Interpretable Convolutional Neural Networks

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu
University of California, Los Angeles

Abstract

This paper proposes a method to modify a traditional convolutional neural network (CNN) into an interpretable CNN, in order to clarify knowledge representations in high conv-layers of the CNN. In an interpretable CNN, each filter in a high conv-layer represents a specific object part. Our interpretable CNNs use the same training data as ordinary CNNs without a need for any annotations of object parts or textures for supervision. The interpretable CNN automatically assigns each filter in a high conv-layer with an object part during the learning process. We can apply our method to different types of CNNs with various structures. The explicit knowledge representation in an interpretable CNN can help people understand the logic inside a CNN, i.e. *what patterns are memorized by the CNN for prediction. Experiments have shown that filters in an interpretable CNN are more semantically meaningful than those in a traditional CNN. The code is available at* `https://github.com/zqs1022/interpretableCNN`.

1. Introduction

Convolutional neural networks (CNNs) [14, 12, 7] have achieved superior performance in many visual tasks, such as object classification and detection. Besides the discrimination power, model interpretability is another crucial property for neural networks. However, the interpretability is always an Achilles' heel of CNNs, and has presented challenges for decades.

In this paper, we focus on a new problem, *i.e. without any additional human supervision, can we modify a CNN to make its conv-layers obtain interpretable knowledge representations?* We expect the CNN to have a certain introspection of its representations during the end-to-end learning process, so that the CNN can regularize its representations to ensure high interpretability. Our problem of improving CNN interpretability is different from conventional off-line visualization [34, 17, 24, 4, 5, 21] and diagnosis [2, 10, 18] of pre-trained CNN representations.

Bau *et al.* [2] defined six kinds of semantics in CNNs, *i.e. objects*, *parts*, *scenes*, *textures*, *materials*, and *colors*.

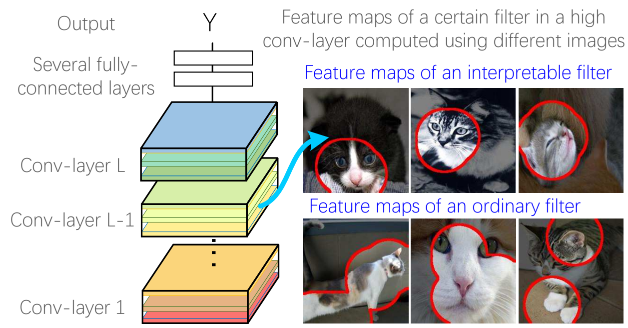

Figure 1. Comparison of a filter's feature maps in an interpretable CNN and those in a traditional CNN.

In fact, we can roughly consider the first two semantics as object-part patterns with specific shapes, and summarize the last four semantics as texture patterns without clear contours. Moreover, filters in low conv-layers usually describe simple textures, whereas filters in high conv-layers are more likely to represent object parts.

Therefore, in this study, we aim to train each filter in a high conv-layer to represent an object part. Fig. 1 shows the difference between a traditional CNN and our interpretable CNN. In a traditional CNN, a high-layer filter may describe a mixture of patterns, *i.e.* the filter may be activated by both the head part and the leg part of a cat. Such complex representations in high conv-layers significantly decrease the network interpretability. In contrast, the filter in our interpretable CNN is activated by a certain part. In this way, we can explicitly identify which object parts are memorized by CNN filters for classification without ambiguity. The goal of this study can be summarized as follows.

- We propose to slightly revise a CNN to improve its interpretability, which can be broadly applied to CNNs with different structures.
- We do **not** need any annotations of object parts or textures for supervision. Instead, our method automatically pushes the representation of each filter towards an object part.
- The interpretable CNN does not change the loss function on the top layer and uses the same training samples as the original CNN.

- As an exploratory research, the design for interpretability may decrease the discrimination power a bit, but we hope to limit such a decrease within a small range.

Method: We propose a simple yet effective loss to push a filter in a specific conv-layer of a CNN towards the representation of an object part. As shown in Fig. 2, we add the loss for the output feature map of each filter. The loss encourages a low entropy of inter-category activations and a low entropy of spatial distributions of neural activations. *I.e.* 1) each filter must encode a distinct object part that is exclusively contained by a single object category, and 2) the filter must be activated by a single part of the object, rather than repetitively appear on different object regions. We assume that repetitive shapes on various regions are more likely to describe low-level textures (*e.g.* colors and edges), instead of high-level parts. For example, the left eye and the right eye may be represented by different filters, because contexts of the two eyes are symmetric, but not the same.

The value of network interpretability: The clear semantics in high conv-layers is of great importance when we need people to trust a network's prediction. As discussed in [38], considering dataset bias and representation bias, a high accuracy on testing images still cannot ensure a CNN to encode correct representations (*e.g.* a CNN may use an unreliable context—eye features—to identify the "lipstick" attribute of a face image). Therefore, human beings usually cannot fully trust a network, unless a CNN can semantically or visually explain its logic, *e.g.* what patterns are learned for prediction. Given an image, current studies for network diagnosis [5, 21, 18] localize image regions that contribute most to prediction outputs at the pixel level. In this study, we expect the CNN to explain its logic at the object-part level. Given an interpretable CNN, we can explicitly show the distribution of object parts that are memorized by the CNN for object classification.

Contributions: In this paper, we focus on a new task, *i.e.* end-to-end learning a CNN whose representations in high conv-layers are interpretable. We propose a simple yet effective method to modify different types of CNNs into interpretable CNNs without any annotations of object parts or textures for supervision. Experiments show that our approach has significantly improved the object-part interpretability of CNNs.

2. Related work

Our previous paper [39] provides a comprehensive survey of recent studies in exploring visual interpretability of neural networks, including 1) the visualization and diagnosis of CNN representations, 2) approaches for disentangling CNN representations into graphs or trees, 3) the learning of CNNs with disentangled and interpretable representations, and 4) middle-to-end learning based on model interpretability.

Network visualization: Visualization of filters in a CNN is the most direct way of exploring the pattern hidden inside a neural unit. [34, 17, 24] showed the appearance that maximized the score of a given unit. up-convolutional nets [4] were used to invert CNN feature maps to images.

