

# Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks

Ruth Fong  
University of Oxford  
ruthfong@robots.ox.ac.uk

Andrea Vedaldi  
University of Oxford  
vedaldi@robots.ox.ac.uk

## Abstract

In an effort to understand the meaning of the intermediate representations captured by deep networks, recent papers have tried to associate specific semantic concepts to individual neural network filter responses, where interesting correlations are often found, largely by focusing on extremal filter responses. In this paper, we show that this approach can favor easy-to-interpret cases that are not necessarily representative of the average behavior of a representation.

A more realistic but harder-to-study hypothesis is that semantic representations are distributed, and thus filters must be studied in conjunction. In order to investigate this idea while enabling systematic visualization and quantification of multiple filter responses, we introduce the Net2Vec framework, in which semantic concepts are mapped to vectorial embeddings based on corresponding filter responses. By studying such embeddings, we are able to show that 1., in most cases, multiple filters are required to code for a concept, that 2., often filters are not concept specific and help encode multiple concepts, and that 3., compared to single filter activations, filter embeddings are able to better characterize the meaning of a representation and its relationship to other concepts.

## 1. Introduction

While deep neural networks keep setting new records in almost all problems in computer vision, our understanding of these black-box models remains very limited. Without developing such an understanding, it is difficult to characterize and work around the limitations of deep networks, and improvements may only come from intuition and trial-and-error.

For deep learning to mature, a much better theoretical and empirical understanding of deep networks is thus required. There are several questions that need answering, such as how a deep network is able to solve a problem such

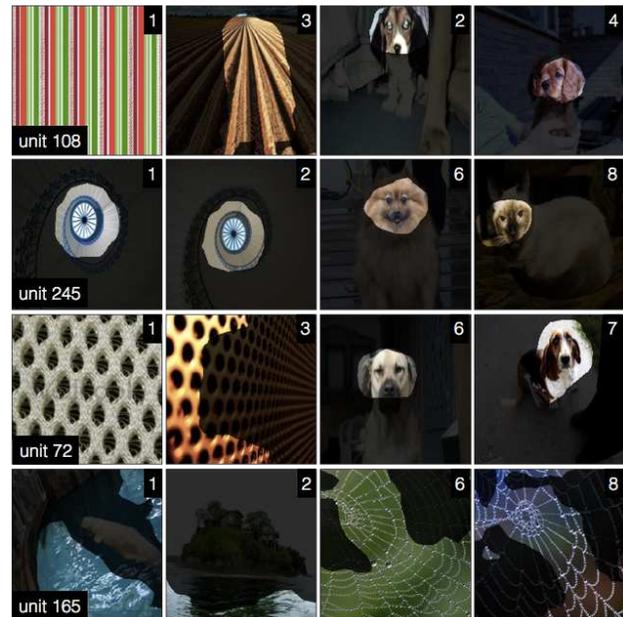


Figure 1. The diversity of BRODEN [4] images that most activate certain AlexNet conv5 filters motivates us to investigate to what extent a single filter encodes a concept fully, without needing other units, and exclusively, without encoding other concepts. An image’s corner number  $n$  denotes that it is the  $n$ -th most maximally activating image for the given filter. Masks were generated by our slightly modified NetDissect [4] approach (section 3.1.1) and are upsampled first before thresholding for smoothness.

as classifying an image, or how it can generalize so well despite having access to limited training data in relation to its own capacity [23]. In this paper, we ask in particular *what a convolutional neural network has learned to do* once training is complete. A neural network can be seen as a sequence of functions, each mapping an input image to some intermediate representation. While the final output of a network is usually easy to interpret (as it provides, hopefully, a solution to the task that the network was trained to solve), the meaning of the intermediate layers is far less clear. Understanding the information carried by these representations is

a first step to understanding how these networks work.

Several authors have researched the possibility that individual filters in a deep network are responsible for capturing particular semantic concepts. The idea is that low-level primitives such as edges and textures are recognized by earlier layers, and more complex objects and scenes by deeper ones. An excellent representative of this line of research is the recent Network Dissection approach by [4]. The authors of this paper introduce a new dataset, BRODEN, which contains pixel-level segmentation for hundreds of low- and high-level visual concepts, from textures to parts and objects. They then study the correlation between extremal filter responses and such concepts, seeking for filters that are strongly responsive for particular ones.

While this and similar studies [24, 22, 10] did find clear correlations between feature responses and various concepts, such an interpretation has intrinsic limitations. This can be seen from a simple counting argument: the number of available feature channels is usually far smaller than the number of different concepts that a neural network may need to encode to interpret a complex visual scene. This suggests that, at the very least, the representation must use combinations of filter responses to represent concepts or, in other words, be at least in part distributed.

**Overview.** The goal of this paper is to go beyond looking at individual filters, and to study instead what information is captured by **combinations** of neural network filters. In this paper, we conduct a thorough analysis to investigate how semantic concepts, such as objects and their parts, are encoded by CNN filters. In order to make this analysis manageable, we introduce the Net2Vec framework (section 3), which aligns semantic concepts with filter activations. It does so via learned concept embeddings that are used to weight filter activations to perform semantic tasks like segmentation and classification. Our concept vectors can be used to investigate both quantitatively *and* qualitatively the “overlap” of filters and concepts. Our novelty lies in outlining methods that go beyond simply demonstrating that multiple filters better encode concepts than single ones [2, 21] to quantifying and describing how a concept is encoded. Principally, we gain unique, interpretive power by formulating concept vectors as embeddings.

