Extraction of an Explanatory Graph to Interpret a CNN

Quanshi Zhang, Xin Wang, Ruiming Cao, Ying Nian Wu, Feng Shi, and Song-Chun Zhu

Abstract—This paper introduces an explanatory graph representation to reveal object parts encoded inside convolutional layers of a CNN. Given a pre-trained CNN, each filter[1] in a conv-layer usually represents a mixture of object parts. We develop a simple yet effective method to learn an explanatory graph, which automatically disentangles object parts from each filter without any part annotations. Specifically, given the feature map of a filter, we mine neural activations from the feature map, which correspond to different object parts. The explanatory graph is constructed to organize each mined part as a graph node. Each edge connects two nodes, whose corresponding object parts usually co-activate and keep a stable spatial relationship. Experiments show that each graph node consistently represented the same object part through different images, which boosted the transferability of CNN features. The explanatory graph transferred features of object parts to the task of part localization, and our method significantly outperformed other approaches.

Index Terms—Convolutional neural networks, graphical model, interpretable deep learning

1 Introduction

In this paper, we investigate the disentanglement of intermediate-layer feature representations of a CNN pre-trained for object classification. We notice that each filter in a CNN usually encodes mixed features of object parts and textural patterns. Therefore, in this paper, given a CNN, we propose to learn an explanatory graph without any part annotations. The explanatory graph automatically reveals how object-part features are organized in the CNN. The explanatory graph

1. disentangles features of object parts from mixed features in intermediate-layers of a CNN;

2. encodes which object parts are usually co-activated and keep the stable spatial relationship.

As Fig. 1 shows, the explanatory graph encodes the compositional hierarchy of object parts encoded inside conv-layers of the CNN, as follows.

- The explanatory graph consists of multiple layers. Each layer of the graph corresponds to a conv-layer of the CNN and contains thousands of nodes.
- Each node represents an object part that is encoded in a filter of a conv-layer. A filter in a conv-layer is usually activated by multiple parts and textural patterns. As Fig. 1 shows, a filter's feature map[1] may be activated by both the head and the neck of a horse.

  Given the feature map of a filter, a graph node can identify neural activations in the feature map, which correspond to a specific object part. Theoretically, a CNN with ReLU layers can be considered to encode high-order piecewise linear representations. An object part corresponding to a node is encoded inside a specific feature space divided by the piecewise partitions. Multiple nodes are learned for each filter, i.e., neural activations in its feature map are divided and explained as different multiple parts.
- A graph edge connects two nodes in adjacent layers. The two connected nodes represent two object parts, which usually appear simultaneously in an image and keep a stable spatial relationship among different images. For example, the ear part and the face part of a horse usually co-appear on different images with similar spatial relationships.

Constructing the explanatory graph is a process of mining object parts from intermediate conv-layers. Nodes in the explanatory graph represent all candidate parts learned from the entire set of training images by the CNN. Consequently, in the inference process, given an input image, the explanatory graph automatically selects a small ratio of nodes. These chosen nodes identify neural activations in intermediate-layer feature maps, which correspond to specific object parts. Given different images, the explanatory graph selects different sets of nodes for explanation. Moreover, since the same part may appear at various locations,

- *Q. Zhang and X. Wang are with John Hopcroft Center and MoE Key Lab of Artificial Intelligence AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {zqs1022, xin.wang}@sjtu.edu.cn.*
- *R. Cao, F. Shi, Y. Nian Wu, and S.-C. Zhu are with the University of California, Los Angeles, CA 90095 USA. E-mail: rcao@berkeley.edu, {ywu, sczhu}@stat.ucla.edu, shi.feng@cs.ucla.edu.*

Manuscript received 16 Dec. 2018; revised 30 Dec. 2019; accepted 21 Apr. 2020. Date of publication 4 May 2020; date of current version 1 Oct. 2021.
(Corresponding author: Quanshi Zhang.)
Recommended for acceptance by N. Vasconcelos.
Digital Object Identifier no. 10.1109/TPAMI.2020.2992207

1. The output of a conv-layer is called the feature map of a conv-layer. Each channel of this feature map is produced by a filter, so we call a channel the feature map of a filter.