Pattern retrieval: Some studies go beyond passive visualization and actively retrieve certain units from CNNs for different applications. Like the extraction of mid-level features [26] from images, pattern retrieval mainly learns mid-level representations from conv-layers. Zhou *et al.* [40, 41] selected units from feature maps to describe "scenes". Simon *et al.* discovered objects from feature maps of unlabeled images [22], and selected a certain filter to describe each semantic part in a supervised fashion [23]. Zhang *et al.* [36] extracted certain neural units from a filter's feature map to describe an object part in a weakly-supervised manner. They also disentangled CNN representations in an And-Or graph via active question-answering [37]. [6] used a gradient-based method to interpret visual question-answering. [11, 31, 29, 15] selected neural units with specific meanings from CNNs for various applications.

Model diagnosis: Many statistical methods [28, 33, 1] have been proposed to analyze CNN features. The LIME method proposed by Ribeiro *et al.* [18], influence functions [10] and gradient-based visualization methods [5, 21] and [13] extracted image regions that were responsible for each network output, in order to interpret network representations. These methods require people to manually check image regions accountable for the label prediction for each testing image. [9] extracted relationships between representations of various categories from a CNN. In contrast, given an interpretable CNN, people can directly identify object parts (filters) that are used for decisions during the inference procedure.

Learning a better representation: Unlike the diagnosis and/or visualization of pre-trained CNNs, some approaches are developed to learn more meaningful representations. [19] required people to label dimensions of the input that were related to each output, in order to learn a better model. Hu *et al.* [8] designed some logic rules for network outputs, and used these rules to regularize the learning process. Stone *et al.* [27] learned CNN representations with better object compositionality, and Liao *et al.* [16] learned compact CNN representations, but they did not make filters obtain explicit part-level or texture-level semantics. Sabour *et al.* [20] proposed a capsule model, which used a dynamic routing mechanism to parse the entire object into a parsing tree of capsules, and each capsule may encode a specific meaning. In this study, we invent a generic loss to regularize the representation of a filter to improve its interpretability.

We can understand the interpretable CNN from the per-

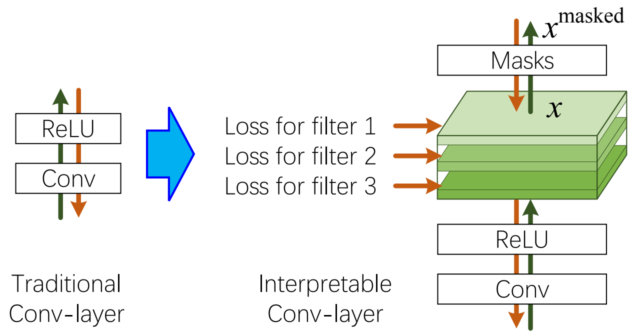

Figure 2. Structures of an ordinary conv-layer and an interpretable conv-layer. Green and red lines indicate the forward and backward propagations, respectively. During the forward propagation, our CNN assigns each interpretable filter with a specific mask *w.r.t.* each input image in an online manner.

spective of the information bottleneck [32] as follows. 1) Our interpretable filters selectively model the most distinct parts of each category to minimize the conditional entropy of the final classification given feature maps of a conv-layer. 2) Each filter represents a single part of an object, which maximizes the mutual information between the input image and middle-layer feature maps (*i.e.* "forgetting" as much irrelevant information as possible).

3. Algorithm

Given a target conv-layer of a CNN, we expect each filter in the conv-layer to be activated by a certain object part of a certain category, while remain inactivated on images of other categories[1]. Let $\mathbf{I}$ denote a set of training images, where $\mathbf{I}_c \subset \mathbf{I}$ represents the subset that belongs to category c, $(c = 1, 2, \ldots, C)$. Theoretically, we can use different types of losses to learn CNNs for multi-class classification, single-class classification (*i.e.* $c = 1$ for images of a category and $c = 2$ for random images), and other tasks.

Fig. 2 shows the structure of our interpretable conv-layer. In the following paragraphs, we focus on the learning of a single filter f in the target conv-layer. We add a loss to the feature map x of the filter f after the ReLu operation.

During the forward propagation, given each input image I, the CNN computes a feature map x of the filter f after the ReLu operation, where x is an $n\times n$ matrix, $x_{ij} \geq 0$. Our method estimates the potential position of the object part on the feature map x as the neural unit with the strongest activation $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_{ij}$, $1 \leq i, j \leq n$. Then, based on the estimated part position $\hat{\mu}$, the CNN assigns a specific mask with x to filter out noisy activations.

Because f's corresponding object part may appear at the n^2 different locations given different images, we design n^2 templates for f, $\{T_{\mu_1}, T_{\mu_2}, \ldots, T_{\mu_{n^2}}\}$. As shown in Fig. 3, each template T_{μ_i} is an $n \times n$ matrix, and it describes the ideal distribution of activations for the feature map x when the target part mainly triggers the i-th unit in x. Our method selects the template $T_{\hat{\mu}}$ *w.r.t.* the part position $\hat{\mu}$ from the n^2 templates as the mask. We compute $x^{\text{masked}} = \max\{x \circ T_{\hat{\mu}}, 0\}$ as the output masked feature map, where $\circ$ denotes the Hadamard (element-wise) product.

Fig. 4 visualizes the masks $T_{\hat{\mu}}$ chosen for different images, and compares the original and masked feature maps. The CNN selects different templates for different images. The mask operation supports gradient back-propagations.

During the back-propagation process, our loss pushes filter f to represent a specific object part of the category c and keep silent on images of other categories. Please see Section 3.1 for the determination of the category c for filter f. Let $\mathbf{X} = \{x | x = f(I), I \in \mathbf{I}\}$ denote feature maps of f after an ReLU operation for different training images. Given an input image I, if $I \in \mathbf{I}_c$, we expect the feature map $x = f(I)$ to exclusively activated at the target part's location; otherwise, the feature map keeps inactivated. In other words, if $I \in \mathbf{I}_c$, the feature map x is expected to fit to the assigned template $T_{\hat{\mu}}$; if $I \notin \mathbf{I}_c$, we design a negative template T^- and hope that the feature map x matches to T^-. Note that during the forward propagation, our method omits the negative template. All feature maps, including those of other categories, select n^2 templates of $\{T_{\mu_i}\}$ as masks.