Using Net2Vec, we look first at two questions (section 4): (1) To what extent are individual filters sufficient to express a concept? Or, are multiple filters required to code for a single concept? (2) To what extent does a filter exclusively code for a single concept? Or, is a filter shared by many, diverse concepts? While answers to these questions depend on the specific filter or concept under consideration, we demonstrate how to **quantify** the “overlap” between filters and concepts and show that there are many cases in which both notions of exclusive overlap do not hold. That

is, if we were to interpret semantic concepts and filter activations as corresponding set of images, in the resulting Venn’s diagram the sets would intersect partially but neither kind of set would contain or be contained by the other.

While quantifying the relationship between concepts and representation may seem an obvious aim, so far much of the research on explaining how concepts are encoded by deep networks roughly falls into two more qualitative categories: (1) Interpretable visualizations of how single filters encode semantic concepts; (2) Demonstrations of distributive encoding with limited explanatory power of how a concept is encoded. In this work, we present methods that seek to marry the interpretive benefits of single filter visualizations with quantitative demonstrations of how concepts are encoded across multiple filters (section 5).

As part of our analysis, we also highlight the problem with visualizing only the inputs that maximally activate a filter and propose evaluating the power of explanatory visualizations by how well they can explain the whole distribution of filter activations (section 5.1).

## 2. Related Work

**Visualizations.** Several methods have been proposed to explain what a single filter encodes by visualizing a real [22] or generated [10, 17, 14] input that most activates a filter; these techniques are often used to argue that single filters substantially encode a concept. In contrast, [20] shows that visualizing the real image patches that most activate a layer’s filters after a random basis has been applied also yields semantically, coherent patches. [24, 4] visualize segmentation masks extracted from filter activations for the most confident or maximally activating images; they also evaluate their visualizations using human judgments.

**Distributed Encodings.** [2] demonstrates that most PASCAL classes require more than a few hidden units to perform classification well. Most similar to [24, 4], [6] concludes that only a few hidden units encode semantic concepts robustly by measuring the overlap between image patches that most activate a hidden unit with ground truth bounding boxes and collecting human judgments on whether such patches encode systematic concepts. [21] compares using individual filter activations with using clusters of activations from all units in a layer and shows that their clusters yielded better parts detectors and qualitatively correlated well with semantic concepts. [3] probes mid-layer filters by training linear classifiers on their activations and analyzing them at different layers and points of training.

## 3. Net2Vec

With our Net2Vec paradigm, we propose aligning concepts to filters in a CNN by (a) recording filter activations

of a pre-trained network when probed by inputs from a reference, “probe” dataset and (b) learning how to weight the collected probe activations to perform various semantic tasks. In this way, for every concept in the probe dataset, a concept weight is learned for the task of recognizing that concept. The resulting weights can then be interpreted as concept embeddings and analyzed to understand how concepts are encoded. For example, the performance on semantic tasks when using learned concept weights that span all filters in a layer can be compared to when using only a single filter or subset of filters.

In the remainder of the section, we provide details for how we learn concept embeddings by learning to segment (3.1) and classify (3.2) concepts. We also outline how we compare embeddings arising from using only a restricted set of filters, including single filters. Before we do so, we briefly discuss the dataset used to learn concepts.

**Data.** We build on the BRODEN dataset recently introduced by [4] and use it to primarily probe AlexNet [9] trained on the ImageNet dataset [16] as a representative model for image classification. BRODEN contains over 60,000 images with pixel- and image-level annotations for 1197 concepts across 6 categories: scenes (468), objects (584), parts (234), materials (32), textures (47), and colors (11). We exclude 8 scene concepts for which there were no validation examples. Thus, of the 1189 concepts we consider, all had image-level annotations, but only 682 had segmentation annotations, as only image-level annotations are provided for scene and texture concepts. Note that our paradigm can be generalized to any probe dataset that contains pixel- or image-level annotations for concepts. To compare the effects of different architectures and supervision, we also probe VGG16 [18] conv5\_3 and GoogLeNet [19] inception5b trained on ImageNet [16] and Places365 [25] as well as conv5 of the following self-supervised, AlexNet networks: tracking [21], audio [15], objectcentric [5], moving [1], and egomotion [7]. Post-ReLU activations are used.

### 3.1. Concept Segmentation

In this section, we show how learning to segment concepts can be used to induce concept embeddings using either all the filters available in a CNN layer or just a single filter. We also show how embeddings can be used to quantify the degree of overlap between filter combinations and concepts. This task is performed on all 682 Broden concepts with segmentation annotations, which excludes scene and texture concepts.

#### 3.1.1 Concept Segmentation by a Single Filter

We start by considering single filter segmentation following [4]’s paradigm with three minor modifications, listed below. For every filter  $k$ , let  $a_k$  be its corresponding activation (at a given pixel location and for a given input image). The  $\tau = 0.005$  activation’s quantile  $T_k$  is determined such that  $P(a_k > T_k) = \tau$ , and is computed with respect to the distribution  $p(a_k)$  of filter activations over all probe images and spatial locations; we use this cut-off point to match [4].

