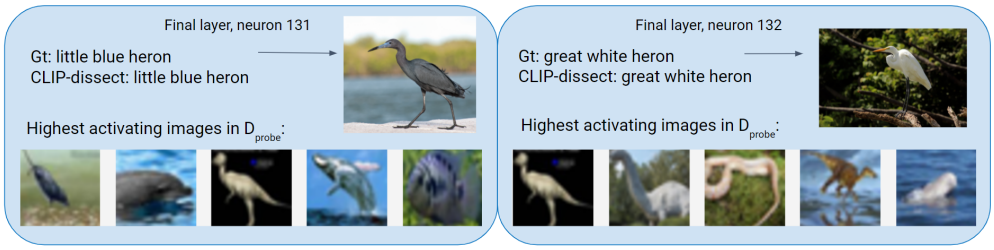



Figure 4: Example of **CLIP-Dissect** correctly labeling neurons that detect the little blue heron and the great white heron based on pictures of dolphins and dinosaurs in CIFAR. Note that CIFAR100 does not contain any bird images but **CLIP-Dissect** can still get correct concept.

species to 22 out of 59 final layer neurons of ResNet-50 trained on ImageNet, while using CIFAR-100 training set as $D_{probe}$, which doesn't include any images of dogs or birds. This is fundamentally impossible for any label based methods like NetDissect Bau et al. (2017) and Compositional neuron Mu & Andreas (2020) (as IoU will be 0 for any concept not in $D_{probe}$), and unlikely for methods based on captioning highly activated images like MILAN Hernandez et al. (2022) (as humans won't assign a captions missing from images). Example labels and highest activating probe images can be seen in Figure 4.

### 4.6 Compositional Concepts

So far our method has focused on choosing the most fitting concept from the pre-defined concept set. While changing the concept set in **CLIP-Dissect** is as easy as editing a text file, we show it can also detect more complex compositional concepts. We experimented with generating explanations by searching over text concatenations of two concepts in our concept space. To reduce computational constraints, we only looked at combinations of 100 most accurate single word labels for each neuron. Example results are shown in Fig 5. While the initial results are promising, some challenges remain to make these compositional explanations more computationally efficient and consistent, which is an important direction for future work.

## 5 Use case of **CLIP-Dissect**

In this section, we present a simple experiment to showcase how we can use **CLIP-Dissect** to gain new insights on neural networks. By inspecting the ResNet-50 network trained on ImageNet with

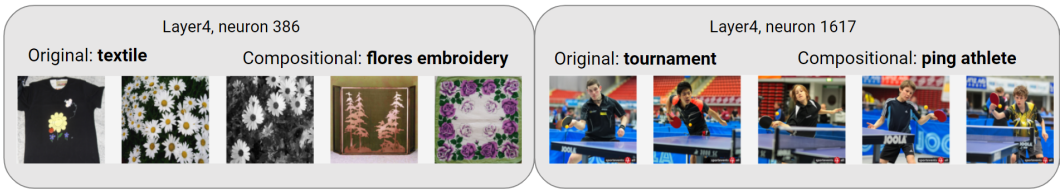
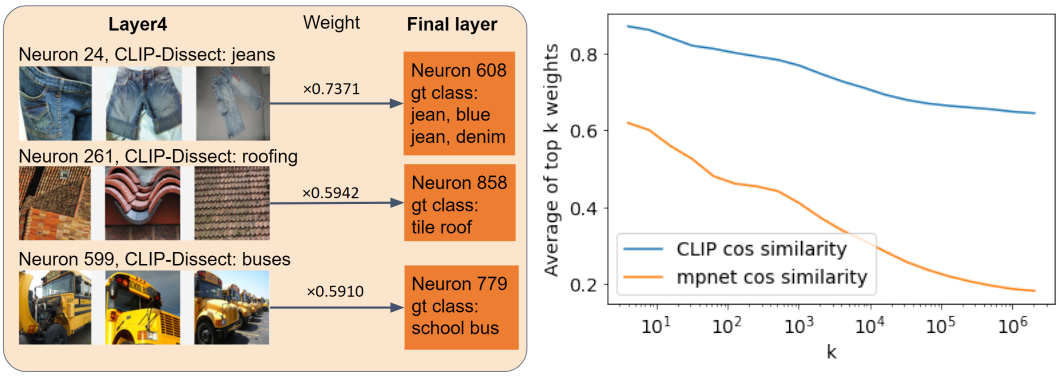



Figure 5: An example of compositional explanations generated by our method for two neurons of ResNet50 trained on ImageNet.




(a) Visualization of 3 highest weights of final layer.

(b) Average cosine similarity between concepts.

Figure 6: a) 3 highest weights of the final layer of ResNet-50 trained on ImageNet, we can see neurons connected by the highest weights are detecting very much the same concept. b) Cosine similarities between the concepts of neurons connected by highest weights. The higher the weight between neurons, the more similar a concept they represent.