Thus, each feature map x is supposed to be exclusively fit to one of the n^2+1 templates $T \in \mathbf{T} = \{T^-, T_{\mu_1}, T_{\mu_2}, \ldots, T_{\mu_{n^2}}\}$. We formulate the loss for f as the minus mutual information between $\mathbf{X}$ and $\mathbf{T}$, *i.e.* $-MI(\mathbf{X}; \mathbf{T})$.

$$\mathbf{Loss}_f = -MI(\mathbf{X}; \mathbf{T}) = -\sum_T p(T) \sum_x p(x|T) \log \frac{p(x|T)}{p(x)} \quad (1)$$

The prior probability of a template is given as $p(T_\mu) = \frac{\alpha}{n^2}$, $p(T^-) = 1 - \alpha$, where α is a constant prior likelihood. The fitness between a feature map x and a template T is measured as the conditional likelihood $p(x|T)$.

$$\forall T \in \mathbf{T}, \qquad p(x|T) = \frac{1}{Z_T} \exp\left[tr(x \cdot T)\right] \quad (2)$$

where $Z_T = \sum_{x \in \mathbf{X}} \exp[tr(x \cdot T)]$. $x \cdot T$ indicates the multiplication between x and T; $tr(\cdot)$ indicates the trace of a matrix, and $tr(x \cdot T) = \sum_{ij} x_{ij} t_{ij}$. $p(x) = \sum_T p(T) p(x|T)$.

Part templates: As shown in Fig. 3, a negative template is given as $T^- = (t^-_{ij})$, $t^-_{ij} = -\tau < 0$, where τ is a positive constant. A positive template corresponding to μ is given as $T_\mu = (t^+_{ij})$, $t^+_{ij} = \tau \cdot \max(1 - \beta \frac{\|[i,j]-\mu\|_1}{n}, -1)$, where $\|\cdot\|_1$ denotes the L-1 norm distance; β is a constant parameter.

3.1. Learning

We train the interpretable CNN via an end-to-end manner. During the forward-propagation process, each filter in the CNN passes its information in a bottom-up manner, just like traditional CNNs. During the back-propagation, each filter in an interpretable conv-layer receives gradients *w.r.t.*

[1] To avoid ambiguity, we evaluate or visualize the semantic meaning of each filter by using the feature map after the ReLU and mask operations.

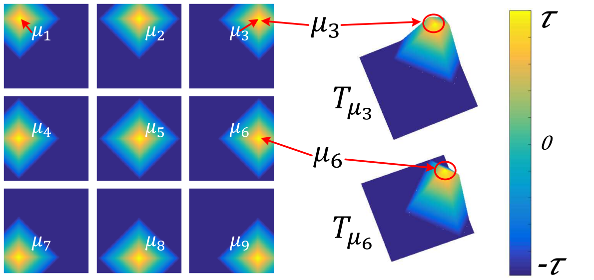

Figure 3. Templates of T_{μ_i}. Each template T_{μ_i} matches to a feature map x when the target part mainly triggers the i-th unit in x. In fact, the algorithm also supports a round template based on the L-2 norm distance. Here, we use the L-1 norm distance instead to speed up the computation.

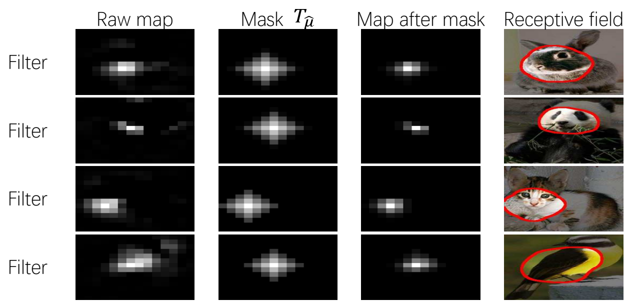

Figure 4. Given an input image I, from the left to the right, we consequently show the feature map of a filter after the ReLU layer x, the assigned mask $T_{\hat{\mu}}$, the masked feature map x^{masked}, and the image-resolution RF of activations in x^{masked} computed by [40].

its feature map x from both the final task loss $\mathbf{L}(\hat{y}_k, y_k^*)$ on the k-th sample and the filter loss, $\mathbf{Loss}_f$, as follows:

$$\frac{\partial \mathbf{Loss}}{\partial x_{ij}} = \lambda \frac{\partial \mathbf{Loss}_f}{\partial x_{ij}} + \frac{1}{N} \sum_{k=1}^{N} \frac{\partial \mathbf{L}(\hat{y}_k, y_k^*)}{\partial x_{ij}} \tag{3}$$

where λ is a weight. Then, we back propagate $\frac{\partial \mathbf{Loss}}{\partial x_{ij}}$ to lower layers and compute gradients *w.r.t.* feature maps and Xgradients *w.r.t.* parameters in lower layers to update the CNN.

For implementation, gradients of $\mathbf{Loss}_f$ *w.r.t.* each element x_{ij} of feature map x are computed as follows.

$$\frac{\partial \mathbf{Loss}_f}{\partial x_{ij}} = \frac{1}{Z_T} \sum_T p(T) t_{ij} e^{tr(x \cdot T)} \Big\{ tr(x \cdot T) - \log \left[Z_T p(x) \right] \Big\}$$
$$\approx \frac{p(\hat{T}) \hat{t}_{ij}}{Z_{\hat{T}}} e^{tr(x \cdot \hat{T})} \Big\{ tr(x \cdot \hat{T}) - \log Z_{\hat{T}} - \log p(x) \Big\} \tag{4}$$

where $\hat{T}$ is the target template for feature map x. If the given image I belongs to the target category of filter f, then $\hat{T} = T_{\hat{\mu}}$, where $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_{ij}$. If image I belongs to other categories, then $\hat{T} = T^-$. Considering $\forall T \in \mathbf{T} \setminus \{\hat{T}\}$, $e^{tr(x \cdot \hat{T})} \gg e^{tr(x \cdot T)}$ after initial learning episodes, we make the above approximation to simplify the computation. Because Z_T is computed using numerous feature maps, we can roughly treat Z_T as a constant to compute gradients in the above equation. We gradually update the value of Z_T during the training process[2]. Similarly, we can also approximate $p(x)$ without huge computation[2].

Determining the target category for each filter: We need to assign each filter f with a target category $\hat{c}$ to approximate gradients in Eqn. (4). We simply assign the filter f with the category $\hat{c}$ whose images activate f the most, *i.e.* $\hat{c} = \text{argmax}_c \text{mean}_{x=f(I):I \in \mathbf{I}_c} \sum_{ij} x_{ij}$.