Filter  $k$  in layer  $l$  is used to generate a segmentation of an image by first thresholding  $A_k(\mathbf{x}) > T_k$ , where  $A_k(\mathbf{x}) \in \mathbb{R}^{H_l \times W_l}$  is the activation map of filter  $k$  on input  $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$  and upsampling the result as needed to match the resolution of the ground truth segmentation mask  $L_c(\mathbf{x})$ , i.e.  $M_k(\mathbf{x}) = S(A_k(\mathbf{x}) > T_k)$ , where  $S$  denotes a bilinear upsampling function.

Images may contain any number of different concepts, indexed by  $c$ . We use the symbol  $\mathbf{x} \in X_c$  to denote the probe images that contain concept  $c$ . To determine which filter  $k$  best segments concept  $c$ , we compute a set IoU score. This score is given by the formula

$$\text{IoU}_{\text{set}}(c; M_k, s) = \frac{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|} \quad (1)$$

which computes the intersection over union (Jakkard index) difference between the binary segmentation masks  $M_k$  produced by the filter and the ground-truth segmentation masks  $L_c$ . Note that sets are merged for all images in the subset  $X_{s,c}$  of the data, where  $s \in \{\text{train, val}\}$ . The best filter  $k^*(c) = \text{argmax}_k \text{IoU}_{\text{set}}(c; M_k, \text{train})$  is then selected on the training set and the validation score  $\text{IoU}_{\text{set}}(c; M_{k^*}, \text{val})$  is reported.

We differ from [4] in the following ways: (1) we threshold before upsampling, in order to more evenly compare to the method described below; (2) we bilinearly upsample without anchoring interpolants at the center of filter receptive fields to speed up the upsampling part of the experimental pipeline; and (3) we determine the best filter for a concept on the training split  $X_{\text{train},c}$  rather than  $X_c$  whereas [4] does not distinguish a training and validation set.

#### 3.1.2 Concept Segmentation by Filter Combinations

In order to compare single-feature concept embeddings to representations that use filter combinations, we also learn to solve the segmentation task using *combinations* of filters extracted by the neural network. For this, we learn weights  $\mathbf{w} \in \mathbb{R}^K$ , where  $K$  is the number of filters in a layer, to linearly combine thresholded activations. Then, the linear combination is passed through the sigmoid function  $\sigma(z) =$

$1/(1 + \exp(-z))$  to predict a segmentation mask  $M(\mathbf{x}; \mathbf{w})$ :

$$M(\mathbf{x}; \mathbf{w}) = \sigma \left( \sum_k w_k \cdot \mathbb{I}(A_k(\mathbf{x}) > T_k) \right) \quad (2)$$

where  $\mathbb{I}(\cdot)$  is the indicator function of an event. The sigmoid is irrelevant for evaluation, for which we threshold the mask predicted by  $M(\mathbf{x}; \mathbf{w})$  by  $\frac{1}{2}$ , but has an effect in training the weights  $\mathbf{w}$ .

Similar to the single filter case, for each concept the weights  $\mathbf{w}$  are learned on  $X_{\text{train},c}$  and the set IoU score computed on thresholded masks for  $X_{\text{val},c}$  is reported. In addition to evaluating on the set IoU score, per-image IoU scores are computed as well:

$$\text{IoU}_{\text{ind}}(\mathbf{x}, c; M) = \frac{|M(\mathbf{x}) \cap L_c(\mathbf{x})|}{|M(\mathbf{x}) \cup L_c(\mathbf{x})|} \quad (3)$$

Note that choosing a single filter is analogous to setting  $\mathbf{w}$  to a one-hot vector, where  $w_k = 1$  for the selected filter and  $w_k = 0$  otherwise, recovering the single-filter segmenter of section 3.1.1, with the output rescaled by the sigmoid function (2).

**Training** For each concept  $c$ , the segmentation concept weights  $\mathbf{w}$  are learned using SGD with momentum (lr =  $10^{-4}$ , momentum  $\gamma = 0.9$ , batch size 64, 30 epochs) to minimize a per-pixel binary cross entropy loss weighted by the mean concept size, i.e.  $1-\alpha$ :

$$\mathcal{L}_1 = -\frac{1}{N_{s,c}} \sum_{\mathbf{x} \in X_{s,c}} \alpha M(\mathbf{x}; \mathbf{w}) L_c(\mathbf{x}) + (1-\alpha)(1 - M(\mathbf{x}; \mathbf{w})(1 - L_c(\mathbf{x}))), \quad (4)$$

where  $N_{s,c} = |X_{s,c}|$ ,  $s \in \{\text{train}, \text{val}\}$ , and  $\alpha = 1 - \frac{\sum_{\mathbf{x} \in X_{\text{train}}} |L_c(\mathbf{x})| / S}{|L_c(\mathbf{x})|}$ , where  $|L_c(\mathbf{x})|$  is the number of foreground pixels for concept  $c$  in the ground truth (g.t.) mask for  $\mathbf{x}$  and  $S = h_s \cdot w_s$  is the number of pixels in g.t. masks.

## 3.2. Concept Classification

As an alternate task to concept segmentation, the problem of classifying concept (i.e., to tell whether the concept occurs somewhere in the image) can be used to induce concept embeddings. In this case, we discuss first learning embeddings using generic filter combinations (3.2.1) and then reducing those to only use a small subset of filters (3.2.2).