**CLIP-Dissect**, we discover the following phenomenon and evidence for it: **the higher the weight between two neurons, the more similar concepts they encode**, as shown in Figure 6. This makes sense since a high positive weight causally makes the neurons activate more similarly, but the extent of this correlation is much larger than we expected, as each final layer neuron has 2048 incoming weights so we would not expect any single weight to have that high of an influence. A consequence of the similarity in concepts is that the second to last layer already encodes quite complete representations of certain final layer classes in individual neurons, instead of the representation for these classes being spread across multiple neurons. For example Fig 6a shows that the 3 neurons with highest outgoing weights already seem to be accurately detecting the final layer concept/class label they're connected to.

To make these results more quantitative, we perform the experiment of Figure 6b, where we measure the similarity of concepts encoded by the neurons connected by highest weights in the final layer of ResNet-50 trained on ImageNet. For layer4 neurons we used CLIP-Dissect to come up with their concept, while for output neurons we used ground truth i.e. class label in text form. We can clearly see that higher weights connect more similar concepts together, and the average similarity decreases exponentially as a function of k when averaging similarities of top k weights. To further test this relationship, we found that the mpnet cos similarity between concepts encoded by two neurons and the weight connecting them are correlated with $r = 0.120$ and p-value $< 10^{-300}$(so the probability of no correlation is practically zero) when calculated over all 2 million weights in the final layer. If we only look at the highest 50000 weights the correlation is even higher with $r = 0.258$, p-value $< 10^{-300}$.

## 6 Conclusions

In this work, we have developed **CLIP-Dissect**, a novel, flexible and computationally efficient framework for generating automated labels for hidden layer neurons. We also proposed new methods to quantitatively compare neuron labeling methods based on labeling final layer neurons. Importantly, we have shown **CLIP-Dissect** can outperform previous automated labeling methods both qualitatively and quantitatively and can even detect concepts missing from the probing dataset. Finally we used **CLIP-Dissect** to discover that neurons connected by a high weight often represent very similar concepts.

## 7 REPRODUCIBILITY

We have described our algorithms and hyperparameters in sufficient detail for reproduction in sections 3 and A.1. In addition we will make all code publicly available prior to publication.

## REFERENCES

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2022.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

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

Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V Le. Combined scaling for zero-shot transfer learning. *arXiv preprint arXiv:2111.10050*, 2021.

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

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18123–18133, 2022.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

# A APPENDIX

## A.1 SIMILARITY FUNCTION DETAILS AND DERIVATION

**Rank reorder hyperparameters:**

The results of Table 3 are using top 5% of most highly activating images for each neuron and using $p = 3$ for the $l_p$-norm.

**WPMI:**

In this section, we show that one choice of similarity function $\texttt{sim}(t_m, q_k; P)$ can be derived based on the weighted point-wise mutual information (wpmi). Note that wpmi is also used in Hernandez et al. (2022) but in a different way – our approach can compute wpmi directly from the CLIP products $P$ and does not require any training, while Hernandez et al. (2022) train two models to estimate wpmi.

To start with, by definition, the wpmi between a concept $t_m$ and the most highly activated images $B_k$ of neuron $k$ can be written as

$$\text{wpmi}(t_m, q_k) = \log p(t_m|B_k) - \lambda \log p(t_m) \tag{5}$$

Here $B_k$ is the set of images that most highly activates neuron $k$, i.e. the top indices of $q_k$. First we can compute $p(t_m|x_i) = \text{softmax}(aP_{i,:})_m$, where $\text{softmax}(z)_n = \frac{e^{z_n}}{\sum_{j=1}^{N} e^{z_j}}$ with $z \in \mathbb{R}^N$, $P_{i,:}$ is the ith row vector of the concept-activation matrix $P$ and $a$ is a scalar temperature constant. This is the probability that CLIP assigns to a concept $t_m$ for image $x_i$ when used as a classifier.

We then define $p(t_m|B_k)$ as the probability that all images in $B_k$ have the concept $t_m$, which gives us $p(t_m|B_k) = \Pi_{x_i \in B_k} p(t_m|x_i)$. Thus, we have

$$\log p(t_m|B_k) = \sum_{x_i \in B_k} \log p(t_m|x_i) \tag{6}$$

which is the 1st term in Eq (5). Next, we can approximate the 2nd term $p(t_m)$ in Eq (5) as follows: $p(t_m)$ is the probability that a random set of images $B$ will be described by $t_m$. Since we don't know the true distribution for a set of images, an efficient way to approximate this is to average the probability of $t_m$ over the different neurons we are probing. This can be described by the following equation:

$$p(t_m) = \mathbb{E}_B[p(t_m|B)] \approx \frac{\sum_{j \in C} p(t_m|B_j)}{|C|} = \frac{\sum_{j \in C} \Pi_{x_i \in B_j} p(t_m|x_i)}{|C|} \tag{7}$$

where $C$ is the set of neurons in the layer we are probing. Thus we can plug Eq. (6) and Eq. (7) in to Eq. (5) to compute wpmi through the CLIP model:

$$\text{wpmi}(t_m, q_k) = \sum_{x_i \in B_k} \log p(t_m|x_i) - \lambda \log \left( \sum_{j \in C} \Pi_{x_i \in B_j} p(t_m|x_i) \right) + \lambda \log |C| \tag{8}$$

So we can use the above Eq (8) in our CLIP-Dissect and set $\texttt{sim}(t_m, q_k; P) = \text{wpmi}(t_m, q_k)$ in the algorithm.