4. Understanding of the loss

In fact, the loss in Eqn. (1) can be re-written as

$$\mathbf{Loss}_f = -H(\mathbf{T}) + H(\mathbf{T}'|\mathbf{X}) + \sum_x p(\mathbf{T}^+, x) H(\mathbf{T}^+|X = x) \tag{5}$$

Where $\mathbf{T}' = \{T^-, \mathbf{T}^+\}$. $H(\mathbf{T}) = -\sum_{T \in \mathbf{T}} p(T) \log p(T)$ is a constant prior entropy of part templates.

Low inter-category entropy: The second term $H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X})$ is computed as

$$H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X}) = -\sum_x p(x) \sum_{T \in \{T^-, \mathbf{T}^+\}} p(T|x) \log p(T|x) \tag{6}$$

where $\mathbf{T}^+ = \{T_{\mu_1}, \ldots, T_{\mu_{n^2}}\} \subset \mathbf{T}$, $p(\mathbf{T}^+|x) = \sum_\mu p(T_\mu|x)$. We define the set of all positive templates $\mathbf{T}^+$ as a single label to represent category c. We use the negative template T^- to denote other categories. This term encourages a low conditional entropy of inter-category activations, *i.e.* a well-learned filter f needs to be exclusively activated by a certain category c and keep silent on other categories. We can use a feature map x of f to identify whether the input image belongs to category c or not, *i.e.* x fitting to either $T_{\hat{\mu}}$ or T^-, without great uncertainty.

Low spatial entropy: The third term in Eqn. (5) is given as

$$H(\mathbf{T}^+|X = x) = \sum_\mu \tilde{p}(T_\mu|x) \log \tilde{p}(T_\mu|x) \tag{7}$$

where $\tilde{p}(T_\mu|x) = \frac{p(T_\mu|x)}{p(\mathbf{T}^+|x)}$. This term encourages a low conditional entropy of spatial distribution of x's activations. *I.e.* given an image $I \in \mathbf{I}_c$, a well-learned filter should only be activated by a single region $\hat{\mu}$ of the feature map x, instead of being repetitively triggered at different locations.

5. Experiments

In experiments, to demonstrate the broad applicability of our method, we applied our method to CNNs with four types of structures. We used object images in three different benchmark datasets to learn interpretable CNNs for single-category classification and multi-category classification. We visualized feature maps of filters in interpretable

[2]We can use a subset of feature maps to approximate the value of Z_T, and continue to update Z_T when we receive more feature maps during the training process. Similarly, we can approximate $p(x)$ using a subset of feature maps. We compute $p(x) = \sum_T p(T) p(x|T) = \sum_T p(T) \frac{\exp[tr(x \cdot T)]}{Z_T} \approx \sum_T p(T) \text{mean}_x \frac{\exp[tr(x \cdot T)]}{Z_T}$.

conv-layers to illustrate semantic meanings of these filters. We used two types of metrics, *i.e.* the object-part interpretability and the location instability, to evaluate the clarity of the part semantics of a filter. Experiments showed that filters in our interpretable CNNs were much more semantically meaningful than those in ordinary CNNs.

Three benchmark datasets: Because we needed ground-truth annotations of object landmarks[3] (parts) to evaluate the semantic clarity of each filter, we chose three benchmark datasets with part annotations for training and testing, including the ILSVRC 2013 DET Animal-Part dataset [36], the CUB200-2011 dataset [30], and the Pascal VOC Part dataset [3]. As discussed in [3, 36], non-rigid parts of animal categories usually present great challenges for part localization. Thus, we followed [3, 36] to select the 37 animal categories in the three datasets for evaluation.

All the three datasets provide ground-truth bounding boxes of entire objects. For landmark annotations, the ILSVRC 2013 DET Animal-Part dataset [36] contains ground-truth bounding boxes of heads and legs of 30 animal categories. The CUB200-2011 dataset [30] contains a total of 11.8K bird images of 200 species, and the dataset provides center positions of 15 bird landmarks. The Pascal VOC Part dataset [3] contain ground-truth part segmentations of 107 object landmarks in six animal categories.

Four types of CNNs: We modified four typical CNNs, *i.e.* the AlexNet [12], the VGG-M [25], the VGG-S [25], the VGG-16 [25], into interpretable CNNs. Considering that skip connections in residual networks [7] usually make a single feature map encode patterns of different filters, in this study, we did not test the performance on residual networks to simplify the story. Given a certain CNN, we modified all filters in the top conv-layer of the original network into interpretable filters. Then, we inserted a new conv-layer with M filters above the original top conv-layer, where M is the channel number of the input of the new conv-layer. We also set filters in the new conv-layer as interpretable filters. Each filter was a $3\times3\times M$ tensor with a bias term. We added zero padding to input feature maps to ensure that output feature maps were of the same size as the input.

Implementation details: We set parameters as $\tau=\frac{0.5}{n^2}$, $\alpha=\frac{n^2}{1+n^2}$, and $\beta=4$. We updated weights of filter losses *w.r.t.* magnitudes of neural activations in an online manner, $\lambda\propto\frac{1}{t}\text{mean}_{x\in\mathbf{X}}\max_{i,j}x_{ij}$ in the t-th epoch. We initialized parameters of fully-connected (FC) layers and the new conv-layer, and loaded parameters of other conv-layers from a traditional CNN that was pre-trained using 1.2M ImageNet images in [12, 25]. We then fine-tuned parameters of all layers in the interpretable CNN using training images in the dataset. To enable a fair comparison, traditional CNNs were also fine-tuned by initializing FC-layer parameters and loading conv-layer parameters.

[3]To avoid ambiguity, a landmark is referred to as the *central position* of a semantic part (a part with an explicit name, *e.g.* a head, a tail). In contrast, the part corresponding to a filter does not have an explicit name.

5.1. Experiments

Single-category classification: We learned four types of interpretable CNNs based on the AlexNet, VGG-M, VGG-S, and VGG-16 structures to classify each category in the ILSVRC 2013 DET Animal-Part dataset [36], the CUB200-2011 dataset [30], and the Pascal VOC Part dataset [3]. Besides, we also learned ordinary AlexNet, VGG-M, VGG-S, and VGG-16 networks using the same data for comparison. We used the logistic log loss for single-category classification. Following experimental settings in [36, 35], we cropped objects of the target category based on their bounding boxes as positive samples with ground-truth labels $y^*=+1$. Images of other categories were regarded as negative samples with ground-truth labels $y^*=-1$.