### 3.2.1 Concept Classification by Filter Combinations

Similar to our segmentation paradigm, for each concept  $c$ , a weight vector  $\mathbf{w} \in \mathbb{R}^K$  and a bias term  $b \in \mathbb{R}$  are learned to combine the spatially-averaged filter activations  $k$ ; the linear combination is then passed through the sigmoid function

$\sigma$  to obtain the concept posterior probability:

$$f(\mathbf{x}; \mathbf{w}, b) = \sigma \left( b + \sum_k w_k \cdot \frac{\sum_{i=1}^{H_l} \sum_{j=1}^{W_l} A_{ijk}(\mathbf{x})}{H_l W_l} \right) \quad (5)$$

where  $H_l$  and  $W_l$  denote the height and width respectively of layer  $l$ 's activation map  $A_k(\mathbf{x})$ .

For each concept  $c$ , the training images  $X_{\text{train}}$  are divided into the positive subset  $X_{\text{train},c+}$  of images that contain concept  $c$  and its complement  $X_{\text{train},c-}$  of images that do not. While in general the positive and negative sets are unbalanced, during training, images from the two sets are sampled with equal probability in order to re-balance the data (supp. sec. 1.2). To evaluate performance, we calculate the classification accuracy over a balanced validation set.

### 3.2.2 Concept Classification by a Subset of Filters

In order to compare using all filters in a layer to just a subset of filters, or even individual filters, we must learn corresponding concept classifiers. Following [2], for each concept  $c$ , after learning weights  $\mathbf{w}$  as explained before, we choose the top  $F$  by their absolute weight  $|w_k|$ . Then, we learn new weights  $\mathbf{w}' \in \mathbb{R}^F$  and bias  $b'$  that are used to weight activations from only these  $F$  filters. With respect to eq. (5), this is analogous to learning new weights  $\mathbf{w}' \in \mathbb{R}^K$ , where  $w'_k = 0$  for all filters  $k$  that are not the top  $F$  ones. We train such classifiers for  $F \in \{1, 2, 3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 80, 100, 128\}$  for the last three AlexNet layers and for all its layers for the special case  $F = 1$ , corresponding to a single filter. For comparison, we use this same method to select subsets of filters for the segmentation task on the last layer using  $F \in \{1, 2, 4, 8, 16, 32, 64, 128, 160, 192, 224\}$ .

## 4. Quantifying the Filter-Concept Overlap

### 4.1. Are Filters Sufficient Statistics for Concepts?

We start by investigating a popular hypothesis: whether concepts are well represented by the activation of individual filters or not. In order to quantify this, we consider how our learned weights, which combine information from all filter activations in a layer, compare to a single filter when being used to perform segmentation and classification on BRODEN.

Figure 2 shows that, on average, using learned weights to combine filters outperforms using a single filter on both the segmentation and classification tasks (sections 3.1.1 and 3.2.2) when being evaluated on validation data. The improvements can be quite dramatic for some concepts and starts in conv1. For instance, even for simple concepts like colors, filter combinations outperform individual filters by up to  $4\times$  (see supp. figs. 2-4 for graphs on the performance of individual concepts). This suggests that, even if

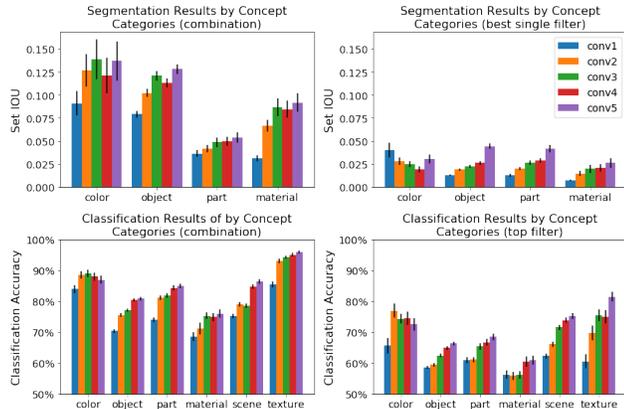


Figure 2. Results by concept category on the segmentation (top) and classification (bottom) tasks show that, on average, using learned weights to combine filters (left) out performs using a single filter (right). Standard error is shown.

filters specific to a concept can be found, these do not optimally encode or fully “overlap” with the concept. In line with the accepted notion that deep layers improve representational quality, task performance generally improves as the layer depth increases, with trends for the color concepts being the notable exception. Furthermore, the average performance varies significantly by concept category and consistently in both the single- and multi-filter classification plots (bottom). This suggests that certain concepts are less well-aligned via linear combination to the filter space.

### How many filters are required to encode a concept?

To answer this question, we observe how varying the number of top conv5 filters,  $F$ , from which we learn concept weights affects performance (section 3.2.2). Figure 3 shows that mean performance saturates at different  $F$  for the various concept categories and tasks. For the classification task (right), most concept categories saturate by  $F = 50$ ; however, scenes reaches near optimal performance around  $F = 15$ , which is much more quickly than that of materials. For the segmentation task (left), performance peaks much earlier at  $F = 8$  for materials and parts,  $F = 16$  for objects, and  $F = 128$  for colors. We also observe performance drops after reaching optimal peaks for materials and parts in the segmentation class. This highlights that the segmentation task is challenging for those concept categories in particular (i.e., object parts are much smaller and harder to segment, materials are most different from network’s original ImageNet training examples of objects); with more filters to optimize for, learning is more unstable and more likely to reach a sub-optimal solution.