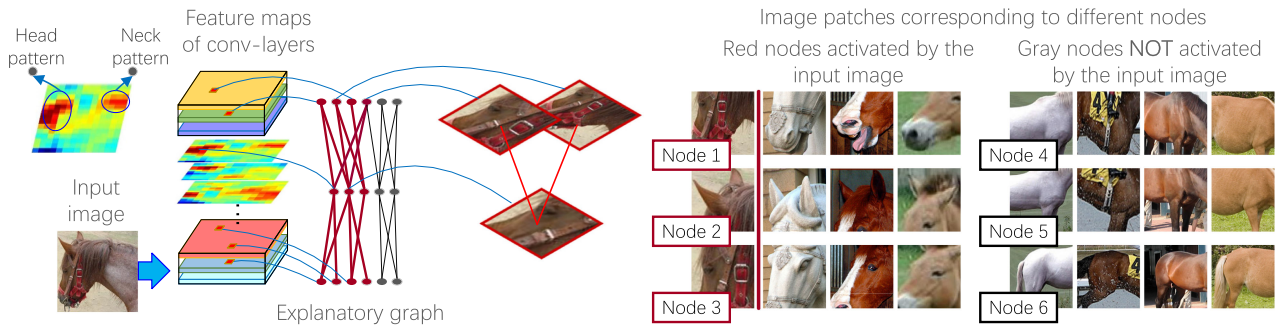

Fig. 1. An explanatory graph represents the compositional hierarchy of object parts encoded in conv-layers of a CNN. Each filter in a pre-trained CNN may be activated by different object parts. Our method disentangles object parts from each filter in an unsupervised manner.

given different images, the same node may identify neural activations at different positions as the target part.

The explanatory graph mainly takes two advantages, i.e., the disentanglement and the transferability, as follows.

Disentangling object parts from a single filter is the core technique of building an explanatory graph. In this study, we develop a simple yet effective method to automatically disentangle different object parts from a single filter without using any annotations of object parts, which presents considerable challenges for state-of-the-art algorithms. In this way, the explanatory graph exclusively localizes neural activations of object parts in the feature map, and ignores noisy activations and activations of textural patterns.

More specifically, for each input image, the explanatory graph (i) infers which parts (nodes) are responsible for the feature map of a filter and (ii) localizes these parts.

Graph Nodes With High Transferability. The explanatory graph contains off-the-shelf features of object parts in a compositional hierarchy, like a dictionary. Thus, the explanatory graph enables us to accurately transfer such object-part features to other tasks. Since all filters in the CNN are learned to encode common features shared by numerous training images, each graph node can be regarded as a transferable detector for common parts among different images.

To demonstrate the above advantages, we learn different explanatory graphs for various CNNs (e.g., the VGG-16, residual networks, and the encoder of a VAE-GAN) and analyze the explanatory graphs from various perspectives as follows.

Visualization & Reconstruction. We visualize object parts encoded by graph nodes using the following two approaches. First, for each graph node, we draw image regions corresponding to the node's part localizations on different input images. Second, we learn another neural network, which uses activation states of graph nodes to reconstruct the input image.

Evaluation of Part Interpretability of Graph Nodes. Given an explanatory graph, we propose a new metric to quantitatively evaluate whether a node consistently represents the same part in different images.

Examination of Location Instability of Graph Nodes. Besides the part interpretability, we also use a new metric, namely location instability, to measure the semantic clarity of each graph node. It is assumed that if a graph node consistently represents the same object part, then the distances between the inferred part and some ground-truth landmarks of the object should not change a lot through different images. Thus, the evaluation metric uses the deviation of such relative distances over images to measure the instability of the part representation.

Testing Transferability. The transferability of graph nodes is tested in the scenario of few-shot part localization. We associate certain graph nodes with explicit part names based on feature maps of very few images, in order to localize the target part. The superior localization performance proves the good transferability of graph nodes.

Contributions of this paper are summarized as follows.

- In this paper, we, for the first time, propose a simple yet effective method to extract and summarize object parts encoded inside intermediate conv-layers of a CNN and organize them using an explanatory graph. Experiments show that each graph node consistently represented the same object part in different input images.
- The proposed method can be used to learn explanatory graphs for various CNNs, e.g., VGGs, residual networks, and the encoder of a VAE-GAN.
- Graph nodes have good transferability, especially in the task of few-shot part localization. Although our graph nodes were learned without part annotations, our transfer-learning-based part localization still outperformed approaches using part annotations to learn part representations.

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

2 Related Work

2.1 Semantics in the CNN

The interpretability of neural networks receives increasing attention in recent years [51]. Different methods have been developed to explore visual concepts encoded inside a CNN.

