Interpretable CNNs

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu, *Fellow, IEEE*

Abstract—This paper proposes a generic method to learn interpretable convolutional filters in a deep convolutional neural network (CNN), where each interpretable filter encodes features of a specific object part. Our method does not require additional annotations of object parts or textures for supervision. Instead, we use the same training data as traditional CNNs. Our method automatically assigns each interpretable filter in a high conv-layer with an object part of a certain category during the learning process. Such explicit knowledge representations in conv-layers of CNN help people clarify the logic encoded in the CNN, *i.e.* answering what patterns the CNN extracts from an input image and uses for prediction. We have tested our method using different benchmark CNNs with various structures to demonstrate the broad applicability of our method. Experiments have shown that our interpretable filters are much more semantically meaningful than traditional filters.

Index Terms—Convolutional Neural Networks, Interpretable Deep Learning

1 Introduction

In recent years, convolutional neural networks (CNNs) [7], [12], [15] have achieved superior performance in many visual tasks, such as object classification and detection. In spite of the good performance, a deep CNN has been considered a black-box model with weak feature interpretability for decades. Boosting the feature interpretability of a deep model gradually attracts increasing attention recently, but it presents significant challenges for state-of-the-art algorithms.

In this paper, we focus on a new task, *i.e.* without any additional annotations for supervision, revising a CNN to make its high conv-layers encode interpretable object-part knowledge. The revised CNN is termed an *interpretable CNN*. We expect the CNN to have some introspection of its feature representations during the end-to-end learning. In this way, the CNN can regularize its features in an online manner to boost its feature interpretability.

Note that our task of improving feature interpretability of a CNN is essentially different from the conventional visualization [4], [5], [18], [23], [26], [37] and diagnosis [2], [10], [17], [20] of pre-trained CNNs, which do not substantially boost feature interpretability.

Bau *et al.* [2] classified all potential semantics into the following six types, *objects*, *parts*, *scenes*, *textures*, *materials*, and *colors*. We can further summarize the semantics of *objects* and *parts* as part patterns with specific contours and consider the other four semantics as textural patterns without explicit shapes.

In addition, as discussed in [2], filters in low conv-layers usually describe textural patterns, while filters in high conv-layers are more likely to represent part patterns. Therefore, we aim to propose a method to ensure each filter in a high conv-layer to represent an object part.

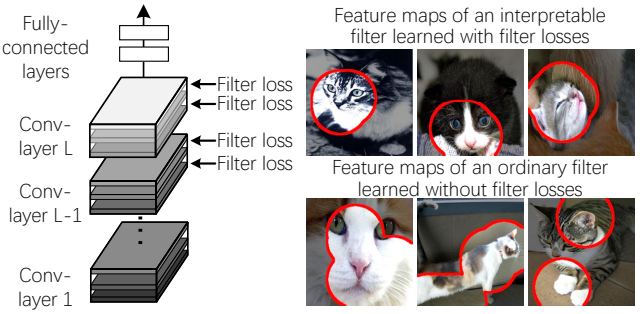


Fig. 1. Comparison of an interpretable filter's feature maps with a filter's feature maps in a traditional CNN.

Fig. 1 visualizes the difference between a traditional filter and our interpretable filter. In a traditional CNN, a filter usually describes a mixture of patterns. For example, the filter may be activated by both the head part and the leg part of a cat. In contrast, the filter in our interpretable CNN is expected to be activated by a certain part. The automatically learned object part may not have an explicit name, *e.g.* describing a partial region of a semantic part or the joint of two parts. However, we require the filter to consistently represent the same region of the object through different images. 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 a generic method to revise a CNN to boost feature interpretability, which can be broadly applied to different benchmark CNNs with various structures.
- Our method does not require any additional an-

• *Quanshi Zhang is with the John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, China. Ying Nian Wu and Song-Chun Zhu are with the University of California, Los Angeles, USA.*

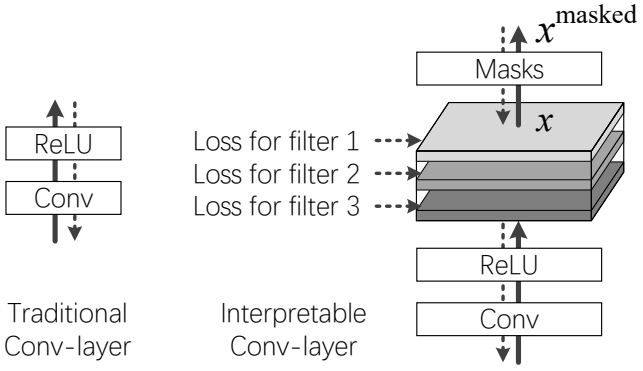


Fig. 2. Structures of an ordinary conv-layer and an interpretable conv-layer. Solid and dashed lines indicate the forward and backward propagations, respectively. During the forward propagation, our CNN assigns each interpretable filter with a specific mask *w.r.t.* each input image in an online manner.

notations of object parts for supervision. We use the same training samples as the original CNN for learning.

- The interpretable CNN does not change the loss function of the task.
- As an exploratory research, the design for boosting feature interpretability may hurt a bit the discrimination power. However, we need to control the decrease within a small range.

Method: As shown in Fig. 2, we propose a simple yet effective loss and add the loss to the feature map of each filter of a high conv-layer. The filter loss pushes the filter towards the representation of an object part.

Theoretically, we can prove that the loss encourages a low entropy of inter-category activations and a low entropy of spatial distributions of neural activations. In other words, (i) each filter must encode an object part of a single object category, instead of representing multiple categories; (ii) the filter must be triggered by a single object part, rather than be simultaneously triggered by different object regions. It is assumed that repetitive patterns on different object regions are more likely to describe low-level textures, instead of high-level parts.

Value of feature interpretability: In critical applications, the clear semantics in high conv-layers helps people trust a network's prediction. As analyzed in [42], a good performance on testing images cannot always ensure correct feature representations considering potential dataset bias. For example, in [42], a CNN used an unreliable context—eye features—to identify the "lipstick" attribute of a face image. Therefore, people need to semantically and visually explain what patterns are learned by the CNN.

In this study, clarifying semantic meanings of CNN features can explain the logic at the object-part level. Given an input image, our interpretable CNN can explicitly show the distribution of object parts that are used by the CNN for prediction.

Contributions: In this paper, we focus on a new task, *i.e.* end-to-end learning an interpretable CNN without any part annotations, where filters of high conv-layers represent specific object parts. We propose a simple yet effective method to learn interpretable filters, and the method can be broadly applied to different benchmark CNNs. Experiments show that our approach has significantly boosted feature interpretability of CNNs.

A preliminary version of this paper appeared in [43].

2 Related work

The interpretability and the discrimination power are two crucial aspects of a CNN [2]. In recent years, different methods are developed to explore the semantics hidden inside a CNN. Our previous paper [45] provides a comprehensive survey of recent studies in exploring visual interpretability of neural networks, including (i) the visualization and diagnosis of CNN representations, (ii) approaches for disentangling CNN representations into graphs or trees, (iii) the learning of CNNs with disentangled and interpretable representations, and (iv) 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 that is encoded by the filter. Gradient-based visualization [18], [26], [37] showed the appearance that maximized the score of a given unit. Furthermore, Bau *et al.* [2] defined and analyzed the interpretability of each filter. Recently, [19] provided tools to visualize filters of a CNN. Dosovitskiy *et al.* [4] proposed up-convolutional nets to invert feature maps of conv-layers to images. However, up-convolutional nets cannot mathematically ensure the visualization result reflects actual neural representations.

Although above studies can produce clear visualization results, theoretically, gradient-based visualization of a filter usually selectively visualizes the strongest activations of a filter in a high conv-layer, instead of illustrating knowledge hidden behind all activations of the filter; otherwise, the visualization result will be chaotic. Similarly, [2] selectively analyzed the semantics of the highest 0.5% activations of each filter. In comparisons, we aim to purify the semantic meaning of each filter in a high conv-layer, *i.e.* letting most activations of a filter be explainable, instead of extracting meaningful neural activations for visualization.