**Failure Cases.** While on average our multi-filter approach significantly outperforms a single-filter approach

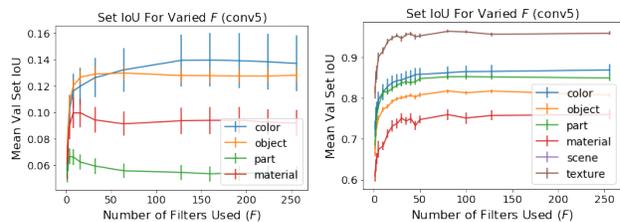


Figure 3. Results by concept category and number of top conv5 filters used for segmentation and classification show that different categories and tasks saturate in performance at different  $F$ .

Table 1. Percent of concepts for which the evaluation metric (set IoU for segmentation and accuracy for classification) is equal to or better when using learned weights than the best single filter.

	conv1	conv2	conv3	conv4	conv5
Segmentation	91.6%	86.8%	84.0%	82.3%	75.7%
Classification	87.8%	90.2%	85.0%	87.9%	88.1%

on both segmentation and classification tasks (fig. 2), Table 1 shows that for around 10% of concepts, this does not hold. For segmentation, this percentage increases with layer depth. Upon investigation, we discovered that the concepts for which our learned weights do not outperform the best filter either have very few examples for that concept, i.e. mostly  $|X_{\text{train},c}| \in [10, 100]$  which leads to overfitting; or are very small objects, of average size less than 1% of an image, and thus training with the size weighted (4) loss is unstable and difficult, particularly at later layers where there is low spatial resolution. A similar analysis on the classification results shows that small concept dataset size is also causing overfitting in failure cases: Of the 133 conv5 failure cases, 103 had at most 20 positive training examples and all but one had less than 100 positive training examples (supplementary material figs. 7 and 8).

### 4.2. Are Filters Shared between Concepts?

Next, we investigate the extent to which a single filter is used to encode many concepts. Note that Figure 1 suggests that a single filter might be activated by different concepts; often, the different concepts a filter appears to be activated by are related by a latent concept that may or may not be human-interpretable, i.e., an ‘animal torso’ filter which also is involved in characterizing animals like ‘sheep’, ‘cow’, and ‘horse’ (fig. 4, supp. fig. 9).

Using the single best filters identified in both the segmentation and classification tasks, we explore how often a filter is selected as the best filter to encode a concept. Figure 5 shows the distribution of how many filters (y-axis) encode how many concepts (x-axis). Interestingly, around 15% of conv1 filters (as well as several in all the other layers) were selected for encoding at least 20 and 30 concepts ( $\#$  of concepts /  $\#$  of conv1 filters = 10.7 and 18.6; supp.



Figure 4. AlexNet conv5 filter 66 appears selective for pastoral animal’s torso. Validation examples for ‘sheep’, ‘horse’, and ‘cow’ with the highest individual IOU scores are given (masks are up-sampled before thresholding for visual smoothness).

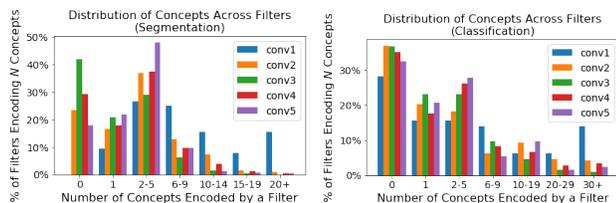


Figure 5. For each filter in a layer, the number of concepts for which it is selected as the best filter in the segmentation (left) and classification (right) tasks is counted and binned.

tbl. 1) for the segmentation and classification tasks respectively and a substantial portion of filters in each layer (except conv1 for the segmentation task) are never selected. The filters selected to encode numerous concepts are not exclusively “overlapped” by a single concept. The filters that were not selected to encode any concepts are likely not be involved in detecting highly discriminative features.

### 4.3. More Architectures, Datasets, and Tasks

Figure 6 shows segmentation (top) and classification (bottom) results when using AlexNet (AN) conv5, VGG16 (VGG) conv5\_3, and GoogLeNet (GN) inception5b trained on both ImageNet (IN) and Places365 (P) as well as conv5 of these self-supervised (SS), AlexNet networks: tracking, audio, objectcentric, moving, and egomotion. GN performed worse than VGG because of its lower spatial resolution ( $7 \times 7$  vs.  $14 \times 14$ ); GN-IN inception4e ( $14 \times 14$ ) outperforms VGG-IN conv5\_3 (supp. fig. 11). In [4], GN detects scenes well, which we exclude due to lack of segmentation data. SS performance improves more than supervised networks (5-6x vs. 2-4x), suggesting that SS networks encode

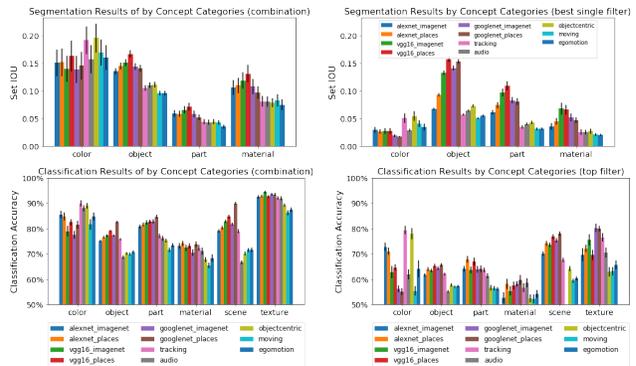


Figure 6. Segmentation (top) and classification (bottom) results for additional networks & datasets.