For our experiments we use $a = 2$, $\lambda = 0.6$ and top 28 most highly activating images for neuron $k$ as $B_k$ which were found to give best quantitave results when describing final layer neurons of ResNet-50.

**SoftWPMI:**

SoftWPMI is an extension of wpmi as defined by Eq. (8) into settings where we have uncertainty over which images should be included in the example set $B_k$. In WPMI the size of example set is defined beforehand, but it is not clear how many images should be included, and this could vary from neuron to neuron. In this description, we assume that there exists a true $B_k$ which includes images from $D_{probe}$ if and only if they represent the concept of neuron $k$. We then define binary indicator random variables $X_i^k = \mathbb{1}[x_i \in B_k]$ which take value 1 if the ith image is is in set the $B_k$, and we define $X^k = \{X_1^k, ..., X_M^k\}$.

Our derivation begins from the observation that we can rewrite $p(t_m|B_k)$ from above as:

$$p(t_m|B_k) = \Pi_{x_i \in B_k} p(t_m|x_i) = \Pi_{x_i \in D_{probe}} p(t_m|x_i)^{\mathbb{1}[x_i \in B_k]} = \Pi_{x_i \in D_{probe}} p(t_m|x_i)^{X_i^k} \tag{9}$$

Now:

$$\mathbb{E}_{X_i^k}[p(t_m|x_i)^{X_i^k}] = p(x_i \in B_k)p(t_m|x_i) + (1 - p(x_i \in B_k)) = 1 + p(x_i \in B_k)(p(t_m|x_i) - 1) \tag{10}$$

If we assume the $X_i^k$ are statistically independent, we can write:

$$\mathbb{E}_{X^k}[p(t_m|B_k)] = \Pi_{x_i \in D_{probe}} \mathbb{E}_{X_i^k}[p(t_m|x_i)^{X_i^k}] = \Pi_{x_i \in D_{probe}}[1 + p(x_i \in B_k)(p(t_m|x_i) - 1)] \tag{11}$$

$$\Rightarrow \log \mathbb{E}_{X^k}[p(t_m|B_k)] = \sum_{x_i \in D_{probe}} \log(1 + p(x_i \in B_k)(p(t_m|x_i) - 1)) \tag{12}$$

Note Equation (10) goes to 1 if $p(x_i \in B_k) = 0$ (i.e. no effect in a product) and to $p(t_m|x_i)$ if $p(x_i \in B_k) = 1$. So Eq. (12) reduces to Eq. (6) of standard WPMI if $p(x_i \in B_k)$ is either 1 or 0 for all $x_i \in D_{probe}$. In other words, we are considering a "soft" membership in $B_k$ instead of "hard" membership of standard WPMI.

To get the second term for wpmi, $p(t_m)$, i.e. probability that text $t_m$ describes a random example set $B_k$, we can approximate it like we did in Eq. (7) this using the example sets for other neurons we are interested in.

$$p(t_m) = \mathbb{E}_{B_i}[\mathbb{E}_{X^i}[p(t_m|B_i)]] \approx \frac{\sum_{j \in C} \mathbb{E}_{X^j}[p(t_m|B_j)]}{|C|}$$

$$\rightarrow \frac{\sum_{j \in C} \mathbb{E}_{X^j}[p(t_m|B_j)]}{|C|} = \frac{\sum_{j \in C} \Pi_{x \in D_{probe}}[1 + p(x \in B_j)(p(t_m|x) - 1)]}{|C|} \tag{13}$$

Finally, we can compute full softWPMI with Eq. (12) and Eq. (13) and use it as similarity function in **CLIP-Dissect**:

$$\text{soft_wpmi}(t_m, q_k) = \sum_{x_i \in D_{probe}} \log(1 + p(x_i \in B_k)(p(t_m|x_i) - 1))$$

$$-\lambda \log \left( \sum_{j \in C} \Pi_{x \in D_{probe}}[1 + p(x \in B_j)(p(t_m|x) - 1)] \right) + \lambda \log |C| \tag{14}$$

One thing we haven't yet discussed is the choice of $p(x \in B_k)$. There is flexibility and this probability could be derived from the activations of neuron $k$ on image $x$, by for example by taking a scaled sigmoid, or it could be based on the ranking of the image.