Pattern retrieval: Unlike passive visualization, some methods actively retrieve certain units with certain meanings from CNNs. Just like mid-level features [28] of images, pattern retrieval mainly focuses on mid-level representations in conv-layers. For example, Zhou *et al.* [46], [47] selected units

from feature maps to describe "scenes". Simon *et al.* discovered objects from feature maps of conv-layers [24], and selected certain filters to represent object parts [25]. Zhang *et al.* [39] extracted certain neural activations of a filter to represent object parts in a weakly-supervised manner. They also disentangled CNN representations via active question-answering and summarized the disentangled knowledge using an And-Or graph [40]. [41] used human interactions to refine the AOG representation of CNN knowledge. [6] used a gradient-based method to explain visual question-answering. Other studies [11], [16], [32], [34] selected filters or neural activations with specific meanings from CNNs for various applications. Unlike the retrieval of meaningful neural activations from noisy features, our method aims to substantially boost the interpretability of features in intermediate conv-layers.

Model diagnosis: Many approaches have been proposed to diagnose CNN features, including exploring semantic meanings of convolutional filters [30], evaluating the transferability of filters [36], and the analysis of feature distributions of different categories [1]. The LIME [20] and the SHAP [17] are general methods to extract input units of a neural network that are used for the prediction score. For CNNs oriented to visual tasks, gradient-based visualization methods [5], [23] and [13] extracted image regions that are responsible for the network output, in order to clarify the logic of network prediction. 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 or filters that are used for prediction.

As discussed by Zhang *et al.* [42], knowledge representations of a CNN may be significantly biased due to dataset bias, even though the CNN sometimes exhibits good performance. For example, a CNN may extract unreliable contextual features for prediction. Network-attack methods [10], [29], [30] diagnosed network representation flaws using adversarial samples of a CNN. For example, influence functions [10] can be used to generate adversarial samples, in order to fix the training set and further debug representations of a CNN. [14] discovered blind spots of knowledge representation of a pre-trained CNN in a weakly-supervised manner.

Learning interpretable feature representations: Unlike the diagnosis and visualization of pre-trained CNNs, some approaches were developed to learn meaningful feature representations in recent years. For example, [21] 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 logic rules to regularize network outputs during the learning process. Sabour *et al.* [22] proposed a capsule model, where each feature dimension of a capsule may represent a specific meaning. Similarly, we invent a generic filter loss to regularize the representation of a filter to improve its interpretability.

Furthermore, some method distilled CNN knowledge into another model with interpretable features for explanations. [31] distilled knowledge of a neural network into an additive model to explain the knowledge inside the network. [44] roughly represented the rationale of each CNN prediction using a semantic tree structure. Each node in the tree represented a decision-making mode of the CNN. Similarly, [38] used a semantic graph to summarize and explain all part knowledge hidden inside conv-layers of a CNN.

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, binary classification of a single class (*i.e.* $c = 1$ for images of a category and $c = 2$ for random images), and other tasks.

In the following paragraphs, we focus on the learning of a single filter f in a conv-layer. Fig. 2 shows the structure of our interpretable conv-layer. We add a loss to the feature map x of the filter f after the ReLU operation. The filter loss $Loss_f$ pushes the filter f to represent a specific object part of the category c and keep silent on images of other categories. Please see Section 3.2 for the determination of the category c for the filter f. Let $\mathbf{X} = \{x | x = f(I) \in \mathbb{R}^{n \times n}, I \in \mathbf{I}\}$ denote a set of feature maps of f after an ReLU operation *w.r.t.* different images. Given an input image $I \in \mathbf{I}_c$, the feature map x is an $n \times n$ matrix, $x_{ij} \geq 0$. If the target part appears, we expect the feature map $x = f(I)$ to exclusively activate at the target part's location; otherwise, the feature map should keep inactivated.

Therefore, a high interpretability of the filter f requires a high mutual information between the feature map $x = f(I)$ and the part location, *i.e.* the part location can roughly determine activations on the feature map x.

Accordingly, we formulate the filter loss as the

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.

minus mutual information, as follows.

$$\mathbf{Loss}_f = -MI(\mathbf{X};\mathbf{\Omega}) = -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\sum_x p(x|\mu)\log\frac{p(x|\mu)}{p(x)} \tag{1}$$

where $MI(\cdot)$ denotes the mutual information; $\mathbf{\Omega} = \{\mu_1, \mu_2, \ldots, \mu_{n^2}\} \cup \{\mu^-\}$. We use $\mu_1, \mu_2, \ldots, \mu_{n^2}$ to denote the n^2 neural units on the feature map x, each $\mu = [i, j] \in \mathbf{\Omega}$, $1 \le i, j \le n$, corresponding to a location candidate for the target part. μ^- denotes a dummy location for the case when the target part does not appear on the image.

- $p(\mu)$ measures the probability of the target part appearing at the location μ. If annotations of part locations are given, then the computation of $p(\mu)$ is simple. People can manually assign a semantic part with the filter f, and then $p(\mu)$ can be determined using part annotations.

However, in our study, the target part of filter f is not pre-defined before the learning process. Instead, the part corresponding to f needs to be determined in an online manner during the learning process. More crucially, we do not have any ground-truth annotations of the target part, which boosts the difficulty of calculating $p(\mu)$. The computation of $p(\mu)$ will be introduced later.

- The conditional likelihood $p(x|\mu)$ measures the fitness between a feature map x and the part location $\mu \in \mathbf{\Omega}$. In order to simplify the computation of $p(x|\mu)$, 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. T_{μ_i} describes the ideal distribution of activations for the feature map x when the target part mainly triggers the i-th unit in x. In addition, we also design a negative template T^- corresponding to the dummy location μ^-. The feature map can match to T^-, when the target part does not appear on the input image.

In this way, we approximately define $p(x|\mu)$ as follows.

$$p(x|\mu) \approx p(x|T_\mu) = \frac{1}{Z_\mu}\exp\left[tr(x \cdot T_\mu)\right] \tag{2}$$

where $Z_\mu = \sum_{x\in\mathbf{X}} \exp[tr(x \cdot T_\mu)]$. $tr(\cdot)$ indicates the trace of a matrix, and $tr(x \cdot T_\mu) = \sum_{ij} x_{ij} t_{ij}$. $p(x) = \sum_\mu p(\mu)p(x|\mu)$.

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.

3.1 Part localization & the mask layer

Given an input image I, the filter f computes a feature map x after the ReLU operation. Without ground-truth annotations of the target part for f, in this study, we determine the part location on x in an online manner during the learning process. We consider the neural unit with the strongest activation $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_{ij}$, $1 \le i, j \le n$ as the target part location.

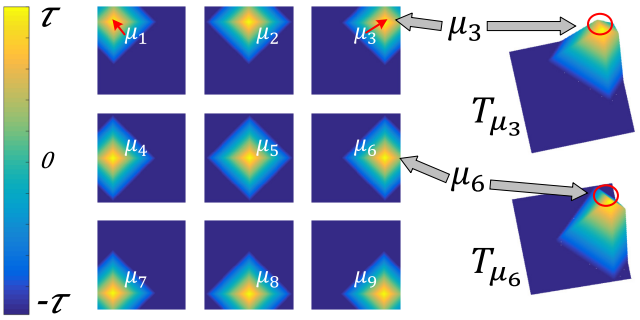


Fig. 3. Templates of T_{μ_i}. We show a toy example of $n = 3$. Each template T_{μ_i} matches to a feature map x when the target part mainly triggers the i-th unit in x. In fact, the algorithm also supports a round template based on the L-2 norm distance. Here, we use the L-1 norm distance instead to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

As shown in Fig. 2, we add a mask layer above the interpretable conv-layer. Based on the estimated part position $\hat{\mu}$, the mask layer assigns a specific mask with x to filter out noisy activations. Our method selects the template $T_{\hat{\mu}}$ *w.r.t.* the part location $\hat{\mu}$ 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. The mask operation supports gradient back-propagations.

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.

Note that during the forward propagation, our method omits the negative template T^-. Our method only selects masks from the n^2 templates $\{T_{\mu_i}\}$, no matter whether the input image contains the target part or not.