Multi-category classification: We used the six animal categories in the Pascal VOC Part dataset [3] and the thirty categories in the ILSVRC 2013 DET Animal-Part dataset [36] respectively, to learn CNNs for multi-category classification. We learned interpretable CNNs based on the VGG-M, VGG-S, and VGG-16 structures. We tried two types of losses, *i.e.* the softmax log loss and the logistic log loss[4] for multi-class classification.

5.2. Quantitative evaluation of part interpretability

As discussed in [2], filters in low conv-layers usually represent simple patterns or object details (*e.g.* edges, simple textures, and colors), whereas filters in high conv-layers are more likely to represent complex, large-scale parts. Therefore, in experiments, we evaluated the clarity of part semantics for the top conv-layer of a CNN. We used the following two metrics for evaluation.

5.2.1 Evaluation metric: part interpretability

We followed the metric proposed by Bau *et al.* [2] to measure the object-part interpretability of filters. We briefly introduce this evaluation metric as follows. For each filter f, we computed its feature maps $\mathbf{X}$ after ReLu/mask operations on different input images. Then, the distribution of activation scores in all positions of all feature maps was computed. [2] set an activation threshold T_f such that $p(x_{ij}>T_f)=0.005$, so as to select top activations from all spatial locations $[i,j]$ of all feature maps $x\in\mathbf{X}$ as valid map regions corresponding to f's semantics. Then, [2] scaled up low-resolution valid map regions to the image resolution,

[4]We considered the output y_c for each category c independent to outputs for other categories, thereby a CNN making multiple independent single-class classifications for each image. Table 7 reported the average accuracy of the multiple classification outputs of an image.

	bird	cat	cow	dog	horse	sheep	**Avg.**
AlexNet	0.332	0.363	0.340	0.374	0.308	0.373	0.348
AlexNet, interpretable	**0.770**	**0.565**	**0.618**	**0.571**	**0.729**	**0.669**	**0.654**
VGG-16	0.519	0.458	0.479	0.534	0.440	0.542	0.495
VGG-16, interpretable	**0.818**	**0.653**	**0.683**	**0.900**	**0.795**	**0.772**	**0.770**
VGG-M	0.357	0.365	0.347	0.368	0.331	0.373	0.357
VGG-M, interpretable	**0.821**	**0.632**	**0.634**	**0.669**	**0.736**	**0.756**	**0.708**
VGG-S	0.251	0.269	0.235	0.275	0.223	**0.287**	0.257
VGG-S, interpretable	**0.526**	**0.366**	**0.291**	**0.432**	**0.478**	0.251	**0.390**

Table 1. Part interpretability of filters in CNNs for single-category classification based on the Pascal VOC Part dataset [3].

thereby obtaining the receptive field (RF)[5] of valid activations on each image. The RF on image I, denoted by S_f^I, described the part region of f.

The compatibility between each filter f and the k-th part on image I was reported as an intersection-over-union score $IoU_{f,k}^I = \frac{\|S_f^I \cap S_k^I\|}{\|S_f^I \cup S_k^I\|}$, where S_k^I denotes the ground-truth mask of the k-th part on image I. Given an image I, we associated filter f with the k-th part if $IoU_{f,k}^I > 0.2$. The criterion of $IoU_{f,k}^I > 0.2$ for part association is stricter than $IoU_{f,k}^I > 0.04$ used in [2]. It is because compared to other CNN semantics discussed in [2] (such as colors and textures), object-part semantics requires a stricter criterion. We computed the probability of the k-th part being associating with the filter f as $P_{f,k} = \text{mean}_{I:\text{with k-th part}} \mathbf{1}(IoU_{f,k}^I > 0.2)$. Note that one filter might be associated with multiple object parts in an image. Among all parts, we reported the highest probability of part association as the interpretability of filter f, *i.e.* $P_f = \max_k P_{f,k}$.

For single-category classification, we used testing images of the target category for evaluation. In the Pascal VOC Part dataset [3], we used four parts for the *bird* category. We merged ground-truth regions of the head, beak, and l/r-eyes as the head part, merged regions of the torso, neck, and l/r-wings as the torso part, merged regions of l/r-legs/feet as the leg part, and used tail regions as the fourth part. We used five parts for the *cat* category. We merged regions of the head, l/r-eyes, l/r-ears, and nose as the head part, merged regions of the torso and neck as the torso part, merged regions of frontal l/r-legs/paws as the frontal legs, merged regions of back l/r-legs/paws as the back legs, and used the tail as the fifth part. We used four parts for the *cow* category, which were defined in a similar way to the cat category. We added l/r-horns to the head part and omitted the tail part. We applied five parts of the *dog* category in the same way as the cat category. We applied four parts of both the *horse* and *sheep* categories in the same way as the cow category. We computed the average part interpretability P_f over all filters for evaluation.

For multi-category classification, we first assigned each filter f with a target category $\hat{c}$, *i.e.* the category that activated the filter most $\hat{c} = \text{argmax}_c \text{mean}_{x=f(I):I\in\mathbf{I}_c} \sum_{i,j} x_{ij}$. Then, we computed the object-part interpretability using images of category $\hat{c}$, as introduced above.

[5][40] computes the RF when the filter represents an object part. Fig. 5 used RFs computed by [40] to visualize filters. However, when a filter in an ordinary CNN does not have consistent contours, it is difficult for [40] to align different images to compute an average RF. Thus, for ordinary CNNs, we simply used a round RF for each valid activation. We overlapped all activated RFs in a feature map to compute the final RF as mentioned in [2]. For a fair comparison, in Section 5.2.1, we uniformly applied these RFs to both interpretable CNNs and ordinary CNNs.

Network	Logistic log loss[4]	Softmax log loss
VGG-16	0.710	0.723
VGG-16, interpretable	**0.938**	**0.897**
VGG-M	0.478	0.502
VGG-M, interpretable	**0.770**	**0.734**
VGG-S	0.479	0.435
VGG-S, interpretable	**0.572**	**0.601**

Table 2. Part interpretability of filters in CNNs that are trained for multi-category classification based on the VOC Part dataset [3]. Filters in our interpretable CNNs exhibited significantly better part interpretability than ordinary CNNs in all comparisons.