Visualization & Interpretability of CNN Filters. Visualization of filters in a CNN is the most direct way of diagnosing representations of a CNN. Dosovitskiy *et al.* [6] proposed up-convolutional nets to invert feature maps of conv-layers to input images. Gradient-based visualization [19], [21], [22], [31] showed the appearance that maximized a network output, the activation score of a specific filter, or certain neural activations in a feature map. Furthermore, Bau *et al.* [3] defined and analyzed the interpretability of each filter. In recent years, [23] provided a reliable tool to visualize filters in different conv-layers of a CNN.

[3] selectively analyzed the semantics among the highest 0.5 percent activations of each filter. In contrast, our method provides a solution to explaining both strong and relatively weak activations from each filter, instead of exclusively extracting significant neural activations.

Active Network Diagnosis. Going beyond "passive" visualization, some methods "actively" diagnose a pre-trained CNN to obtain insight understanding of CNN representations. Many statistical methods [1], [35] have been proposed to analyze CNN features. [35] explored semantic meanings of convolutional filters. [1], [17] computed feature distributions of different categories.

The model bias and dataset bias are typical problems in deep learning, which has been illustrated in recent studies of [14], [20], [24]. Zhang *et al.* [47] has proposed a method to discover biased representations due to dataset bias. The CNN usually uses unreliable contexts for classification. For example, a CNN may extract features from hairs as a context to identify the smiling attribute.

Therefore, in order to ensure the correctness of feature representations, network-attack methods [12], [34], [35] diagnosed network representations by computing adversarial samples for a CNN. In particular, influence functions [12] were proposed to compute adversarial samples, create training samples to attack the learning of CNNs, fix the training set, and further debug a CNN. [13] discovered blind spots of CNN representations in a weakly-supervised manner. In comparison, our method disentangles features of object parts from a pre-trained CNN and builds an explanatory graph to reveal object parts encoded inside the CNN. It is because just like And-Or graphs [39], [40], [41], our explanatory graph naturally represents the local, bottom-up, and top-down information to construct a hierarchical object representation.

Diagnosis of Network Predictions. Some previous studies aimed to explain the reason for each network prediction. Methods of [7], [27] propagated gradients of feature maps *w.r.t.* the CNN loss back to the image, in order to estimate the image regions that directly contribute the network output. The LIME [24], the SHAP [18], and [4], [42] extracted input units that were closely related to a specific prediction.

Pattern Retrieval. Some studies retrieve specific activation units with specific meanings from intermediate-layer feature maps. Like middle-level feature extraction [33], pattern retrieval mainly learns mid-level representations of CNN features. Zhou *et al.* [52], [53] selected activation units from feature maps to describe scenes. In particular, [52] accurately computed the image-resolution receptive field of neural activations in a feature map. Theoretically, the actual receptive field of a neural activation is smaller than that computed using the filter size. Simon *et al.* discovered objects from feature maps of unlabeled images [29], and selected a filter to describe each part in a supervised fashion [30]. However, most methods simply assumed that each filter mainly encoded a single visual concept, and ignored the case that a filter in high conv-layers encoded a mixture of object parts and textural patterns. [44], [45], [46] extracted certain neural activation units from a filter's feature map to describe an object part in a weakly-supervised manner (i.e., learning from active question answering and human interactions).

In this study, the explanatory graph disentangles features of different object parts from the CNN without part annotations. Compared to raw feature maps, graph nodes are well disentangled and more interpretable.

CNN Semanticization. Compared to the diagnosis of CNN representations and the pattern retrieval, semanticization of CNN representations is closer to the spirit of building interpretable representations.

Hu *et al.* [11] designed logic rules for network outputs, and used these rules to regularize neural networks and learn meaningful representations. [3], [52] extracted visual semantics from intermediate layers of a CNN. [37] distilled representations of a neural network into an additive model to explain the network. [53] also used additive structures, i.e., the global average pooling layer to explain neural networks. [50] used a tree structure to approximate the rationale of the CNN prediction on each specific sample. Capsule nets [26] and interpretable CNNs [49] used specific network structures and loss functions, respectively, to make the network automatically encode interpretable features in intermediate layers.

In comparison, we aim to explore the compositional hierarchy of object parts encoded inside conv-layers of a CNN. The explanatory graph boosts the transferability of CNN features to other part-based tasks.

2.2 Weakly-Supervised Knowledge Transferring

Knowledge transferring has been widely used in deep learning. Typical research includes end-to-end fine-tuning and transferring CNN representations between different datasets [8]. In contrast, a transparent representation of the explanatory graph will create a new possibility of transferring object-part features to other applications. Experiments have demonstrated the superior transferability of graph nodes in few-shot part localization.