For our experiments we found ranking based probability to perform the best, and used $p(x \in B_k)$ linearly decreasing from 0.998 of the most highly activating image for neuron $k$ to 0.97 for 100th most highly activating image and 0 for all other images. Thus in practice we only have to use the 100 images when calculating SoftWPMI instead of full $D_{probe}$ which is much more computationally efficient. For other hyperparameters we used $a = 10$ and $\lambda = 1$.

### A.2 Limitations

The main limitation of our method compared to previous work is that it's not taking advantage of the spatial information of neuron activations. This causes some difficulties in detecting lower level

concepts, but we are still able to detect many low level/localized patterns as discussed in detail in Section A.3. Secondly, our method currently works well only on concepts and images that CLIP works well on, and while this is a larger set of tasks than other existing neuron labeling methods can be applied on, **CLIP-Dissect** may not work out of the box on networks trained on tasks that require highly specific knowledge such as classifying astronomical images. However, our method is compatible with future large vision-language models as long as they share a similar structure to CLIP which will likely be even more general, CLIP-like models trained for a specific target domain.

### A.3 Low level concepts

In this section we show additional results of probing low level concepts, using two networks trained on ImageNet, ResNet-50 as well as ResNet-152. For ResNet-152 we also compare against human annotations for these neurons from MILAnnotations (Hernandez et al., 2022).

The results for ResNet-152 can be seen in Figure 9. We can see CLIP-Dissect is able to accurately detect many lower level concepts, such as colors in Conv1 neurons 1,3,10 and Layer1 neuron 4, as well as detecting that neuron 3 of Layer1 activates specifically for the *text/label*, without having access to the activation pattern, while MILAN fails to detect this.

However, CLIP-Dissect does also have some failure modes on lower level patterns.

- **Failure to differentiate between concept and correlated objects:** This leads to higher level outputs than desired. For example: conv1 neuron 16 which detects the concept *underwater*, while the true concept is probably more similar to *blue*(human annotators also made this mistake), or neuron 28 where CLIP-Dissect outputs *zebra* while the neuron is likely just detecting *stripes*. The worst example of this is neuron conv1 neuron 24, which simply activates on white background, but this is entirely missed by CLIP-Dissect as it's not good at focusing on the background.
- **Unintrepretable neurons:** Some neurons seem to not be interpretable, i.e. Conv1 neuron 2. CLIP-Dissect outputs *music* which seems incorrect, but neither human annotators or MILAN were able to assign a clear concept for them.

Similar observations also hold for the ResNet-50 model as can be seen in Figure 10.

### A.4 Vision Transformer

Since our method does not rely on the specifics of CNNs in its operation, we can easily extend it to work on different architectures, such as the Vision Transformer, specifically ViT-B/16 (Dosovitskiy et al., 2020) model trained on ImageNet. We have visualized highly activating neurons and their descriptions in Figure 11. Interestingly we found the highly interpretable neurons to be very location focused, i.e. *kitchens* or *highways* despite the network being trained on object level labels(ImageNet).