5.2.2 Evaluation metric: location instability

The second metric measures the instability of part locations, which was proposed in [35]. Given a feature map x of filter f, we regarded the unit $\hat{\mu}$ with the highest activation as the location inference of f. We assumed that if f consistently represented the same object part through different objects, then distances between the inferred part location $\hat{\mu}$ and some object landmarks[3] should not change a lot among different objects. For example, if f represented the shoulder, then the distance between the shoulder and the head should keep stable through different objects.

Therefore, [35] computed the deviation of the distance between the inferred position $\hat{\mu}$ and a specific ground-truth landmark among different images, and used the average deviation *w.r.t.* various landmarks to evaluate the location instability of f. Let $d_I(p_k, \hat{\mu}) = \frac{\|\mathbf{p}_k - \mathbf{p}(\hat{\mu})\|}{\sqrt{w^2+h^2}}$ denote the normalized distance between the inferred part and the k-th landmark $\mathbf{p}_k$ on image I, where $\mathbf{p}(\hat{\mu})$ denotes the center of the unit $\hat{\mu}$'s RF when we backward propagated the RF to the image plane. $\sqrt{w^2+h^2}$ denotes the diagonal length of the input image. We computed $D_{f,k} = \sqrt{\text{var}_I[d_I(p_k, \hat{\mu})]}$ as the *relative location deviation* of filter f *w.r.t.* the k-th landmark, where $\text{var}_I[d_I(p_k, \hat{\mu})]$ is referred to as the variation of the distance $d_I(p_k, \hat{\mu})$. Because each landmark could not appear in all testing images, for each filter f, the metric only used inference results with the top-100 highest activation scores $x_{\hat{\mu}}$ on images containing the k-th landmark to compute $D_{f,k}$. Thus, we used the average of relative location deviations of all the filters in a conv-layer *w.r.t.* all landmarks, *i.e.* $\text{mean}_f \text{mean}_{k=1}^K D_{f,k}$, to measure the location instability of f, where K denotes the number of landmarks.

More specifically, object landmarks for each category

	gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion	tiger	bear	rabb.	hams.	squi.
AlexNet	0.161	0.167	0.152	0.153	0.175	0.128	0.123	0.144	0.143	0.148	0.137	0.142	0.144	0.148	0.128	0.149
AlexNet, interpretable	**0.084**	**0.095**	**0.090**	**0.107**	**0.097**	**0.079**	**0.077**	**0.093**	**0.087**	**0.095**	**0.084**	**0.090**	**0.095**	**0.095**	**0.077**	**0.095**
VGG-16	0.153	0.156	0.144	0.150	0.170	0.127	0.126	0.143	0.137	0.148	0.139	0.144	0.143	0.146	0.125	0.150
VGG-16, interpretable	**0.076**	**0.099**	**0.086**	**0.115**	**0.113**	**0.070**	**0.084**	**0.077**	**0.069**	**0.086**	**0.067**	**0.097**	**0.081**	**0.079**	**0.066**	**0.065**
VGG-M	0.161	0.166	0.151	0.153	0.176	0.128	0.125	0.145	0.145	0.150	0.140	0.145	0.144	0.150	0.128	0.150
VGG-M, interpretable	**0.088**	**0.088**	**0.089**	**0.108**	**0.099**	**0.080**	**0.074**	**0.090**	**0.082**	**0.103**	**0.079**	**0.089**	**0.101**	**0.097**	**0.082**	**0.095**
VGG-S	0.158	0.166	0.149	0.151	0.173	0.127	0.124	0.143	0.142	0.148	0.138	0.142	0.143	0.148	0.128	0.146
VGG-S, interpretable	**0.087**	**0.101**	**0.093**	**0.107**	**0.096**	**0.084**	**0.078**	**0.091**	**0.082**	**0.101**	**0.082**	**0.089**	**0.097**	**0.091**	**0.076**	**0.098**
	horse	zebra	swine	hippo	catt.	sheep	ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.		**Avg.**
AlexNet	0.152	0.154	0.141	0.141	0.144	0.155	0.147	0.153	0.159	0.160	0.139	0.125	0.140	0.125		0.146
AlexNet, interpretable	**0.098**	**0.084**	**0.091**	**0.089**	**0.097**	**0.101**	**0.085**	**0.102**	**0.104**	**0.095**	**0.090**	**0.085**	**0.084**	**0.073**		**0.091**
VGG-16	0.150	0.153	0.141	0.140	0.140	0.150	0.144	0.149	0.154	0.163	0.136	0.129	0.143	0.125		0.144
VGG-16, interpretable	**0.106**	**0.077**	**0.094**	**0.083**	**0.102**	**0.097**	**0.091**	**0.105**	**0.093**	**0.100**	**0.074**	**0.084**	**0.067**	**0.063**		**0.085**
VGG-M	0.151	0.158	0.140	0.140	0.143	0.155	0.146	0.154	0.160	0.161	0.140	0.126	0.142	0.127		0.147
VGG-M, interpretable	**0.095**	**0.080**	**0.095**	**0.084**	**0.092**	**0.094**	**0.077**	**0.104**	**0.102**	**0.093**	**0.086**	**0.087**	**0.089**	**0.068**		**0.090**
VGG-S	0.149	0.155	0.139	0.140	0.141	0.155	0.143	0.154	0.158	0.157	0.140	0.125	0.139	0.125		0.145
VGG-S, interpretable	**0.096**	**0.080**	**0.092**	**0.088**	**0.094**	**0.101**	**0.077**	**0.102**	**0.105**	**0.094**	**0.090**	**0.086**	**0.078**	**0.072**		**0.090**

Table 3. Location instability of filters ($\mathbf{E}_{f,k}[D_{f,k}]$) in CNNs that are trained for single-category classification using the ILSVRC 2013 DET Animal-Part dataset [36]. Filters in our interpretable CNNs exhibited significantly lower localization instability than ordinary CNNs.

	bird	cat	cow	dog	horse	sheep	**Avg.**
AlexNet	0.153	0.131	0.141	0.128	0.145	0.140	0.140
AlexNet, interpretable	**0.090**	**0.089**	**0.090**	**0.088**	**0.087**	**0.088**	**0.088**
VGG-16	0.145	0.133	0.146	0.127	0.143	0.143	0.139
VGG-16, interpretable	**0.101**	**0.098**	**0.105**	**0.074**	**0.097**	**0.100**	**0.096**
VGG-M	0.152	0.132	0.143	0.130	0.145	0.141	0.141
VGG-M, interpretable	**0.086**	**0.094**	**0.090**	**0.087**	**0.084**	**0.084**	**0.088**
VGG-S	0.152	0.131	0.141	0.128	0.144	0.141	0.139
VGG-S, interpretable	**0.089**	**0.092**	**0.092**	**0.087**	**0.086**	**0.088**	**0.089**

Table 4. Location instability of filters ($\mathbf{E}_{f,k}[D_{f,k}]$) in CNNs that are trained for single-category classification using the Pascal VOC Part dataset [3]. Filters in our interpretable CNNs exhibited significantly lower localization instability than ordinary CNNs.