3 Algorithm

A single filter is usually activated by different parts of the object (see Fig. 2). Let us assume that given an input image, a filter is activated at N positions, i.e., there are N activation peaks on the filter's feature map. Some peaks represent common parts of the object. Other activation peaks may correspond to background noises or textural patterns.

Our goal is to disentangle activation peaks corresponding to object parts from a filter's feature map. I.e., we select certain neural activations, which represent specific object parts. We propose an explanatory graph for the disentanglement. Each activation peak of a filter corresponding to an object part is represented as a graph node. Let an activation peak represent a specific object part. Then, it is assumed that the CNN usually contains other filters to represent neighboring parts of the target part. I.e., some activation peaks of other filters must keep stable spatial relationships with the target part. Such spatial relationships are encoded in edges of the explanatory graph, which connect each node in a layer to some nodes in the neighboring upper layer.

Object parts are mined layer by layer. Given object parts mined from the upper layer, we extract activation peaks that keep stable spatial relationships with specific upper-layer parts through different images, as parts in the current layer.

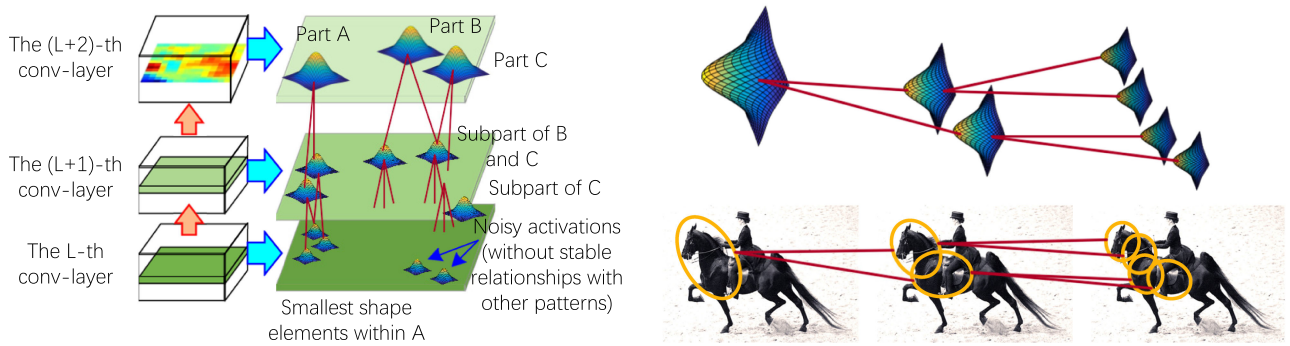

Fig. 2. Schematic illustration of the explanatory graph. The explanatory graph encodes spatial and co-activation relationships between object parts in the explanatory graph. Nodes in high layers help localize nodes in low layers. From another perspective, we can regard low-layer nodes represent compositional parts of high-layer nodes.

Nodes in high layers usually represent large-scale object parts, while nodes in low layers mainly describe small and relatively simple shapes, which are usually compositions of high-layer parts. Nodes in high layers are usually discriminative, and the explanatory graph uses high-layer nodes to filter out noisy activations. Nodes in low layers are disentangled based on their spatial relationship with high-layer nodes.

3.1 Explanatory Graph

Before the introduction of technical details of the algorithm, we first give a brief overview of the explanatory graph.

We are given a CNN, which is learned using a set of training samples $\mathbf{I}$. We construct an explanatory graph G based on this CNN and all training samples in $\mathbf{I}$. As Fig. 4 illustrates, G contains several layers, each corresponding to a single conv-layer in the CNN. Each layer of the explanatory graph is composed of hundreds/thousands of nodes, which represent object parts encoded in this conv-layer. Each node is linked with some graph nodes in the upper layer. The linkage/edge indicates that object parts of the two linked nodes usually co-appear in the image with stable spatial relationship. In this way, an explanatory graph can be considered as a dictionary of object parts, which are extracted from various images.

In the training phase, each node in G is supposed to disentangle a part from a conv-layer's feature maps. In inference phase, given feature maps of an input image I, the explanatory graph G uses its nodes to localize neural activations corresponding to different parts.

3.2 Top-Down Iterative Learning of the Explanatory Graph

Given all training images $\mathbf{I}$, we expect that (i) all nodes in the explanatory graph can be well fitted to feature maps of all images, and (ii) nodes in the lower layer always keep consistent spatial relationships with nodes in the upper layer given each input images. Therefore, the learning of an explanatory graph is conducted in a top-down manner as follows.