3.2 Learning

We train the interpretable CNN in 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.* 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:

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

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 gradients *w.r.t.* parameters in lower layers to update the CNN.

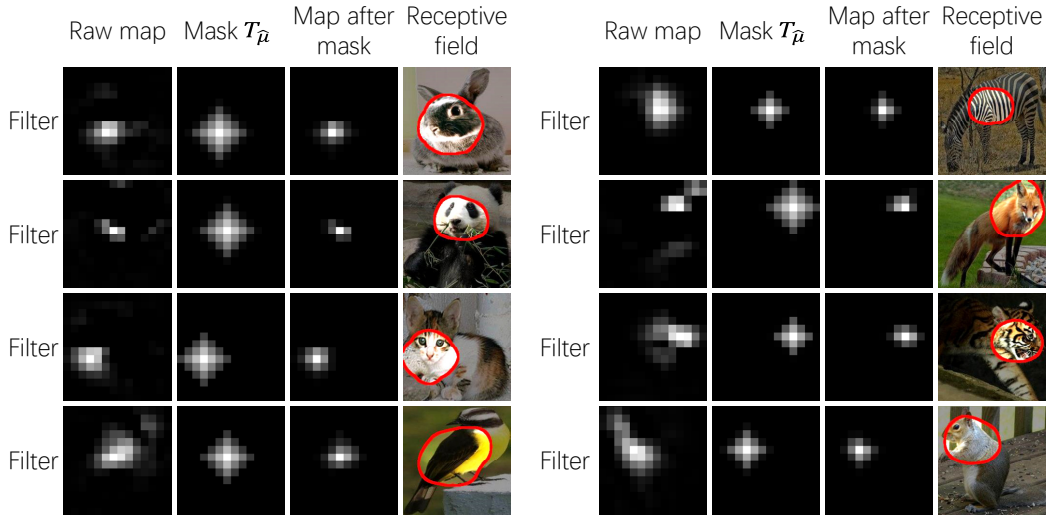


Fig. 4. Given an input image I, from left to right, we consequently show the feature map of a filter after the ReLU layer x, the assigned mask $T_{\hat{\mu}}$, the masked feature map x^{masked}, and the image-resolution RF of activations in x^{masked} computed by [46].

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

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

where $T_{\hat{\mu}}$ is the target template for feature map x. If the input image I belongs to the target category of filter f, then $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_{ij}$. If image I belongs to other categories, then $\hat{\mu} = \mu^-$.

Considering $\forall \mu \in \mathbf{\Omega} \setminus \{\hat{\mu}\}$, $e^{tr(x\cdot T_{\hat{\mu}})} \gg e^{tr(x\cdot T_\mu)}$ and $p(\hat{\mu}) \gg p(\mu)$ after initial learning episodes, we make the above approximation to simplify the computation. Because $Z_{\hat{\mu}}$ is computed using numerous feature maps, we can roughly treat $Z_{\hat{\mu}}$ as a constant to compute gradients in the above equation. We gradually update the value of $Z_{\hat{\mu}}$ 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 Equation (4). We simply assign the filter f with the category $\hat{c}$ whose images activate f the most, *i.e.* $\hat{c} = \text{argmax}_c \mathbb{E}_{x=f(I):I\in\mathbf{I}_c} \sum_{ij} x_{ij}$.

2. We can use a subset of feature maps to approximate the value of Z_μ, and continue to update Z_μ when we receive more feature maps during the training process. Similarly, we can approximate $p(x)$ using a subset of feature maps. We compute $p(x) = \sum_\mu p(\mu)p(x|\mu) = \sum_\mu p(\mu)\frac{\exp[tr(x\cdot T_\mu)]}{Z_\mu} \approx \sum_\mu p(\mu)\mathbb{E}_x\frac{\exp[tr(x\cdot T_\mu)]}{Z_\mu}$.

The proof of Equation (4) is given as follows.

$$\begin{aligned}
\frac{\partial \mathbf{Loss}}{\partial x_{ij}} &= -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\Big\{\frac{\partial p(x|\mu)}{\partial x_{ij}}\left[\log p(x|\mu) - \log p(x) + 1\right] \\
&\quad - p(x|\mu)\frac{\partial \log p(x)}{\partial x_{ij}}\Big\} \\
&= -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\Big\{\frac{\partial p(x|\mu)}{\partial x_{ij}}\left[\log p(x|\mu) - \log p(x) + 1\right] \\
&\quad - p(x|\mu)\frac{1}{p(x)}\frac{\partial p(x)}{\partial x_{ij}}\Big\} \\
&= -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\Big\{\frac{\partial p(x|\mu)}{\partial x_{ij}}\left[\log p(x|\mu) - \log p(x) + 1\right] \\
&\quad - p(x|\mu)\frac{1}{p(x)}\sum_{\mu'}\Big[p(\mu')\frac{\partial p(x|\mu')}{\partial x_{ij}}\Big]\Big\} \\
&= -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\Big\{\frac{\partial p(x|\mu)}{\partial x_{ij}}\left[\log p(x|\mu) - \log p(x) + 1\right]\Big\} \\
&\quad + \sum_{\mu\in\mathbf{\Omega}} p(\mu)\frac{\partial p(x|\mu)}{\partial x_{ij}}\frac{\sum_{\mu'} p(\mu')p(x|\mu')}{p(x)} \\
&= -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\Big\{\frac{\partial p(x|\mu)}{\partial x_{ij}}\left[\log p(x|\mu) - \log p(x) + 1\right]\Big\} \\
&\quad + \sum_{\mu\in\mathbf{\Omega}} p(\mu)\frac{\partial p(x|\mu)}{\partial x_{ij}} \\
&= -\sum_{\mu\in\mathbf{\Omega}} \frac{\partial p(x|\mu)}{\partial x_{ij}} p(\mu)\left[\log p(x|\mu) - \log p(x)\right] \\
&= -\sum_{\mu\in\mathbf{\Omega}} \frac{t_{ij}p(\mu)e^{tr(x\cdot T)}}{Z_\mu}\Big\{tr(x\cdot T) - \log\left[Z_\mu p(x)\right]\Big\}
\end{aligned} \quad (5)$$

3.3 Understanding the filter loss

The filter loss in Equation (1) can be re-written as

$$\mathbf{Loss}_f = -H(\mathbf{\Omega}) + H(\mathbf{\Omega}'|\mathbf{X}) + \sum_x p(\mathbf{\Omega}^+, x) H(\mathbf{\Omega}^+|X=x) \tag{6}$$

Where $\mathbf{\Omega}' = \{\mu^-, \mathbf{\Omega}^+\}$. $H(\mathbf{\Omega}) = -\sum_{\mu\in\mathbf{\Omega}} p(\mu)\log p(\mu)$ is a constant prior entropy of part locations.

Equation (6) is proved as follows.

$$\begin{aligned}
\mathbf{Loss} &= -MI(\mathbf{X};\mathbf{\Omega}) \qquad //\ \mathbf{\Omega} = \{\mu^-, \mu_1, \mu_2, \ldots, \mu_{n^2}\} \\
&= -H(\mathbf{\Omega}) + H(\mathbf{\Omega}|\mathbf{X}) \\
&= -H(\mathbf{\Omega}) - \sum_x p(x) \sum_{\mu\in\mathbf{\Omega}} p(\mu|x)\log p(\mu|x) \\
&= -H(\mathbf{\Omega}) - \sum_x p(x)\Big\{p(\mu^-|x)\log p(\mu^-|x) \\
&\quad + \sum_{\mu\in\mathbf{\Omega}^+} p(\mu|x)\log p(\mu|x)\Big\} \\
&= -H(\mathbf{\Omega}) - \sum_x p(x)\Big\{p(\mu^-|x)\log p(\mu^-|x) \\
&\quad + \sum_{\mu\in\mathbf{\Omega}^+} p(\mu|x)\log\big[\frac{p(\mu|x)}{p(\mathbf{\Omega}^+|x)} p(\mathbf{\Omega}^+|x)\big]\Big\} \\
&= -H(\mathbf{\Omega}) - \sum_x p(x)\Big\{p(\mu^-|x)\log p(\mu^-|x) \\
&\quad + p(\mathbf{\Omega}^+|x)\log p(\mathbf{\Omega}^+|x) + \sum_{\mu\in\mathbf{\Omega}^+} p(\mu|x)\log\frac{p(\mu|x)}{p(\mathbf{\Omega}^+|x)}\Big\} \\
&= -H(\mathbf{\Omega}) + H(\mathbf{\Omega}' = \{\mu^-, \mathbf{\Omega}^+\}|\mathbf{X}) \\
&\quad + \sum_x p(\mathbf{\Omega}^+, x) H(\mathbf{\Omega}^+|X=x)
\end{aligned} \tag{7}$$

where $p(\mathbf{\Omega}^+|x) = \sum_{\mu\in\mathbf{\Omega}^+} p(\mu|x)$, $H(\mathbf{\Omega}^+|X=x) = \sum_{\mu\in\mathbf{\Omega}^+} \tilde{p}(\mu|X=x)\log\tilde{p}(\mu|X=x)$, $\tilde{p}(\mu|X=x) = \frac{p(\mu|x)}{p(\mathbf{\Omega}^+|x)}$.