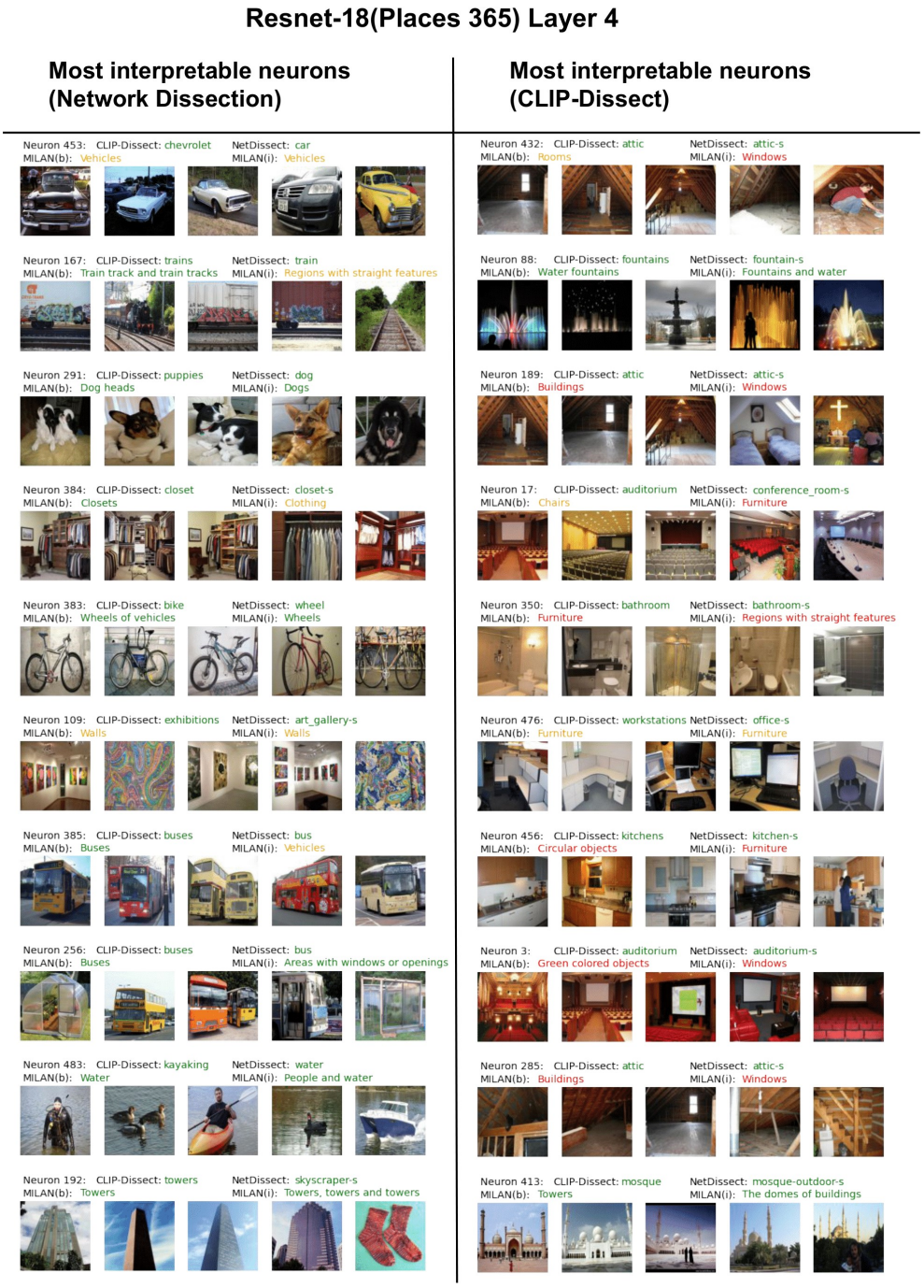



Figure 7: Explanations of most interpretable neurons in the second to last layer of ResNet-18 trained on Places365. Displayed together with 5 most highly activating images for that neuron. We have subjectively colored the descriptions green if they match these 5 images, yellow if they match but are too generic and red if they do not match. Both Network Dissection and **CLIP-Dissect** do very well while MILAN struggles to explain some neurons. MILAN(b) is trained on both ImageNet and Places365 networks, while MILAN(i) is only trained on ImageNet. Both MILAN networks perform similarly here but the ImageNet version misses/is too generic for more neurons, such as labeling a bus neuron as "vehicles". The neurons on the left have highest IoU according to MILAN while neurons on the right have highest similarity to the concept according to our similarity function.