The learning of an explanatory graph is conducted layer by layer. We first disentangle parts from the top conv-layer of the CNN and construct the top layer of the explanatory graph. Then, we conduct position inferences for all nodes in the top layer (the inference process will be introduced in Section 3.3). We use inference results to help disentangle parts from the neighboring lower conv-layer. In this way, the lower layer of the explanatory graph is constructed using inference results of the neighboring upper layer.

Construction of the Lth[2] *Layer*. In the following paragraphs, we will introduce how to recursively learn the Lth layer of the explanatory graph given the $(L+1)$-th layer.

Our method disentangles the dth filter of the Lth conv-layer into $N_{L,d}$ parts. These parts are modeled as a set of $N_{L,d}$ nodes in the Lth layer of G, denoted by $\Omega_{L,d}$. $\Omega_L = \cup_d \Omega_{L,d}$ denotes the entire node set for the Lth layer. In following paragraphs, we can simply omit the subscript L without ambiguity. $\boldsymbol{\theta}$ represents parameters of nodes in the Lth layer, which mainly encode spatial relationships between these nodes and nodes in the $(L+1)$-th layer. Table 1 summarizes the notation used in this paper.

Given an input image $I \in \mathbf{I}$, the Lth conv-layer of the CNN generates a feature map[1], denoted by $\mathbf{X}^I$. Then, for each node $V \in \Omega_d$, the explanatory graph infers whether or not the part indicated by V appears in the dth channel[1] of $\mathbf{X}^I$, as well as its part location (if the part appears).

For each node V in the Lth layer, our method learns the following two terms: (i) the parameter $\mu_V \in \boldsymbol{\theta}$ and (ii) a set of nodes $E_V \in \boldsymbol{\theta}$ in the upper layer that are connected to V. $\mu_V \in \boldsymbol{\theta}$ denotes the prior location of V. Thus, for each node $V' \in E_V$, $\mu_V - \mu_{V'}$ corresponds the prior displacement between V and node V' in the upper layer. The explanatory graph uses the displacement $\mu_V - \mu_{V'}$ to model the spatial relationships between nodes.

Just like an EM algorithm, we use the current explanatory graph to fit feature maps of training images. Then, we use matching results as feedback to modify the prior location μ_V and edges E_V of each node V in the Lth layer, in order to make the explanatory graph better fit the feature maps. We repeat this process iteratively to obtain the optimal prior location and edges for V.

In other words, our method automatically extracts pairs of related nodes and learns the optimal spatial relationships

2. Note that our method is not limited to using consecutive conv-layers to learn the explanatory graph. People can select inconsecutive conv-layers. Without loss of generality, the Lth ranked layer among all conv-layers, which are selected from the CNN, is termed as the Lth conv-layer for simplicity.

TABLE 1
Notation

V	A node in the explanatory graph
$N_{L,d}$ (or N_d)	The node number extracted from the dth channel of the Lth conv-layer
$\Omega_{L,d}$ (or Ω_d)	The node set extracted from the dth channel of the Lth conv-layer
Ω_L (or Ω)	The node set extracted from the Lth conv-layer
$\boldsymbol{\theta}$	Parameters of nodes in the Lth layer
$\mathbf{X}^I$	The feature maps of the Lth conv-layer given input image I
$x \in \mathbf{X}^I$	A neural activation unit in the feature map $\mathbf{X}^I$
$\mathbf{R}^I$	Position inference results of nodes in the $L+1$th layer, which are represented using spatial coordinates.
$\mathbf{p}_x$	The center of the receptive field in the image plane of the neural activation unit x
$\mathbf{p}_{V'}$	The position inference result (i.e., the spatial coordinate) in the image plane of the node V' given input image I
μ_V	The average position of the node V in the image plane
E_V	The set of parent nodes of node V, which are localized in the upper layer.

between them during the iterative learning process, which best fit feature maps of training images.