BRODEN concepts more distributedly.

## 5. Interpretability

In this section, we propose a new standard for visualizing non-extreme examples, show how the single- and multi-filter perspectives can be unified, and demonstrate how viewing concept weights as embeddings in filter space give us novel explanatory power.

### 5.1. Visualizing Non-Maximal Examples

Many visual explanation methods demonstrate their value by showing visualizations of inputs that maximally activate a filter, whether that be real, maximally-activating image patches [22]; learned, generated maximally-activated inputs [11, 14]; or filter segmentation masks for maximally-activating images from a probe dataset [4].

While useful, these approaches fail to consider how visualizations differ across the distribution of examples. Figure 7 shows that using a single filter to segment concepts [4] yields  $\text{IoU}_{\text{ind}}$  scores of 0 for many examples; such examples are simply not considered by the set IoU metric. This often occurs because no activations survive the  $\tau$ -thresholding step, which suggests that a single filter does not consistently fire strongly on a given concept.

We argue that a visualization technique should still work on and be informative for non-maximal examples. In Figure 8, we automatically select and visualize examples at each decile of the non-zero portion of the individual  $\text{IoU}$  distribution (fig. 7) using both learned concept weights and the best filters identified for each of the visualized categories. For ‘dog’ and ‘airplane’ visualizations using our weighted combination method, the predicted masks are informative and salient for most of the examples, even the lowest 10th percentile (leftmost column). Ideally, using this decile sampling method, the visualizations should appear salient even for examples from lower deciles. However, for examples using the best single filter (odd rows), the visualizations are not interpretable until higher deciles (rightmost

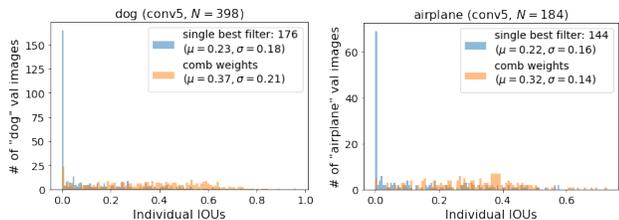


Figure 7. The empirical  $\text{IoU}_{\text{ind}}$  distribution when using the best single filter and the learned weights for ‘dog’ (left) and ‘train’ (right) ( $\mu, \sigma$  computed on the non-zero part of each distribution).

columns). This is in contrast to the visually appealing, maximally activating examples shown in supp. fig. 13.

## 5.2. Unifying Single- & Multi-Filter Views

Figure 9 highlights that single filter performance is often strongly, linearly correlated with the learned weights  $\mathbf{w}$ , thereby showing that individual filter performance is indicative of how weighted it’d be in a linear filter combination. Visually, a filter’s set IoU score appears correlated with its associated weight value passed through a ReLU, i.e.,  $\max(w_k, 0)$ . For each of the 682 BRODEN segmentation concepts and each AlexNet layer, we computed the correlation between  $\max(\mathbf{w}, 0)$  and  $\{\text{IoU}_{\text{set}}(c; M_k, \text{val})\}_{k=1\dots K}$ . By conv3, around 80% of segmentation concepts are significantly correlated ( $p < 0.01$ ): conv1: 47.33%, conv2: 69.12%, conv3: 81.14%, conv4: 79.13%, conv5: 82.47%. Thus, we show how the single filter perspective can be unified with and utilized to explain the distributive perspective: we can quantify how much a single filter  $k$  contributes to concept  $c$ ’s encoding from either  $\frac{|w_k|}{\|\mathbf{w}\|_1}$  where  $\mathbf{w}$  is  $c$ ’s learned weight vector or  $\frac{\text{IoU}_{\text{set}}(c; M_{k^*}, \text{val})}{\text{IoU}_{\text{set}}(c; M(\cdot; \mathbf{w}), \text{val})}$ .

## 5.3. Explanatory Power via Concept Embeddings

Finally, the learned weights can be considered as embeddings, where each dimension corresponds to a filter. Then, we can leverage the rich literature [12, 13, 8] on word embeddings derived from textual data to better understand which concepts are similar to each other in network space. To our knowledge, this is the first work that learns semantic embeddings aligned to the filter space of a network from visual data alone. (For this section, concept weights are normalized to be unit length, i.e.,  $\mathbf{w}' = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ ).