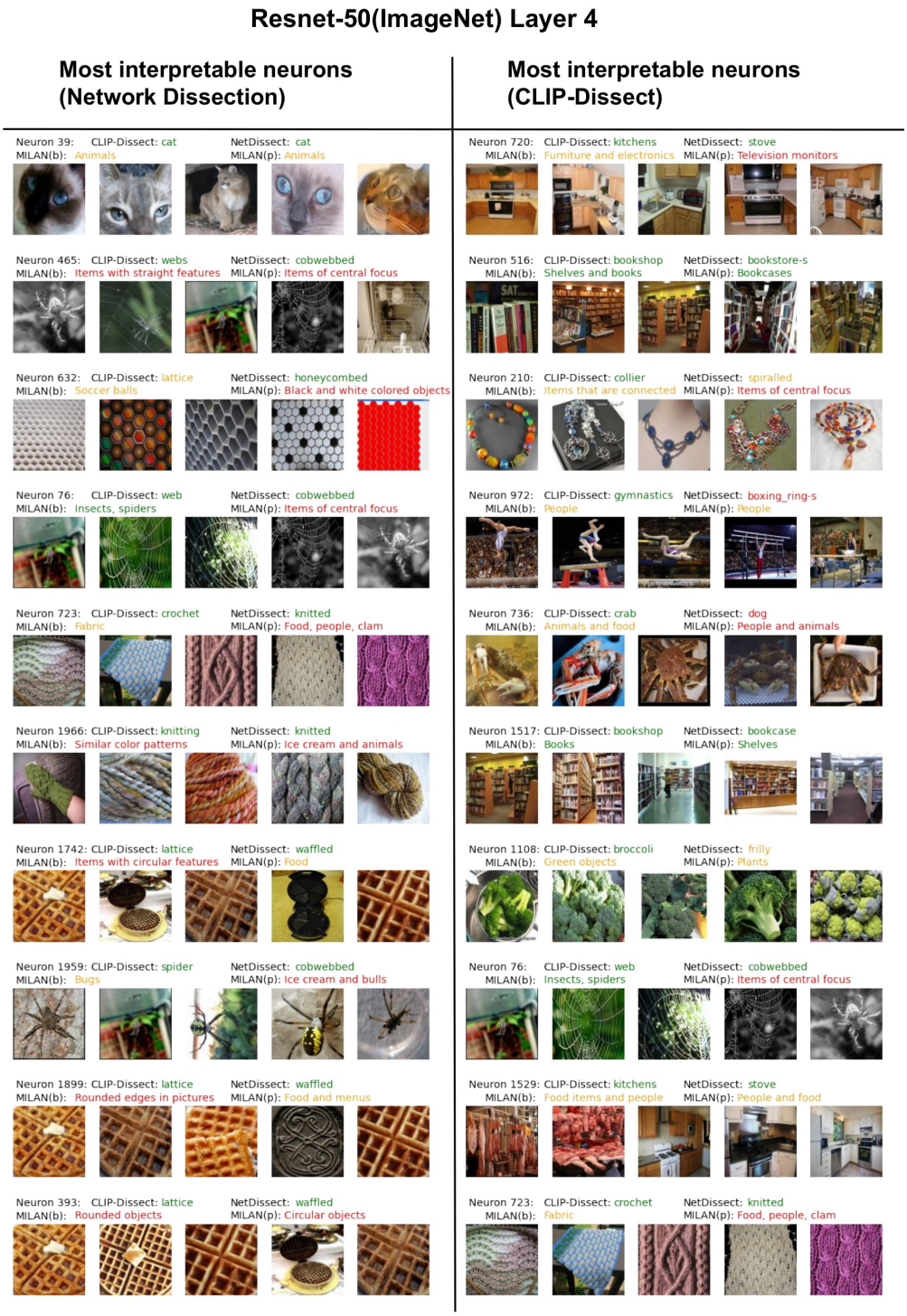



Figure 8: Explanations of most interpretable neurons in the second to last layer of ResNet-50 trained on ImageNet. Displayed together with 5 most highly activating images for that neuron. We have subjectively colored the descriptions green if they match these 5 images, yellow if they match but are too generic and red if they do not match. Both **CLIP-Dissect** and Network Dissection perform well on these most interpretable neurons except for a few failures by Network Dissection, while MILAN often gives concepts that are too generic. MILAN(b) is trained on both ImageNet and Places365 networks, while MILAN(p) is only trained on Places365. We can see the Places trained model is struggling more with concepts like spiders, indicating issues with generalization.