Therefore, the objective function of learning the Lth layer is formulated as

$$\operatorname{argmax}_{\boldsymbol{\theta}} \prod\nolimits_{I\in\mathbf{I}} P(\mathbf{X}^I|\mathbf{R}^I,\boldsymbol{\theta}). \tag{1}$$

Let us focus on the feature map $\mathbf{X}^I$ of image I. Without ambiguity, we ignore the superscript I to simplify notations in following paragraphs. We can regard $\mathbf{X}$ as a distribution of "neural activation entities." The neural response of each unit $x \in \mathbf{X}$ can be considered as the number of "activation entities." In other words, each neural activation unit x in the feature map $\mathbf{X}$ is identified by its spatial position $\mathbf{p}_x$[3] and its channel number d_x (i.e., an activation unit of the d_x-th filter). We use $F(x) = \beta \cdot \max\{f_x, 0\}$ to measure the number of activation entities at the location $\mathbf{p}_x$, where f_x is the normalized activation value of x; β is a constant. We use $\mathbf{R}^I$ to represent position inference results of all nodes in the upper conv-layer (i.e., the $L+1$-th conv-layer).

Just like a Gaussian mixture model, all nodes in Ω_d comprise a mixture model, which explains the distribution of activation entities on the dth channel of $\mathbf{X}$.

$$\begin{aligned} P(\mathbf{X}|\mathbf{R},\boldsymbol{\theta}) &= \prod\nolimits_{x\in\mathbf{X}} P(\mathbf{p}_x|\mathbf{R},\boldsymbol{\theta})^{F(x)} \\ &= \prod\nolimits_{x\in\mathbf{X}} \left\{ \sum_{V\in\Omega_d\cup\{V_{\text{none}}\}} P(V)P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta}) \right\}^{F(x)}_{d=d_x}, \end{aligned} \tag{2}$$

where each node $V \in \Omega_d$ is treated as a hidden variable or an alternative component in the mixture model to describe activation entities. $P(V) = \frac{1}{N_d+1}$ is a constant prior probability. $P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta})$ measures the compatibility of using node V to describe an activation entity at $\mathbf{p}_x$. In particular, we add a dummy node V_{none} to the mixture model for noisy activations, in order to explain neural activations unrelated to object parts, e.g., those of noises and textural patterns. The compatibility between V and $\mathbf{p}_x$ is based on spatial relationship between V and its connected nodes in G, which is approximated as

$$P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta}) = \begin{cases} \gamma \prod_{V'\in E_V} P(\mathbf{p}_x|\mathbf{p}_{V'},\boldsymbol{\theta})^{\lambda} & V \in \Omega_{d_x} \\ \gamma\tau, & V = V_{\text{none}} \end{cases} \tag{3}$$

$$P(\mathbf{p}_x|\mathbf{p}_{V'},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{p}_x|\mu_{V'\to V}, \sigma^2_{V'}). \tag{4}$$

In the above equations, V has M related nodes in the upper layer. The set of nodes $E_V \in \boldsymbol{\theta}$ connected to V would be determined during the learning process. The overall compatibility $P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta})$ is divided into the spatial compatibility between node V and each related node V', $P(\mathbf{p}_x|\mathbf{p}_{V'},\boldsymbol{\theta})$. $\forall V' \in E_V$, $\mathbf{p}_{V'} \in \mathbf{R}$ denotes the position inference result of V', which have been given. $\lambda = \frac{1}{M}$ is a constant for normalization. γ is a constant, which roughly ensures $\int P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta})\mathbf{dp}_x = 1$ and can be eliminated during the learning process.

As Fig. 3 shows, an intuitive idea is that the relative displacement between V and V' should not change a lot among different images. Then, $\mathbf{p}_x - \mathbf{p}_{V'}$ will approximate to the prior displacement $\mu_V - \mu_{V'}$, if node V can well fit the activation at $\mathbf{p}_x$. Given E_V, we assume the spatial relationship between V and V' follows a Gaussian distribution in Eq. (4), where we define $\mu_{V'\to V} = \mu_V - \mu_{V'} + \mathbf{p}_{V'}$ as the prior localization of V given V'. The variation $\sigma^2_{V'}$ can be estimated from data.[4]