Network	Avg. location instability
AlexNet	0.150
AlexNet, interpretable	**0.070**
VGG-16	0.137
VGG-16, interpretable	**0.076**
VGG-M	0.148
VGG-M, interpretable	**0.065**
VGG-S	0.148
VGG-S, interpretable	**0.073**

Table 5. Location instability of filters ($\mathbf{E}_{f,k}[D_{f,k}]$) in CNNs for single-category classification using the CUB200-2011 dataset.

were selected as follows. For the ILSVRC 2013 DET Animal-Part dataset [36], we used the *head* and *frontal legs* of each category as landmarks for evaluation. For the Pascal VOC Part dataset [3], we selected the *head*, *neck*, and *torso* of each category as the landmarks. For the CUB200-2011 dataset [30], we used ground-truth positions of the *head*, *back*, *tail* of birds as landmarks. It was because these landmarks appeared on testing images most frequently.

For multi-category classification, we needed to determine two terms for each filter f, *i.e.* 1) the category that f mainly represented and 2) the relative location deviation

Dataset	ILSVRC Part [36]	Pascal VOC Part [3]	
Network	Logistic log loss[4]	Logistic log loss[4]	Softmax log loss
VGG-16	–	0.128	0.142
interpretable	–	**0.073**	**0.075**
VGG-M	0.167	0.135	0.137
interpretable	**0.096**	**0.083**	**0.087**
VGG-S	0.131	0.138	0.138
interpretable	**0.083**	**0.078**	**0.082**

Table 6. Location instability of filters ($\mathbf{E}_{f,k}[D_{f,k}]$) in CNNs that are trained for multi-category classification. Filters in our interpretable CNNs exhibited significantly lower localization instability than ordinary CNNs in all comparisons.

$D_{f,k}$ *w.r.t.* landmarks in f's target category. Because filters in ordinary CNNs did not exclusively represent a single category, we simply assigned filter f with the category whose landmarks can achieve the lowest location deviation to simplify the computation. *I.e.* we used the average location deviation $\text{mean}_f \min_c \text{mean}_{k\in Part_c} D_{f,k}$ to evaluate the location instability, where $Part_c$ denotes the set of part indexes belonging to category c.

5.2.3 Experimental results and analysis

Tables 1 and 2 show part interpretability of CNNs for single-category classification and that of CNNs for multi-category classification, respectively. Tables 3, 4, and 5 list location instability of CNNs for single-category classification. Table 6 compares location instability of CNNs for multi-category classification. Our interpretable CNNs exhibited much higher part interpretability and much lower location instability than ordinary CNNs in almost all comparisons. Table 7 compares classification accuracy of different CNNs. Ordinary CNNs performed better in single-category classification, while interpretable CNNs outperformed in multi-category classification.

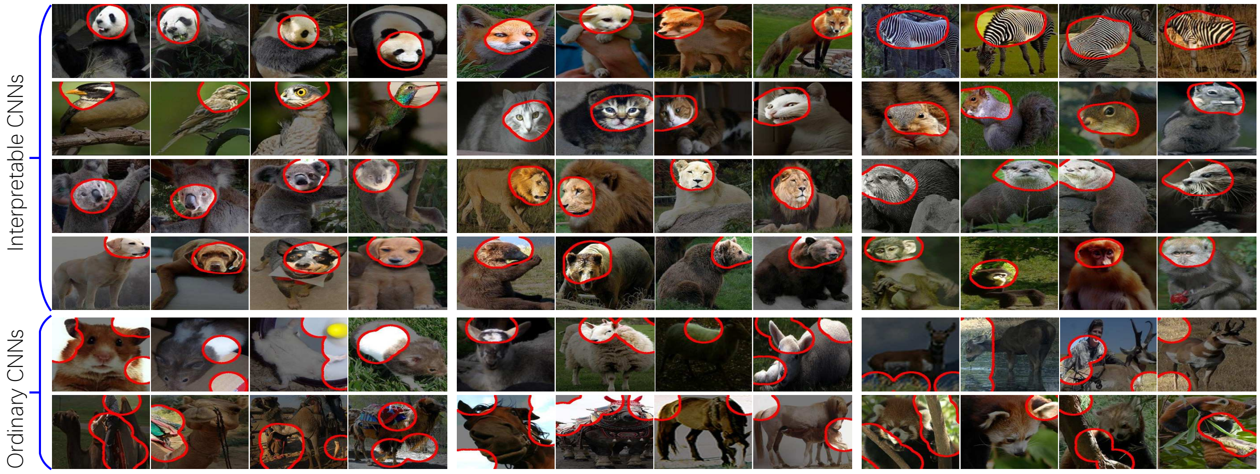

Figure 5. Visualization of filters in top conv-layers. We used [40] to estimate the image-resolution receptive field of activations in a feature map to visualize a filter's semantics. The top four rows visualize filters in interpretable CNNs, and the bottom two rows correspond to filters in ordinary CNNs. We found that interpretable CNNs usually encoded head patterns of animals in its top conv-layer for classification, although no part annotations were used to train the CNN.

	multi-category			single-category		
	ILSVRC Part	VOC Part		ILSVRC Part	VOC Part	CUB200
	logistic[4]	logistic[4]	softmax			
AlexNet	–	–	–	**96.28**	**95.40**	**95.59**
interpretable	–	–	–	95.38	93.93	95.35
VGG-M	96.73	93.88	81.93	**97.34**	**96.82**	**97.34**
interpretable	**97.99**	**96.19**	**88.03**	95.77	94.17	96.03
VGG-S	96.98	94.05	78.15	**97.62**	**97.74**	**97.24**
interpretable	**98.72**	**96.78**	**86.13**	95.64	95.47	95.82
VGG-16	–	97.97	89.71	**98.58**	**98.66**	**98.91**
interpretable	–	**98.50**	**91.60**	96.67	95.39	96.51

Table 7. Classification accuracy based on different datasets. In single-category classification, ordinary CNNs performed better, while in multi-category classification, interpretable CNNs exhibited superior performance.