Table 2 shows the five closest concepts in cosine distance, where 1 denotes that  $\mathbf{w}'_1$  is  $0^\circ$  from  $\mathbf{w}'_2$  and  $-1$  denotes that  $\mathbf{w}'_1$  is  $180^\circ$  from  $\mathbf{w}'_2$ . These examples suggest that the embeddings from the segmentation and classification tasks capture slightly different relationships between concepts. Specifically, the nearby concepts in segmentation space appear to be similar-category objects (i.e., animals in the case of ‘cat’ and ‘horse’ being nearest to ‘dog’),

whereas the nearby concepts in classification space appear to be concepts that are related compositionally (i.e., parts of an object in the case of ‘muzzle’ and ‘paw’ being nearest to ‘dog’). Note that ‘street’ and ‘bedroom’ are categorized as scenes and thus lack segmentation annotations.

**Understanding the Embedding Space.** Table 3 shows that we can also do vector arithmetic by adding and subtracting concept embeddings to get meaningful results. For instance, we observe an analogy relationship between ‘grass’–‘green’ and ‘sky’–‘blue’ and other coherent results, such as non-green, ‘ground’-like concepts for ‘grass’ minus ‘green’ and floral concepts for ‘tree’ minus ‘wood’. t-SNE visualizations and K-means clustering (see supp. table 2 and supp. figs. 16 and 17) also demonstrate that networks learn meaningful, semantic relationships between concepts.

**Comparing Embeddings from Different Learned Representations.** The learned embeddings extracted from individual networks can be compared with one another quantitatively (as well as to other semantic representations). Let  $d(W) : \mathbb{R}^{C \times K} \rightarrow \mathbb{R}^{C \times C} = W \cdot W^T$  compute the cosine distance matrix for  $C$  concepts of a given representation (e.g., AlexNet), whose normalized embeddings  $\mathbf{w}'$  form the rows of  $W$ . Then,  $D_{i,j} = \|d(W^i) - d(W^j)\|_2^2$  quantifies the distance between two embedding spaces  $W^i, W^j$ , and  $D_{i,j,c} = \|d(W^i)_c - d(W^j)_c\|_2^2$  does that for concept  $c$ . Figure 10 (left) shows  $D_{i,j}$  between 24 embedding spaces: 2 tasks  $\times$  11 network, WordNet (WN), and Word2Vec (W2V) ( $C = 501$ , the number of BRODEN concepts available for all embeddings; see supp. sec. 3.2.1). It shows that tracking and audio (T, A) classification embeddings are quite different from others, and that classification embeddings (-C) are more aligned to WN and W2V than segmentation ones (-S). Figure 10 (right) shows select mean  $D_{i,j,c}$  distances averaged over concept categories. It demonstrates that colors are quite similar between WN and network embeddings and that materials most differ between audio and the WN and W2V embeddings.

## 6. Conclusion

We present a paradigm for learning concept embeddings that are aligned to a CNN layer’s filter space. Not only do we answer the binary questions, “does a single filter encode a concept fully and exclusively?,” we also introduce the idea of filter and concept “overlap” and outline methods for answering the scalar extension questions, “to what extent...?” We also propose a more fair standard for visualizing non-extreme examples and show how to explain distributed concept encodings via embeddings. While powerful and interpretable, our approach is limited by its linear nature; future work should explore non-linear ways concepts can be better