Algorithm 1. Learning Sub-Graph in the Lth Layer

Inputs: feature map $\mathbf{X}$ of the Lth conv-layer, inference results $\mathbf{R}$ in the upper conv-layer.
Outputs: μ_V, E_V for $\forall V \in \Omega$.
Initialization: $\forall V$, $E_V = \{V_{\text{dummy}}\}$, a random value for $\mu_V^{(0)}$
for $iter = 1$ to T **do**
  $\forall V \in \Omega$, compute $P(\mathbf{p}_x, V|\mathbf{R},\boldsymbol{\theta})$.
  for $V \in \Omega$ **do**
    Update μ_V via an EM algorithm,
    $\mu_V^{(iter)} = \mu_V^{(iter-1)} + \eta \sum_{I\in\mathbf{I},x\in\mathbf{X}} \mathbf{E}_{P(V|\mathbf{p}_x,\mathbf{R},\boldsymbol{\theta})} \left[F(x) \cdot \frac{\partial \log P(\mathbf{p}_x,V|\mathbf{R},\boldsymbol{\theta})}{\partial \mu_V} \right]$.
    Select M nodes from the upper layer $V' \in \Omega_{L+1}$ to construct E_V based on a greedy strategy, which maximize $\prod_{I\in\mathbf{I}} P(\mathbf{X}|\mathbf{R},\boldsymbol{\theta})$.
  end for
end for

The explanatory graph is learned in a top-down manner, and the learning process is summarized in Algorithm 1. Our method first learns nodes in the top-layer of G, and then learns for the neighboring lower layer. For the sub-graph in the Lth layer, our method recursively estimates μ_V and E_V for nodes in the sub-graph.

The special case is the node in the top conv-layer. For each node V in the top conv-layer, we simply define $E_V = \{V_{\text{dummy}}\}$, $\mu_{V_{\text{dummy}}} = \mathbf{p}_{V_{\text{dummy}}} = \mathbf{0}$, where V_{dummy} is a

3. To make unit positions in different conv-layers comparable with each other (e.g., $\mu_{V'\to V}$ in Eq. (4)), we project the position of unit x to the image plane. We define the coordinate $\mathbf{p}_x$ on the image plane, instead of on the feature-map plane.

4. We can prove that for each $V \in \Omega_d$, $P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta}) \propto \mathcal{N}(\mathbf{p}_x|\mu_V + \Delta_{I,V}, \tilde{\sigma}^2_V)$, where $\Delta_{I,V} = \sum_{V'\in E_V} \frac{\mathbf{p}_{V'}-\mu_{V'}}{\sigma^2_{V'}} / \sum_{V'\in E_V} \frac{1}{\sigma^2_{V'}}$; $\tilde{\sigma}^2_V = 1/\mathbf{E}_{V'\in E_V} \frac{1}{\sigma^2_{V'}}$. Therefore, we can either directly use $\tilde{\sigma}^2_V$ as σ^2_V, or compute the variation of $\mathbf{p}_x - \mu_V - \Delta_{I,V}$ *w.r.t.* different images to obtain σ^2_V.

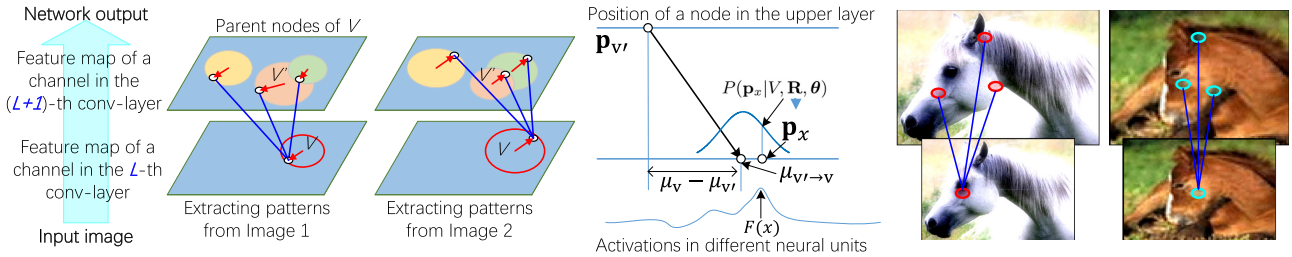

Fig. 3. Schematic illustration of related nodes V and V'. The related nodes keep similar spatial relationships among different images. Circle centers represent the prior part positions, e.g., μ_V and $\mu_{V'}$. Red arrows denote relative displacements between the inferred positions and prior positions, e.g., $\mathbf{p}_V - \mu_V$. In particular, the middle sub-figure illustrates different variables in a one-dimensional space for simplicity.

node in the dummy layer above the top conv-layer. Based on Eqs. (3) and (4), we obtain $P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{p}_x|\mu_V,\sigma_V^2)$.