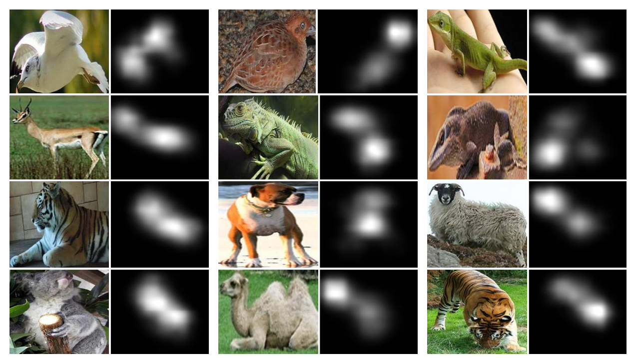

Figure 6. Heat maps for distributions of object parts that are encoded in interpretable filters. We use all filters in the top conv-layer to compute the heat map.

5.3. Visualization of filters

We followed the method proposed by Zhou *et al.* [40] to compute the RF of neural activations (after ReLU and mask operations) of a filter, which was scaled up to the image resolution. Fig. 5 shows RFs[5] of filters in top conv-layers of CNNs, which were trained for single-category classification. Filters in interpretable CNNs were mainly activated by a certain object part, whereas feature maps of ordinary CNNs after ReLU operations usually did not describe explicit semantic meanings. Fig. 6 shows heat maps for distributions of object parts that were encoded in interpretable filters. Interpretable filters usually selectively modeled distinct object parts of a category and ignored other parts.

6. Conclusion and discussions

In this paper, we have proposed a general method to modify traditional CNNs to enhance their interpretability. As discussed in [2], besides the discrimination power, the interpretability is another crucial property of a network. We design a loss to push a filter in high conv-layers toward the representation of an object part without any part annotations for supervision. It is because compared to low conv-layers, high conv-layers are more likely to represent large-scale parts. Experiments have shown that our interpretable CNNs encoded more semantically meaningful knowledge in high conv-layers than traditional CNNs. When we use an interpretable CNN to classify a large number of categories simultaneously, filters in a conv-layer are assigned with different categories, which makes each category corresponds to only a few filters. In this case, the CNN's classification accuracy may decrease a bit.

Acknowledgement

This work is supported by ONR MURI project N00014-16-1-2007 and DARPA XAI Award N66001-17-2-4029, and NSF IIS 1423305.

References

[1] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. *In ICCV*, 2015. 2

[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*, 2017. 1, 5, 6, 8

[3] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *In CVPR*, 2014. 5, 6, 7

[4] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016. 1, 2

[5] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In arXiv:1704.03296v1*, 2017. 1, 2

[6] Y. Goyal, A. Mohapatra, D. Parikh, and D. Batra. Towards transparent ai systems: Interpreting visual question answering models. *In arXiv:1608.08974v2*, 2016. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2016. 1, 5

[8] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. P. Xing. Harnessing deep neural networks with logic rules. *In arXiv:1603.06318v2*, 2016. 2

[9] V. K. Ithapu. Decoding the deep: Exploring class hierarchies of deep representations using multiresolution matrix factorization. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017. 2

[10] P. Koh and P. Liang. Understanding black-box predictions via influence functions. *In ICML*, 2017. 1, 2

[11] S. Kolouri, C. E. Martin, and H. Hoffmann. Explaining distributed neural activations via unsupervised learning. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017. 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012. 1, 5

[13] D. Kumar, A. Wong, and G. W. Taylor. Explaining the unexplained: A class-enhanced attentive response (clear) approach to understanding deep neural networks. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017. 2

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 1998. 1

[15] B. J. Lengerich, S. Konam, E. P. Xing, S. Rosenthal, and M. Veloso. Visual explanations for convolutional neural networks via input resampling. *In ICML Workshop on Visualization for Deep Learning*, 2017. 2

[16] R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. *In NIPS*, 2016. 2

[17] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *In CVPR*, 2015. 1, 2

[18] M. T. Ribeiro, S. Singh, and C. Guestrin. “why should i trust you?” explaining the predictions of any classifier. *In KDD*, 2016. 1, 2

[19] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *In arXiv:1703.03717v1*, 2017. 2

[20] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *In NIPS*, 2017. 2

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In arXiv:1610.02391v3*, 2017. 1, 2

[22] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015. 2

[23] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. *In ACCV*, 2014. 2

[24] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. *In arXiv:1312.6034*, 2013. 1, 2

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015. 5

[26] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. *In ECCV*, 2012. 2

[27] A. Stone, H. Wang, Y. Liu, D. S. Phoenix, and D. George. Teaching compositionality to cnns. *In CVPR*, 2017. 2

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *In arXiv:1312.6199v4*, 2014. 2

[29] C. Ventura, D. Masip, and A. Lapedriza. Interpreting cnn models for apparent personality trait regression. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017. 2

[30] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, In California Institute of Technology, 2011. 5, 7

[31] A. S. Wicaksana and C. C. S. Liem. Human-explainable features for job candidate screening prediction. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017. 2

[32] N. Wolchover. New theory cracks open the black box of deep learning. *In Quanta Magazine*, 2017. 3

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *In NIPS*, 2014. 2

[34] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014. 1, 2

[35] Q. Zhang, R. Cao, F. Shi, Y. Wu, and S.-C. Zhu. Interpreting cnn knowledge via an explanatory graph. *In AAAI*, 2018. 5, 6

[36] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable part graphs on convnets via multi-shot learning. *In AAAI*, 2016. 2, 5, 7

[37] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Mining object parts from cnns via active question-answering. *In CVPR*, 2017. 2

[38] Q. Zhang, W. Wang, and S.-C. Zhu. Examining cnn representations with respect to dataset bias. *In AAAI*, 2018. 2

[39] Q. Zhang and S.-C. Zhu. Visual interpretability for deep learning: a survey. *in Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018. 2

[40] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015. 2, 4, 6, 8

[41] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016. 2