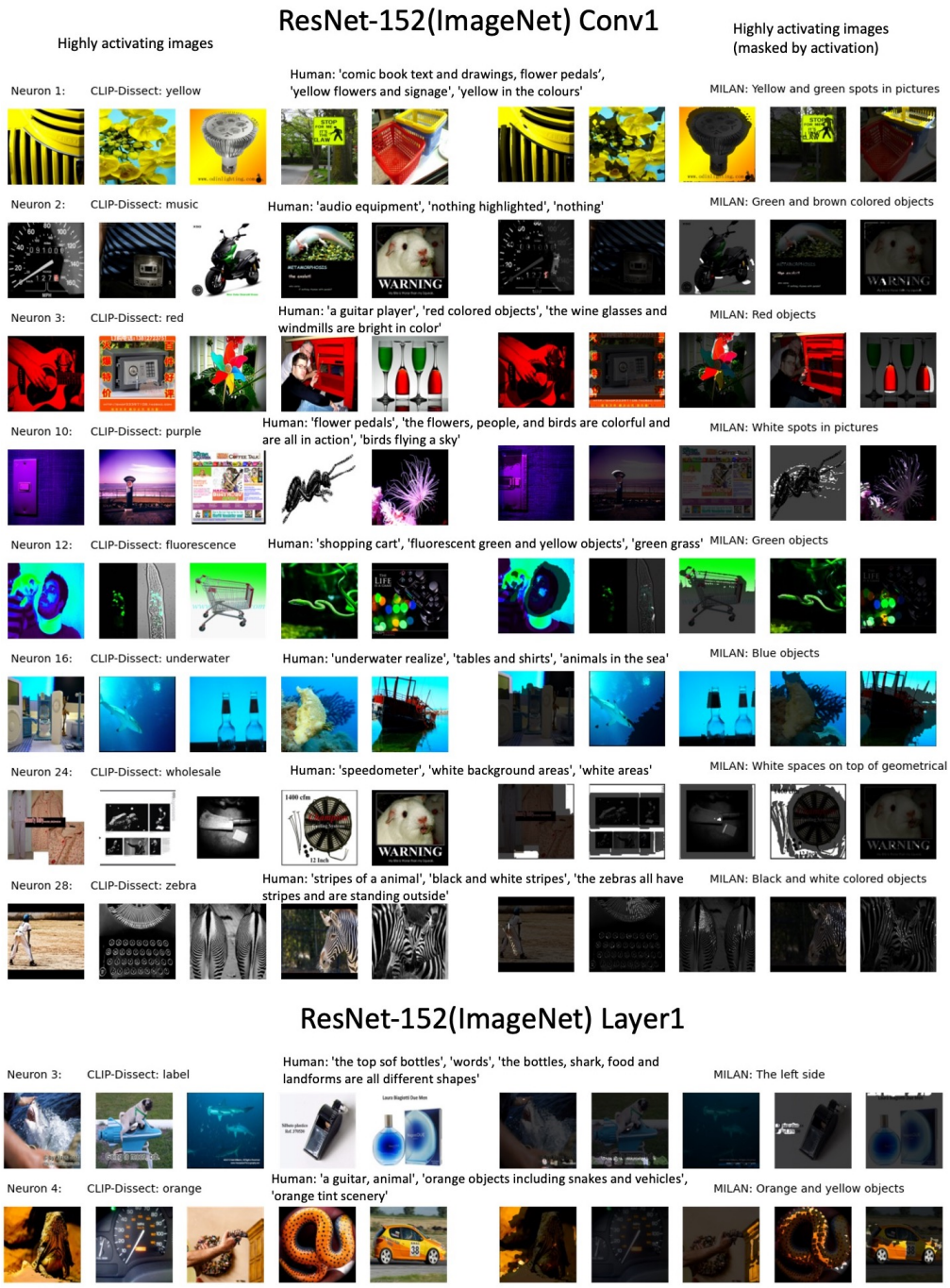



Figure 9: Descriptions for select neurons in early layers of ResNet-152, showcasing both the successes and failure modes of CLIP-Dissect. For this figure we used max summary function to be comparable to MILAnnotations. For evaluation we used the MILAN model trained on only Places models to avoid overfitting.