●**Low inter-category entropy:** The second term $H(\mathbf{\Omega}' = \{\mu^-, \mathbf{\Omega}^+\}|\mathbf{X})$ is computed as

$$H(\mathbf{\Omega}' = \{\mu^-, \mathbf{\Omega}^+\}|\mathbf{X}) = -\sum_x p(x) \sum_{\mu\in\{\mu^-, \mathbf{\Omega}^+\}} p(\mu|x)\log p(\mu|x) \tag{8}$$

where $\mathbf{\Omega}^+ = \{\mu_1, \ldots, \mu_{n^2}\} \subset \mathbf{\Omega}$, $p(\mathbf{\Omega}^+|x) = \sum_{\mu\in\mathbf{\Omega}^+} p(\mu|x)$. We define the set of all real locations $\mathbf{\Omega}^+$ as a single label to represent category c. We use the dummy location μ^- to roughly indicate matches to 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 or not the input image belongs to category c, *i.e.* x fitting to either $T_{\hat{\mu}}$ or T^-, without significant uncertainty.

●**Low spatial entropy:** The third term in Equation (6) is given as

$$H(\mathbf{\Omega}^+|X=x) = \sum_{\mu\in\mathbf{\Omega}^+} \tilde{p}(\mu|x)\log\tilde{p}(\mu|x) \tag{9}$$

where $\tilde{p}(\mu|x) = \frac{p(\mu|x)}{p(\mathbf{\Omega}^+|x)}$. This term encourages a low conditional entropy of the spatial distribution of x's activations. *I.e.* given an image $I \in \mathbf{I}_c$, a well-learned filter should only be activated in a single region $\hat{\mu}$ of the feature map x, instead of being repetitively triggered at different locations.

4 Experiments

In experiments, we applied our method to modify four types of CNNs with various structures into interpretable CNNs and learned interpretable CNNs based on three benchmark datasets, in order to demonstrate the broad applicability. We learned interpretable CNNs for binary classification of a single category and multi-category classification. We used different techniques to visualize the knowledge encoded in interpretable filters, in order to qualitatively illustrate semantic meanings of these filters. Furthermore, we used two types of evaluation metrics, *i.e.* the object-part interpretability and the location instability, to measure the clarity of the meaning of a filter.

Our experiments showed that an interpretable filter in our interpretable CNN usually consistently represented the same part through different input images, while a filter in an ordinary CNN mainly described a mixture of semantics.

We chose three benchmark datasets with part annotations for training and testing, including the ILSVRC 2013 DET Animal-Part dataset [39], the CUB200-2011 dataset [33], and the VOC Part dataset [3]. These datasets provide ground-truth bounding boxes of entire objects. For landmark annotations, the ILSVRC 2013 DET Animal-Part dataset [39] contains ground-truth bounding boxes of heads and legs of 30 animal categories. The CUB200-2011 dataset [33] contains a total of 11.8K bird images of 200 species, and the dataset provides center positions of 15 bird landmarks. The VOC Part dataset [3] contains ground-truth part segmentations of 107 object landmarks in six animal categories.

We used these datasets, because they contain ground-truth annotations of object landmarks[3] (parts) to evaluate the semantic clarity of each filter. As mentioned in [3], [39], animals usually consist of non-rigid parts, which present considerable challenges for part localization. As in [3], [39], we selected animal categories in the three datasets for testing.

We learned interpretable filters based on structures of four typical CNNs for evaluation, including the AlexNet [12], the VGG-M [27], the VGG-S [27], the VGG-16 [27]. Note that skip connections in residual networks [7] make a single feature map contain patterns of different filters. Thus, we did not use residual networks for testing to simplify the evaluation. Given a CNN, all filters in the top conv-layer were set as

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.

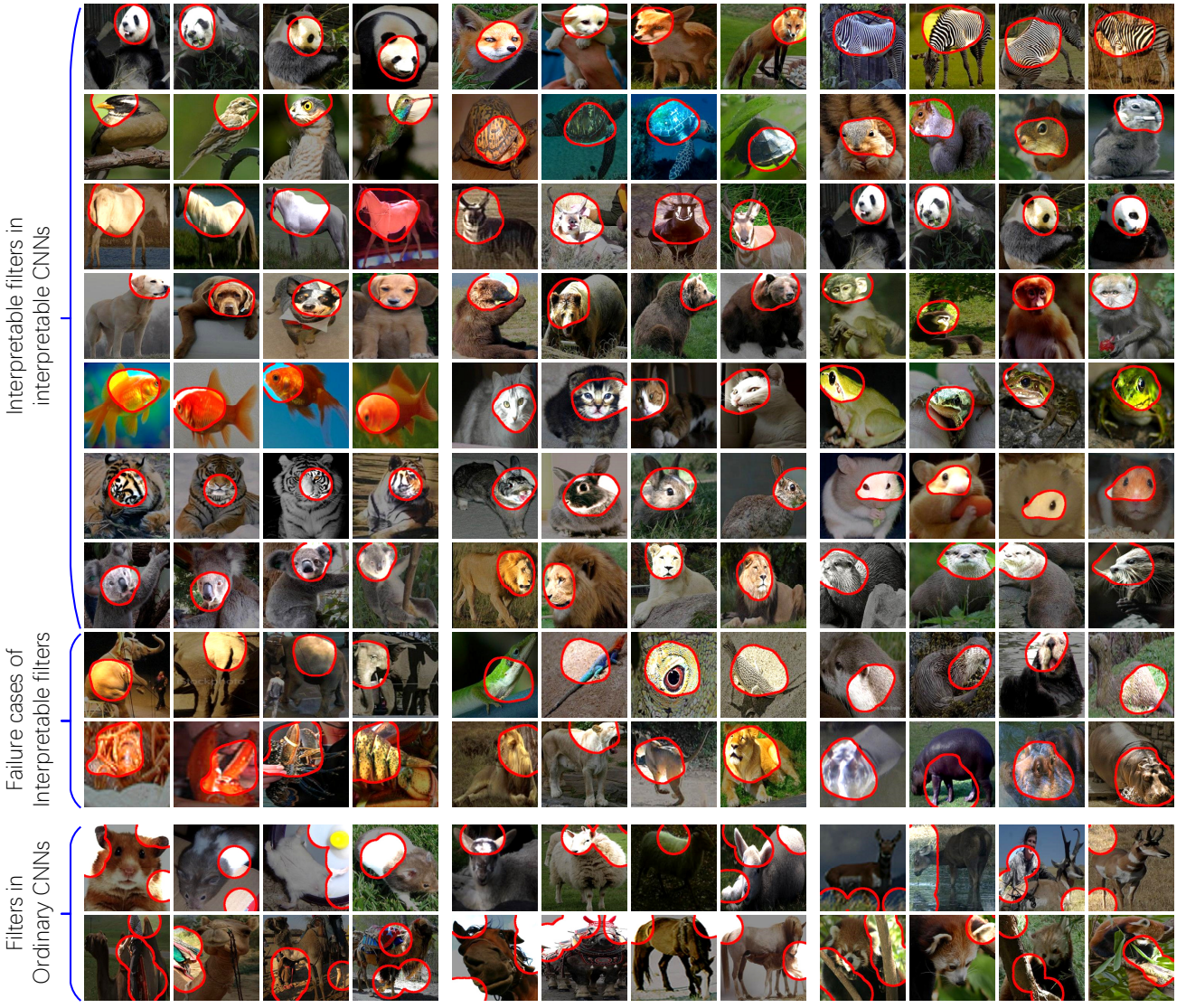


Fig. 5. Visualization of filters in top conv-layers. We used [46] to estimate the image-resolution receptive field of activations in a feature map to visualize a filter's semantics. The top four rows visualize filters in interpretable CNNs, and the bottom two rows correspond to filters in ordinary CNNs.

interpretable filters. Then, we inserted another conv-layer with M filters above the top conv-layer, which did not change the size of output feature maps. Filters in the new conv-layer were also interpretable filters. Each filter was a $3 \times 3 \times M$ tensor with a bias term.