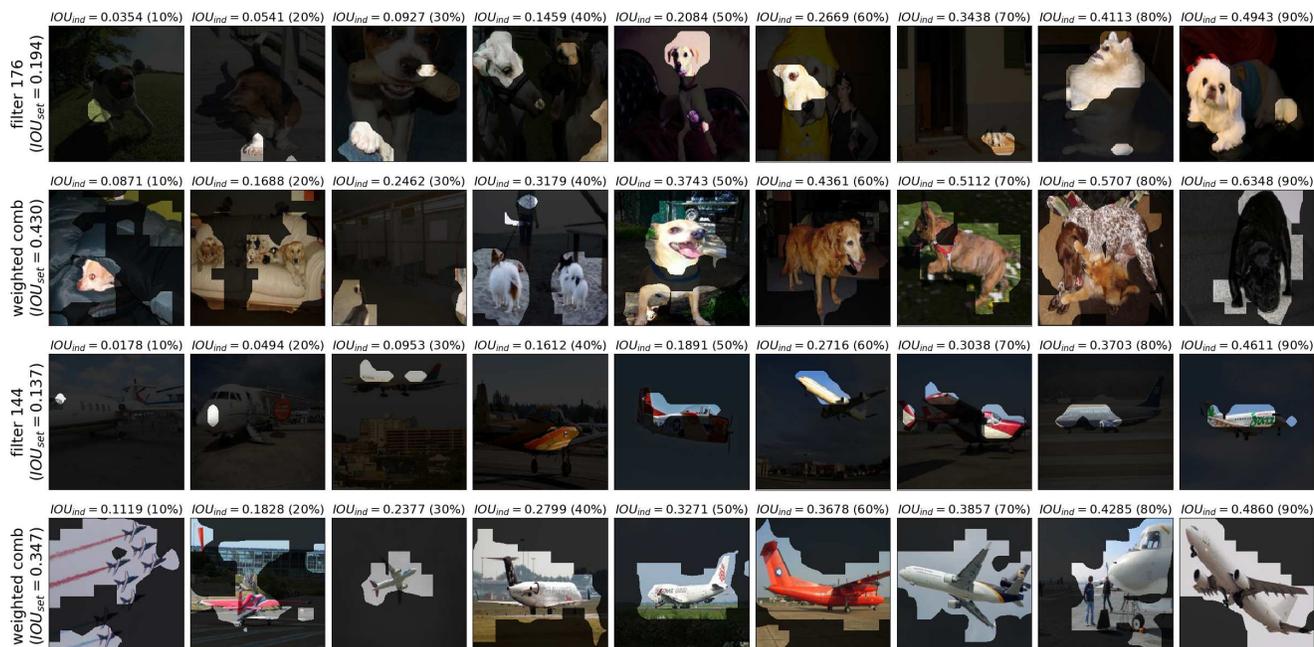


Figure 8. For the ‘dog’ and ‘airplane’ concepts, an example is automatically selected at each decile of the non-zero portion of the distribution of individual IoU scores (Figure 7), and the predicted conv5 segmentation masks using the best filter (odd rows) as well as the learned weights (even rows) are overlaid.

Table 2. Nearest concepts (in cos distance) using segmentation (left sub-columns) and classification (right) conv5 embeddings.

dog		house		wheel		street		bedroom	
cat (0.81)	muzzle (0.73)	building (0.77)	path (0.56)	bicycle (0.86)	headlight (0.66)	n/a	sidewalk (0.74)	n/a	headboard (0.90)
horse (0.73)	paw (0.65)	henhouse (0.62)	dacha (0.54)	motorbike (0.66)	car (0.53)	n/a	streetlight (0.73)	n/a	bed (0.85)
muzzle (0.73)	tail (0.52)	balcony (0.56)	hovel (0.54)	carriage (0.54)	bicycle (0.52)	n/a	license plate (0.73)	n/a	pillow (0.84)
ear (0.72)	nose (0.47)	bandstand (0.54)	chimney (0.53)	wheelchair (0.53)	road (0.51)	n/a	traffic light (0.73)	n/a	footboard (0.82)
tail (0.72)	torso (0.44)	watchtower (0.52)	earth (0.52)	water wheel (0.48)	license plate (0.49)	n/a	windshield (0.71)	n/a	shade (0.74)

Table 3. Vector arithmetic using segmentation, conv5 weights.

grass + blue - green	grass - green	tree - wood	person - torso
sky (0.17)	earth (0.22)	plant (0.36)	foot (0.12)
patio (0.10)	path (0.21)	flower (0.29)	hand (0.10)
greenhouse (0.10)	brown (0.18)	brush (0.29)	grass (0.09)
purple (0.09)	sand (0.16)	bush (0.28)	mountn. pass (0.09)
water (0.09)	patio (0.15)	green (0.25)	backpack (0.09)

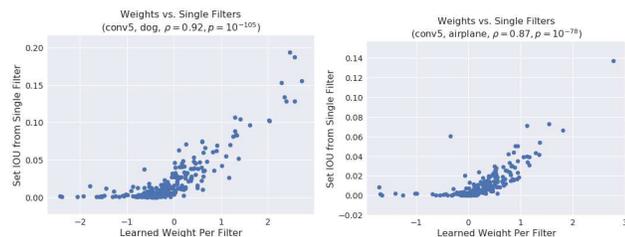


Figure 9. Correlation between learned segmentation weights and each filter’s set IoU score for ‘dog’ (left) and ‘airplane’ (right).

aligned to the filter space.

**Acknowledgements.** We gratefully acknowledge the support of the Rhodes Trust for Ruth Fong and ERC 677195-IDIU for Andrea Vedaldi.

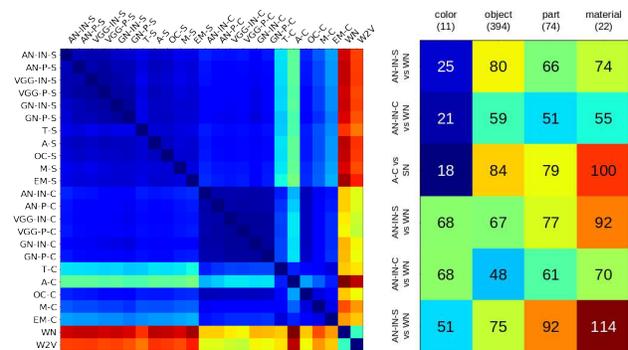


Figure 10. Comparing Net2Vec embeddings quantitatively. Left: Each cell corresponds to distance  $D_{i,j}$  for embedding spaces  $i$  and  $j$  (see section 4.3 for abbreviations). Right: Each cell corresponds to mean distance  $D_{i,j,c}$  for each concept category.

## References

- [1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.

- [2] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.
- [3] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.
- [4] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017.
- [5] R. Gao, D. Jayaraman, and K. Grauman. Object-centric representation learning from unlabeled videos. In *ACCV*, 2016.
- [6] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *IJCV*, 2016.
- [7] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015.
- [8] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. In *TACL*, 2014.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [10] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.
- [11] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.
- [12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
- [13] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, 2013.
- [14] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, 2016.
- [15] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.
- [16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [17] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR workshop*, 2014.
- [18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.
- [21] J. Wang, Z. Zhang, C. Xie, V. Premachandran, and A. Yuille. Unsupervised learning of object semantic parts from internal states of cnns by population encoding. *arXiv preprint arXiv:1511.06855*, 2015.
- [22] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, 2013.
- [23] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, 2016.
- [24] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.
- [25] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *T-PAMI*, 2016.