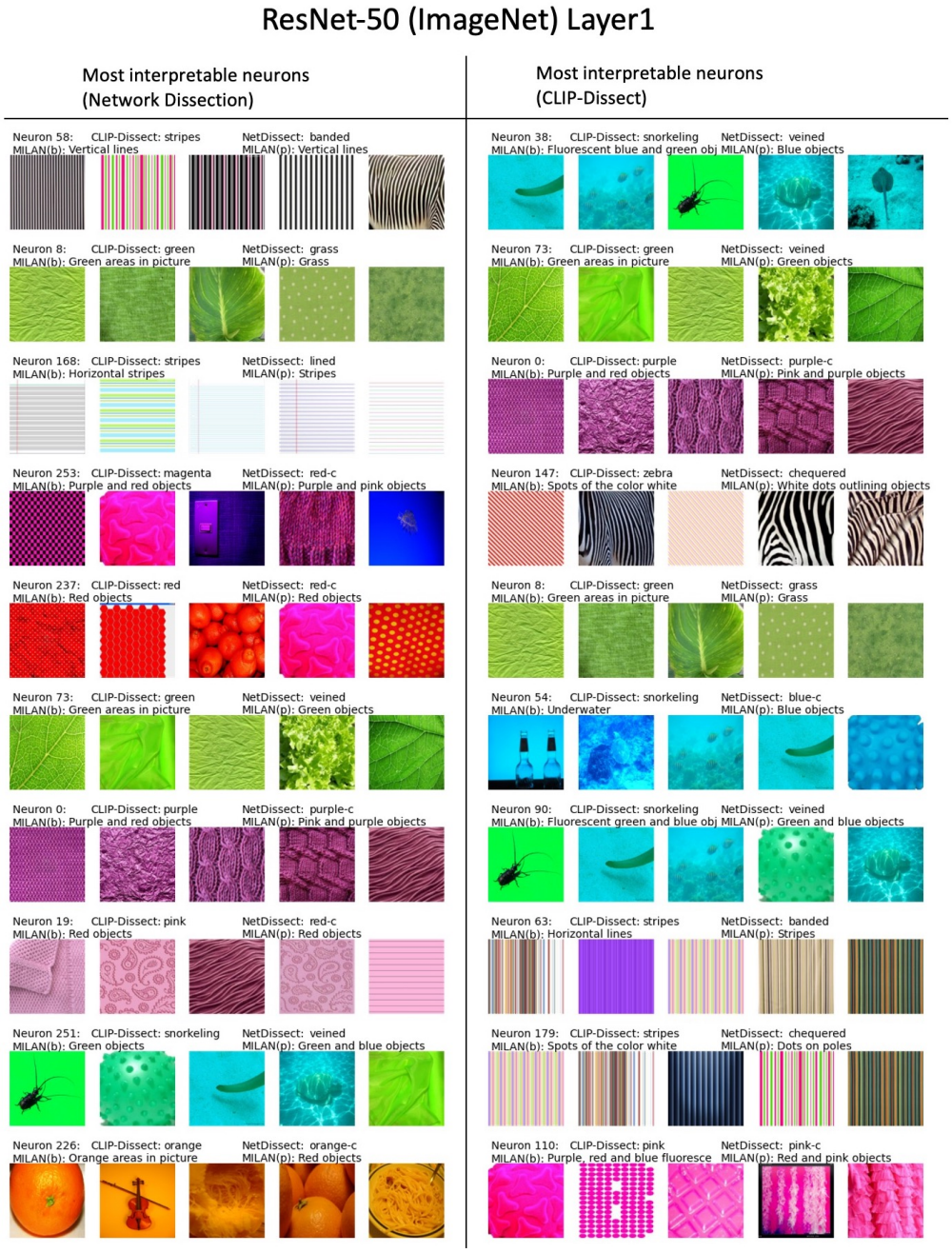



Figure 10: Descriptions of most interpretable neurons (highest similarity/IoU) of an early layer in ResNet-50.

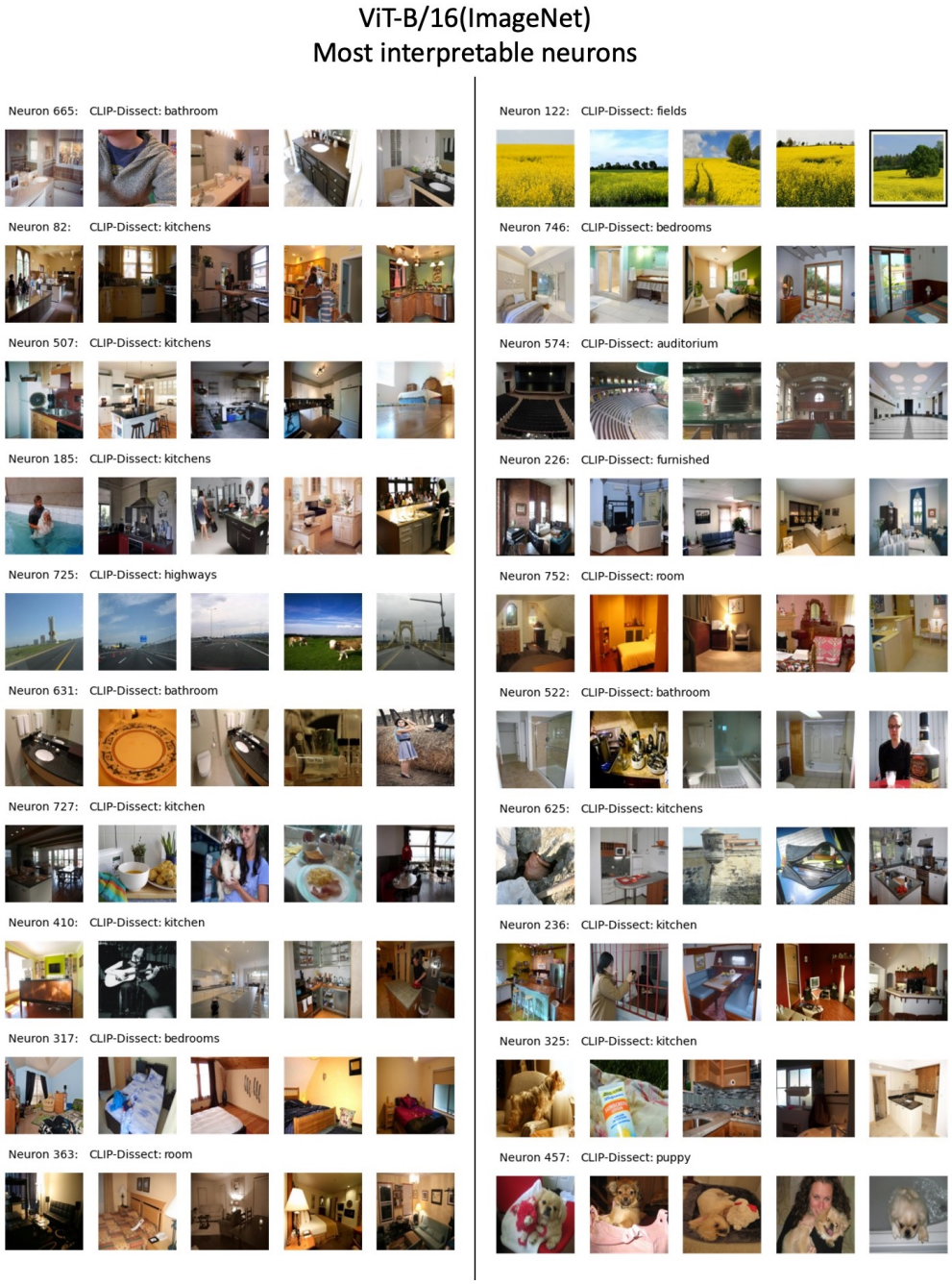



Figure 11: Most interpretable neurons after the 'encoder' module (one of the last layers) of a Vision Transformer.