Implementation details: We set parameters as $\tau = \frac{0.5}{n^2}$. $\beta \approx 4$. β was updated in an online manner. We set a decreasing weight for filter losses, *i.e.* $\lambda \propto \frac{1}{t}\mathbb{E}_{x \in \mathbf{X}} \max_{i,j} x_{ij}$ for the t-th epoch. We initialized fully-connected (FC) layers and the new conv-layer, but we loaded parameters of the lower conv-layers from a CNN that was pre-trained using [12], [27]. We then fine-tuned parameters of all layers in the interpretable CNN using training images in the dataset. To enable a fair comparison, when we learned the traditional CNN as a baseline, we also initialized FC layers of the traditional CNN, used pre-trained parameters in conv-layers, and then fine-tuned the CNN.

4.1 Experiments

Binary classification of a single category: We learned interpretable CNNs based on above four types of network structures to classify each animal category in above three datasets. We also learned ordinary CNNs using the same data for comparison. We used the logistic log loss for binary classification of a single category from random images. We followed experimental settings in [38], [39] to crop objects of the target category as positive samples. Images of other categories were regarded as negative samples.

Multi-category classification: We learned interpretable CNNs to classify the six animal categories in the VOC Part dataset [3] and also learned interpretable CNNs

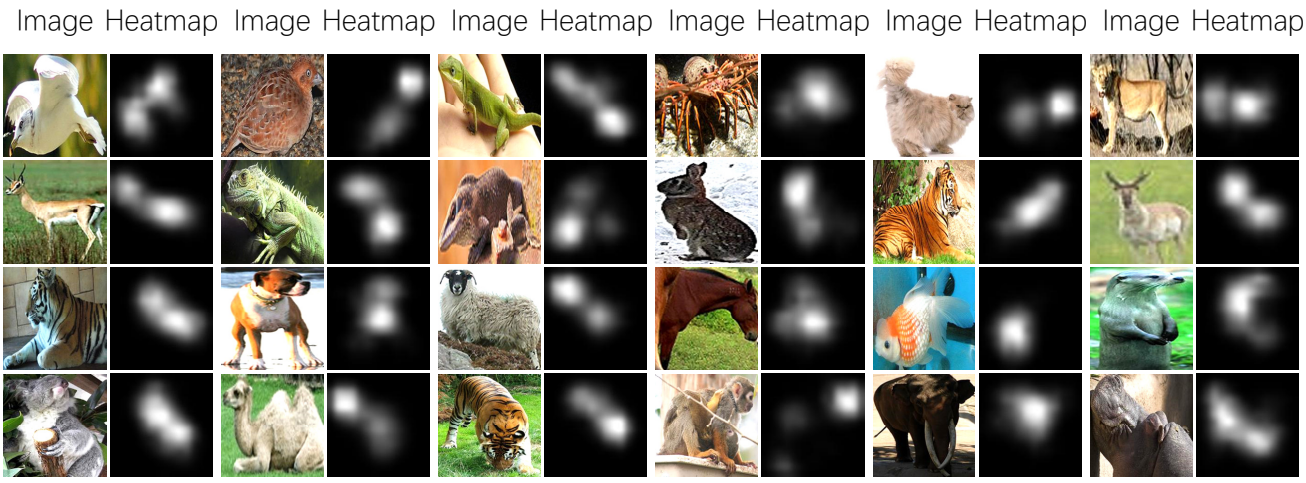


Fig. 6. Heatmaps for distributions of object parts that are encoded in interpretable filters. We use all filters in the top conv-layer to compute the heatmap. Interpretable filters usually selectively modeled distinct object parts of a category and ignored other parts.

to classify the thirty categories in the ILSVRC 2013 DET Animal-Part dataset [39]. In experiments, we tried both the softmax log loss and the logistic log loss[4] for multi-category classification.

4.2 Qualitative Visualization of filters

We followed the method proposed by Zhou *et al.* [46] to compute the receptive fields (RFs) of neural activations of a filter. We used neural activations after ReLU and mask operations and scaled up RFs to the image resolution. As discussed in [2], the traditional idea of directly propagating the theoretical receptive field of a neural unit in a feature map back to the image plane cannot accurately reflect the real image-resolution RF of the neural unit (*i.e.* the image region that contributes most to the score of the neural unit). Therefore, we used the method of [46] to compute real RFs.

Studies in both [46] and [2] have introduced methods to compute real RFs of neural activations on a given feature map. [46] accurately computes the RF when the filter represents an object part. However, when a filter in an ordinary CNN represents textures without consistent contours, it is difficult for [46] to compute RFs, because this method needs to align activation regions through different images. Therefore, for ordinary CNNs, we simply used a round RF for each neural activation. We overlapped all activated RFs in a feature map to compute the final RF of the feature map.

Fig. 5 shows RFs[5] of filters in top conv-layers of CNNs, which were trained for binary classification of a single category. Filters in interpretable CNNs were mainly activated by a certain object part, whereas feature maps of ordinary CNNs after ReLU operations usually represented various object parts and textures.

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. We can understand such results from the perspective of the information bottleneck [35] as follows. (i) Our interpretable filters selectively encode the most distinct parts of each category (*i.e.* the head for most categories), which minimizes the conditional entropy of the final classification given feature maps of a conv-layer. (ii) Each interpretable filter represents a specific part of an object, which minimizes the mutual information between the input image and middle-layer feature maps. The interpretable CNN "forgets" as much irrelevant information as possible.

In addition to the visualization of RFs, we also visualized heatmaps for part distributions and the grad-CAM attention map of an interpretable conv-layer. Fig. 6 shows heatmaps for distributions of object parts that were encoded in interpretable filters. Fig. 7 compares grad-CAM visualizations [23] of an interpretable conv-layer and those of a traditional conv-layer. We chose the top conv-layer of the traditional VGG-16 net and the top conv-layer of the interpretable VGG-16 net for visualization. Interpretable filters usually selectively modeled distinct object parts of a category and ignored other parts.

4.3 Quantitative evaluation of part interpretability

Filters in low conv-layers usually represent simple patterns or object details, whereas those in high conv-layers are more likely to describe large-scale parts. Therefore, in experiments, we used the following two

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

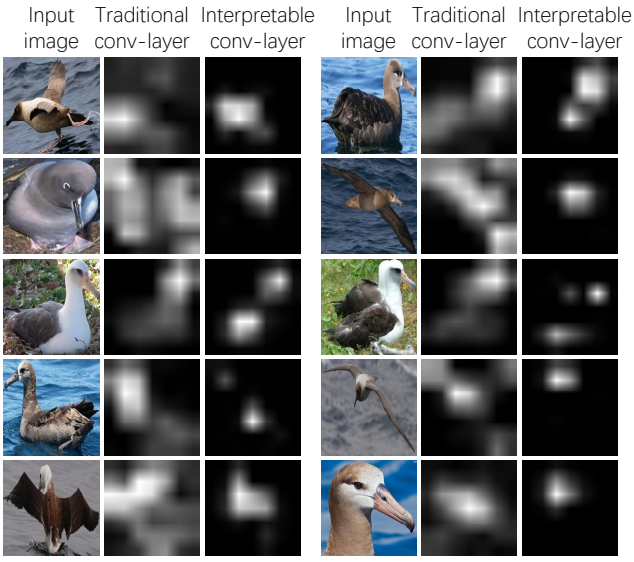


Fig. 7. Grad-CAM visualizations [23] of the traditional conv-layer and the interpretable conv-layer. Unlike the traditional conv-layer, the interpretable conv-layer usually selectively modeled distinct object parts of a category and ignored other parts.

metrics to evaluate the clarity of part semantics of the top conv-layer of a CNN.

4.3.1 Evaluation metric: part interpretability

The metric was originally proposed by Bau *et al.* [2] to measure the object-part interpretability of filters. For each filter f, $\mathbf{X}$ denotes a set of feature maps after ReLU/mask operations on different input images. Then, the distribution of activation scores over all positions in all feature maps was computed. [2] set a threshold T_f such that $p(x_{ij} > T_f) = 0.005$ to select strongest activations from all positions $[i, j]$ from $x \in \mathbf{X}$ as valid activations for f's semantics.

Then, image-resolution RFs of valid neural activations of each input image I were computed[5]. The RFs on image I, termed S_f^I, corresponded to part regions of f.

The fitness between the filter f and the k-th part on image I was reported as the 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 represents the ground-truth mask of the k-th part on image I. Given an image I, the filter f was associated with the k-th part if $IoU_{f,k}^I > 0.2$. The criterion $IoU_{f,k}^I > 0.2$ was stricter than $IoU_{f,k}^I > 0.04$ in [2], because object-part semantics usually needs a stricter criterion than textural semantics and color semantics in [2]. The average probability of the k-th part being associating with the filter f was reported as $P_{f,k} = \mathbb{E}_{I:\text{with k-th part}} \mathbf{1}(IoU_{f,k}^I > 0.2)$. Note that a single filter may be associated with multiple object parts in an image. The highest probability of part association for each filter was used as the interpretability of filter f, *i.e.* $P_f = \max_k P_{f,k}$.

For the binary classification of a single category, we used testing images of the target category to evaluate the feature interpretability. In the VOC Part dataset [3], four parts were chosen for the *bird* category. We merged segments of the head, beak, and l/r-eyes as the head part, merged segments of the torso, neck, and l/r-wings as the torso part, merged segments of l/r-legs/feet as the leg part, and used the tail segment as the fourth part. We used five parts for both the *cat* category and the *dog* category. We merged segments of the head, l/r-eyes, l/r-ears, and nose as the head part, merged segments of the torso and neck as the torso part, merged segments of frontal l/r-legs/paws as the frontal legs, merged segments of back l/r-legs/paws as the back legs, and used the tail as the fifth part. Part definitions for the *cow*, *horse*, and *sheep* category were similar those for the cat category, except for that we omitted the tail part of these categories. In particular, we added l/r-horn segments of the horse to the head part. The average part interpretability P_f over all filters was computed for evaluation.

For the multi-category classification, we first determined the target category $\hat{c}$ for each filter f *i.e.* $\hat{c} = \text{argmax}_c \mathbb{E}_{x=f(I):I\in\mathbf{I}_c} \sum_{i,j} x_{ij}$. Then, we computed f's object-part interpretability using images of the target category $\hat{c}$ by following above instructions.

4.3.2 Evaluation metric: location instability

The second metric measures the instability of part locations, which was used in [38], [43]. It is assumed that if f consistently represented the same object part through different objects, then distances between the inferred part $\hat{\mu}$ and some ground-truth landmarks[3] should keep stable among different objects. For example, if f represented the shoulder part without ambiguity, then the distance between the inferred position and the head will not change a lot among different objects.

Therefore, the deviation of the distance between the inferred position $\hat{\mu}$ and a specific ground-truth landmark among different images was computed. The location $\hat{\mu}$ was inferred as the neural unit with the highest activation on f's feature map. We reported the average deviation *w.r.t.* different landmarks as the location instability of f.

Please see Fig. 8. Given an input image I, $d_I(p_k, \hat{\mu}) = \frac{\|\mathbf{p}_k - \mathbf{p}(\hat{\mu})\|}{\sqrt{w^2+h^2}}$ denotes the normalized distance between the inferred part and the k-th landmark $\mathbf{p}_k$, where $\mathbf{p}(\hat{\mu})$ is referred to as the center of the unit

5. [46] computes the RF when the filter represents an object part. Fig. 5 used RFs computed by [46] to visualize filters. However, when a filter in an ordinary CNN does not have consistent contours, it is difficult for [46] 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 4.3.1, we uniformly applied these RFs to both interpretable CNNs and ordinary CNNs.

	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 binary classification of a single category based on the VOC Part dataset [3].

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

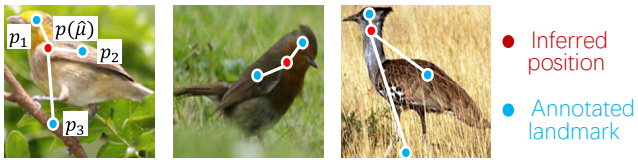


Fig. 8. Notation for computing the location instability.

$\hat{\mu}$'s RF. $\sqrt{w^2+h^2}$ measures the diagonal length of I. $D_{f,k} = \sqrt{\text{var}_I[d_I(p_k, \hat{\mu})]}$ is termed 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 the variation of $d_I(p_k, \hat{\mu})$. Because each landmark could not appear in all testing images, for each filter f, the metric only used inference results on top-ranked 100 images with the highest inference scores to compute $D_{f,k}$. In this way, the average of relative location deviations of all the filters in a conv-layer *w.r.t.* all K landmarks, *i.e.* $\mathbb{E}_f \mathbb{E}_{k=1}^{K} D_{f,k}$, was reported as the location instability of f.

We used the most frequent object parts as landmarks to measure the location instability. For the ILSVRC 2013 DET Animal-Part dataset [39], we used the *head* and *frontal legs* of each category as landmarks for evaluation. For the VOC Part dataset [3], we selected the *head*, *neck*, and *torso* of each category as landmarks. For the CUB200-2011 dataset [33], we used the *head*, *back*, *tail* of birds as landmarks.

In particular, for multi-category classification, we first determined the target category of for each filter f and then computed the relative location deviation $D_{f,k}$ using landmarks of f's target category. Because filters in baseline 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.* for a baseline CNN, we used $\mathbb{E}_f \min_c \mathbb{E}_{k \in Part_c} D_{f,k}$ to evaluate the location instability, where $Part_c$ denotes the set of part indexes belonging to category c.

4.3.3 Comparisons between metrics of filter interpretability and location instability

Although the filter interpretability [2] and the location instability [43] are the two most state-of-the-art metrics to evaluate the interpretability of a convolution filter, these metrics still have some limitations.

Firstly, the filter interpretability [2] assumes that the feature map of an automatically learned filter should well match the ground-truth segment of a semantic part (with an explicit part name), an object, or a texture. For example, it assumes that a filter may represent the exact segment of the head part. However, without ground-truth annotations of object parts or textures for supervision, there is no mechanism to assign explicit semantic meanings with filters during the learning process. In most cases, filters in an interpretable CNN (as well as a few filters in traditional CNNs) may describe a specific object part without explicit names, *e.g.* the region of both the head and neck or the region connecting the torso and the tail. Therefore, in both [2] and [43], people did not require the inferred object region to describe the exact segment of a semantic part, and simply set a relatively loose criterion $IoU_{f,k}^{I} > 0.04$ or 0.2 to compute the filter interpretability.

Secondly, the location instability was proposed in [43]. The location instability of a filter is evaluated using the average deviation of distances between the inferred position and some ground-truth landmarks. There is also an assumption for this evaluation metric, *i.e.* the distance between an inferred part and a specific landmark should not change a lot through different images. As a result, people cannot set landmarks as

	gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion
AlexNet	0.161	0.167	0.152	0.153	0.175	0.128	0.123	0.144	0.143	0.148	0.137
AlexNet+ordinary layer	0.154	0.157	0.143	0.146	0.170	0.120	0.118	0.127	0.117	0.136	0.120
AlexNet, interpretable	**0.084**	**0.095**	**0.090**	**0.107**	**0.097**	**0.079**	**0.077**	**0.093**	**0.087**	**0.095**	**0.084**
VGG-16	0.153	0.156	0.144	0.150	0.170	0.127	0.126	0.143	0.137	0.148	0.139
VGG-16+ordinary layer	0.136	0.127	0.120	0.136	0.147	0.108	0.111	0.111	0.097	0.134	0.102
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**
VGG-M	0.161	0.166	0.151	0.153	0.176	0.128	0.125	0.145	0.145	0.150	0.140
VGG-M+ordinary layer	0.147	0.144	0.135	0.142	0.159	0.114	0.115	0.119	0.111	0.128	0.114
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**
VGG-S	0.158	0.166	0.149	0.151	0.173	0.127	0.124	0.143	0.142	0.148	0.138
VGG-S+ordinary layer	0.150	0.132	0.133	0.138	0.156	0.113	0.111	0.110	0.104	0.125	0.112
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**

	tiger	bear	rabb.	hams.	squi.	horse	zebra	swine	hippo.	catt.	sheep
AlexNet	0.142	0.144	0.148	0.128	0.149	0.152	0.154	0.141	0.141	0.144	0.155
AlexNet+ordinary layer	0.123	0.133	0.136	0.112	0.145	0.149	0.142	0.137	0.139	0.141	0.149
AlexNet, interpretable	**0.090**	**0.095**	**0.095**	**0.077**	**0.095**	**0.098**	**0.084**	**0.091**	**0.089**	**0.097**	**0.101**
VGG-16	0.144	0.143	0.146	0.125	0.150	0.150	0.153	0.141	0.140	0.140	0.150
VGG-16+ordinary layer	0.127	0.112	0.119	0.100	0.112	0.134	0.140	0.126	0.126	0.131	0.135
VGG-16, interpretable	**0.097**	**0.081**	**0.079**	**0.066**	**0.065**	**0.106**	**0.077**	**0.094**	**0.083**	**0.102**	**0.097**
VGG-M	0.145	0.144	0.150	0.128	0.150	0.151	0.158	0.140	0.140	0.143	0.155
VGG-M+ordinary layer	0.124	0.131	0.134	0.108	0.132	0.138	0.141	0.133	0.131	0.135	0.142
VGG-M, interpretable	**0.089**	**0.101**	**0.097**	**0.082**	**0.095**	**0.095**	**0.080**	**0.095**	**0.084**	**0.092**	**0.094**
VGG-S	0.142	0.143	0.148	0.128	0.146	0.149	0.155	0.139	0.140	0.141	0.155
VGG-S+ordinary layer	0.117	0.127	0.127	0.105	0.122	0.136	0.137	0.133	0.131	0.130	0.143
VGG-S, interpretable	**0.089**	**0.097**	**0.091**	**0.076**	**0.098**	**0.096**	**0.080**	**0.092**	**0.088**	**0.094**	**0.101**

	ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.			**Avg.**
AlexNet	0.147	0.153	0.159	0.160	0.139	0.125	0.140	0.125			0.146
AlexNet+ordinary layer	0.148	0.143	0.145	0.151	0.125	0.116	0.127	0.102			0.136
AlexNet, interpretable	**0.085**	**0.102**	**0.104**	**0.095**	**0.090**	**0.085**	**0.084**	**0.073**			**0.091**
VGG-16	0.144	0.149	0.154	0.163	0.136	0.129	0.143	0.125			0.144
VGG-16+ordinary layer	0.122	0.121	0.134	0.143	0.108	0.110	0.115	0.102			0.121
VGG-16, interpretable	**0.091**	**0.105**	**0.093**	**0.100**	**0.074**	**0.084**	**0.067**	**0.063**			**0.085**
VGG-M	0.146	0.154	0.160	0.161	0.140	0.126	0.142	0.127			0.147
VGG-M+ordinary layer	0.130	0.135	0.140	0.150	0.120	0.112	0.120	0.106			0.130
VGG-M, interpretable	**0.077**	**0.104**	**0.102**	**0.093**	**0.086**	**0.087**	**0.089**	**0.068**			**0.090**
VGG-S	0.143	0.154	0.158	0.157	0.140	0.125	0.139	0.125			0.145
VGG-S+ordinary layer	0.125	0.133	0.135	0.147	0.119	0.111	0.118	0.100			0.126
VGG-S, interpretable	**0.077**	**0.102**	**0.105**	**0.094**	**0.090**	**0.086**	**0.078**	**0.072**			**0.090**

TABLE 3
Location instability of filters ($\mathbb{E}_{f,k}[D_{f,k}]$) in CNNs that are trained for the binary classification of a single category using the ILSVRC 2013 DET Animal-Part dataset [39]. Filters in our interpretable CNNs exhibited significantly lower localization instability than ordinary CNNs.

the head and the tail of a snake, because the distance between different parts of a snake continuously changes when the snake moves.

Generally speaking, there are two advantages to use the location instability for evaluation:

- The computation of the location instability [43] is independent to the size of the receptive field (RF) of a neural activation. This solves a big problem with the evaluation of filter interpretability, *i.e.* state-of-the-art methods of computing a neural activation's image-resolution RFs (*e.g.* [46]) can only provide an approximate scale of the RF. The metric of location instability only uses central positions of part inferences of a filter, rather than use the entire inferred part segment, for evaluation. Thus, the location instability is a robust metric to evaluate the object-part interpretability of a filter.
- The location instability allows a filter to represent an object part without an explicit name (a half of the head).

Nevertheless, the evaluation metric for filter interpretability is still an open problem.

	bird	cat	cow	dog	horse	sheep	**Avg.**
AlexNet	0.153	0.131	0.141	0.128	0.145	0.140	0.140
AlexNet+ordinary layer	0.147	0.125	0.139	0.112	0.146	0.143	0.136
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+ordinary layer	0.125	0.121	0.137	0.102	0.131	0.137	0.125
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+ordinary layer	0.142	0.120	0.139	0.115	0.141	0.142	0.133
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+ordinary layer	0.137	0.115	0.133	0.107	0.133	0.138	0.127
VGG-S, interpretable	**0.089**	**0.092**	**0.092**	**0.087**	**0.086**	**0.088**	**0.089**

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

Neural network	Avg. location instability
AlexNet	0.150
AlexNet+ordinary layer	0.118
AlexNet, interpretable	**0.070**
VGG-16	0.137
VGG-16+ordinary layer	0.097
VGG-16, interpretable	**0.076**
VGG-M	0.148
VGG-M+ordinary layer	0.107
VGG-M, interpretable	**0.065**
VGG-S	0.148
VGG-S+ordinary layer	0.103
VGG-S, interpretable	**0.073**

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

	ILSVRC Part [39]	VOC Part [3]	
	Logistic log loss[4]	Logistic log loss[4]	Softmax log loss
VGG-16	–	0.128	0.142
ordinary layer	–	0.096	0.099
interpretable	–	**0.073**	**0.075**
VGG-M	0.167	0.135	0.137
ordinary layer	–	0.117	0.107
interpretable	**0.096**	**0.083**	**0.087**
VGG-S	0.131	0.138	0.138
ordinary layer	–	0.127	0.099
interpretable	**0.083**	**0.078**	**0.082**

TABLE 6
Location instability of filters ($\mathbb{E}_{f,k}[D_{f,k}]$) 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.

4.3.4 Experimental results and analysis

Feature interpretability of different CNNs is evaluated in Tables 1, 2, 3, 4, 5, and 6. Tables 1 and 2 show results based on the metric in [2]. Tables 3, 4, and 5 list location instability of CNNs for binary classification of a single category. Table 6 reports location instability of CNNs that were learned for multi-category classification.

We compared our interpretable CNNs with both original CNNs and CNNs with an additional conv-layer on the top, termed *AlexNet/VGG-16/VGG-M/VGG-S+ordinary layer*. To construct the CNN with a new conv-layer, we put a new conv-layer on the top of conv-layer. The filter size of the new conv-layer was $3 \times 3 \times channelnumber$, and output feature maps of the new conv-layer were in the same size of input feature maps. Because our interpretable CNN had an additional interpretable conv-layer, we designed the baseline CNN with a new conv-layer to enable fair comparisons. Our interpretable filters exhibited significantly higher part interpretability and lower location instability than traditional filters in baseline CNNs over almost all comparisons. Table 7 reports the classification accuracy of different CNNs. Ordinary CNNs exhibited better performance in binary classification, while interpretable CNNs outperformed baseline CNNs in multi-category classification.

In addition, to prove the discrimination power of the learned filter, we further tested the average accuracy when we used the maximum activation score in a single filters feature map as a metric for binary classification between birds in the CUB200-2011 dataset [33] and random images. In the scenario of classifying birds from random images, filters in the CNN was expected to learn the common appearance of birds, instead of summarizing knowledge from random images. Thus, we chose filters in the top conv-layer. If the maximum activation score of a filter exceeded a threshold, then we classified the input image as a bird; otherwise not. The threshold was set to the one that maximized the classification accuracy. Table 8 reports

Multi-category classification

	ILSVRC Part	VOC Part	
	logistic[4]	logistic[4]	softmax
VGG-M	96.73	93.88	81.93
interpretable	**97.99**	**96.19**	**88.03**
VGG-S	96.98	94.05	78.15
interpretable	**98.72**	**96.78**	**86.13**
VGG-16	–	97.97	89.71
interpretable	–	**98.50**	**91.60**

Binary classification of a single category

	ILSVRC Part	VOC Part	CUB200
AlexNet	**96.28**	**95.40**	**95.59**
interpretable	95.38	93.93	95.35
VGG-M	**97.34**	**96.82**	**97.34**
interpretable	95.77	94.17	96.03
VGG-S	**97.62**	**97.74**	**97.24**
interpretable	95.64	95.47	95.82
VGG-16	**98.58**	**98.66**	**98.91**
interpretable	96.67	95.39	96.51

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

	filters in ordinary CNNs	filters in interpretable CNNs
AlexNet	68.7	**75.1**
VGG-M	69.9	**80.2**
VGG-16	72.1	**82.4**

TABLE 8
Classification accuracy based on a single filter. We reported the average accuracy to demonstrate the discrimination power of individual filters.

the average classification accuracy over all filters. Our interpretable filters outperformed ordinary filters.

Given a CNN for binary classification of an animal category in the VOC Part dataset [3], we manually annotated the part name corresponding to the learned filters in the CNN. Table 9 reports the ratio of interpretable filters that corresponds to each object part.

5 Conclusion and Discussions

In this paper, we have proposed a general method to enhance feature interpretability of CNNs. We design a loss to push a filter in high conv-layers towards the representation of an object part during the learning process without any part annotations. Experiments have shown that each interpretable filter consistently represents a certain object part of a category through different input images. Whereas, a filter in traditional CNNs usually represents a mixture of parts and textures. Note that filters in a conv-layer are assigned with different categories in the scenario of multi-category classification. Thus, when we need to classify a large number of categories, theoretically, each category can only obtain a few filters, which will decrease a bit the classification performance.

	bird	cow	cat	dog	horse	sheep
head	19.2	–	–	–	15.4	–
neck	21.2	–	–	5.8	–	–
torso	36.5	32.7	3.8	38.5	75.0	52.0
hip & tail	5.8	–	–	–	–	–
foot	5.8	–	–	–	9.6	3.8
wing	11.5	–	–	–	–	–
eye	–	30.8	55.8	11.5	7.7	–
nose & mouth	–	15.4	32.7	3.8	19.2	–
side face	–	9.6	–	–	–	–
leg	–	11.5	5.8	23.1	–	–
ear & horn	–	–	1.9	17.3	17.3	–

TABLE 9
Statistics of semantic meanings of interpretable filters. "–" indicates that the part is not selected as a label to describe the filter in a CNN. Except for CNNs for the bird and the horse, CNNs for other animals paid attention to detailed structures of the head. Thus, we annotated fine-grained parts inside the head for these CNNs.

Acknowledgments

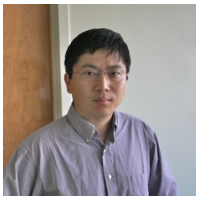
This work is supported by DARPA XAI Award N66001-17-2-4029, NSF IIS 1423305, and ARO project W911NF1810296.

References

[1] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. *In ICCV*, 2015.
[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*, 2017.
[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.
[4] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016.
[5] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In arXiv:1704.03296v1*, 2017.
[6] Y. Goyal, A. Mohapatra, D. Parikh, and D. Batra. Towards transparent ai systems: Interpreting visual question answering models. *In arXiv:1608.08974v2*, 2016.
[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2016.
[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.
[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.
[10] P. Koh and P. Liang. Understanding black-box predictions via influence functions. *In ICML*, 2017.
[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.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012.
[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.
[14] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. *In AAAI*, 2017.
[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 1998.
[16] 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.
[17] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *In NIPS*, 2017.
[18] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *In CVPR*, 2015.
[19] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.
[20] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. *In KDD*, 2016.
[21] 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.
[22] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *In NIPS*, 2017.
[23] 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 ICCV*, 2017.
[24] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015.
[25] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. *In ACCV*, 2014.
[26] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. *In arXiv:1312.6034*, 2013.
[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.
[28] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. *In ECCV*, 2012.
[29] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *In arXiv:1710.08864*, 2017.
[30] 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.
[31] J. Vaughan, A. Sudjianto, E. Brahimi, J. Chen, and V. N. Nair. Explainable neural networks based on additive index models. *in arXiv:1806.01933*, 2018.
[32] 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.
[33] 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.
[34] 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.
[35] N. Wolchover. New theory cracks open the black box of deep learning. *In Quanta Magazine*, 2017.
[36] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *In NIPS*, 2014.
[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014.
[38] Q. Zhang, R. Cao, F. Shi, Y. Wu, and S.-C. Zhu. Interpreting cnn knowledge via an explanatory graph. *In AAAI*, 2018.
[39] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable part graphs on convnets via multi-shot learning. *In AAAI*, 2016.
[40] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Mining object parts from cnns via active question-answering. *In CVPR*, 2017.
[41] Q. Zhang, R. Cao, S. Zhang, M. Edmonds, Y. N. Wu, and S.-C. Zhu. Interactively transferring cnn patterns for part localization. *In arXiv:1708.01783*, 2017.
[42] Q. Zhang, W. Wang, and S.-C. Zhu. Examining cnn representations with respect to dataset bias. *In AAAI*, 2018.
[43] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. *In CVPR*, 2018.
[44] Q. Zhang, Y. Yang, Y. N. Wu, and S.-C. Zhu. Interpreting cnns via decision trees. *In arXiv:1802.00121*, 2018.
[45] 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.
[46] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015.
[47] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016.

Quanshi Zhang received the B.S. degree in machine intelligence from Peking University, China, in 2009 and M.S. and Ph.D. degrees in center for spatial information science from the University of Tokyo, Japan, in 2011 and 2014, respectively. In 2014, he went to the University of California, Los Angeles, as a post-doctoral associate. Now, he is an associate professor at the Shanghai Jiao Tong University. His research interests include computer vision, machine learning, and robotics.



Ying Nian Wu received a Ph.D. degree from the Harvard University in 1996. He was an Assistant Professor at the University of Michigan between 1997 and 1999 and an Assistant Professor at the University of California, Los Angeles between 1999 and 2001. He became an Associate Professor at the University of California, Los Angeles in 2001. From 2006 to now, he is a professor at the University of California, Los Angeles. His research interests include statistics, machine learning, and computer vision.

Song-Chun Zhu received a Ph.D. degree from Harvard University, and is a professor with the Department of Statistics and the Department of Computer Science at UCLA. His research interests include computer vision, statistical modeling and learning, cognition and AI, and visual arts. He received a number of honors, including the Marr Prize in 2003 with Z. Tu et. al. on image parsing,the Aggarwal prize from the Int'l Association of Pattern Recognition in 2008, twice Marr Prize honorary nominations in 1999 for texture modeling and 2007 for object modeling with Y.N. Wu et al., a Sloan Fellowship in 2001, the US NSF Career Award in 2001, and the US ONR Young Investigator Award in 2001. He is a Fellow of IEEE.