3.3 Part Localization

Given feature maps of an input image, we can assign nodes with different activations peaks on feature maps, in order to infer object parts represented by these neural activations. The explanatory graph simply assigns node $V \in \Omega_d$ with the unit $\hat{x} = \text{argmax}_{x\in\mathbf{X}:d_x=d} S^I_{V\to x}$ on the feature map as the inference of V. $S^I_{V\to x} = F(x)P(\mathbf{p}_x|V,\mathbf{R},\boldsymbol{\theta})$ denotes the score of assigning V to x. Accordingly, $\mathbf{p}_{V'} = \mathbf{p}_{\hat{x}}$ represents the inferred location of V. In particular, in Eqn. (1), we define $\mathbf{R} = \{\mathbf{p}_{V'}\}_{V'\in\Omega_{L+1}}$.

4 Experiments

In this section, we conducted several experiments to demonstrate the effectiveness, board applicability, and the high accuracy of our method. We learned explanatory graphs to interpret four types of CNNs, i.e., the VGG-16 [32], the 50-layer and 152-layer Residual Networks [10], and the encoder of the VAE-GAN [15]. These CNNs learned using a total of 37 animal categories in three datasets, which included the ILSVRC 2013 DET Animal-Part dataset [44], the CUB200-2011 dataset [38], and the VOC Part dataset [5]. As discussed in [5], [44], animals usually contain non-rigid parts, which presents a key challenge for part localization. Thus, we selected animal categories in the three datasets for testing.

We designed three experiments to evaluate the explanatory graph from different perspectives. In the first experiment, we visualized object parts corresponding to nodes in the explanatory graph. The second experiment was designed to evaluate the interpretability of nodes, i.e., checking whether or not a node consistently represents the same object part among different images. We compared our nodes with three types of middle-level features and network features. In the third experiment, we used our graph nodes for the task of few-shot part localization, in order to test the transferability of nodes. We learned an And-Or graph (AOG) with very few part annotations, which associated the well learned nodes with explicit part names. We used the AOG to conduct part localization and compared its performance with fourteen baselines.

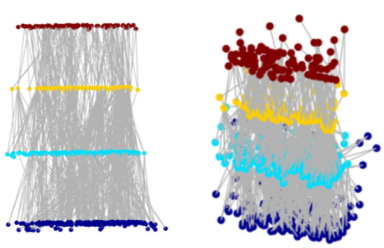

Fig. 4. A four-layer explanatory graph. For clarity, we put all nodes of different filters in the same conv-layer on the same plane and only show 1 percent of the nodes with 10 percent of their edges from two perspectives.

4.1 Implementation Details

We first trained/fine-tuned a CNN using object images of a category, which were cropped using object bounding boxes. Then, we set parameters $\tau = 0.1$, $M = 15$(except for results in Table 9), $T = 20$, and $\beta = 1$ to learn an explanatory graph for the CNN.

We learned explanatory graphs for the VGG-16, residual networks, and the VAE-GAN. We mainly extracted object parts from high conv-layers of these neural networks, because as discussed in [3], high conv-layers contain large-scale parts.

- *VGG-16:* The VGG-16 was first pre-trained using the 1.3M images in the ImageNet dataset [25]. We then fine-tuned all conv-layers of the VGG-16 using object images in a category. The loss for fine-tuning was for binary classification between the target category and background images. The VGG-16 has thirteen conv-layers and three fully connected layers. We selected the ninth, tenth, twelfth, and thirteenth conv-layers of the VGG-16 as four valid conv-layers, and accordingly, we built a four-layer graph. We extracted N_d nodes from the dth filter of the Lth layer, where we set $N_d = 40$ for all channels of the first and second conv-layers ($L = 1$, or 2) and set $N_d = 20$ for all channels of the third and fourth conv-layer ($L = 3$, or 4).
- *Residual Networks:* Two residual networks, i.e., the 50-layer and 152-layer ones, were used in experiments. The fine-tuning process for each network was exactly the same as that for VGG-16. We built a three-layer graph based on each residual network by selecting the last conv-layer with a $28 \times 28 \times 128$ feature output, the last conv-layer with a $14 \times 14 \times 256$ feature map, and the last conv-layer with a $7 \times 7 \times 512$ feature map as valid conv-layers. We set N_d as 40, 20, and 10 for all channels in the first, second, and third conv-layers, respectively.
- *VAE-GAN:* For each category, we used the cropped object images to train a VAE-GAN. We learned a three-layer graph based on all three conv-layers of the encoder of the VAE-GAN. We set N_d as 52, 26, and 13 for all channels for the first, second, and third conv-layers, respectively.

4.2 Experiment 1: Part Visualization

The global structure of an explanatory graph for a VGG-16 network is visualized in Fig. 4. Fig. 12 shows the histogram

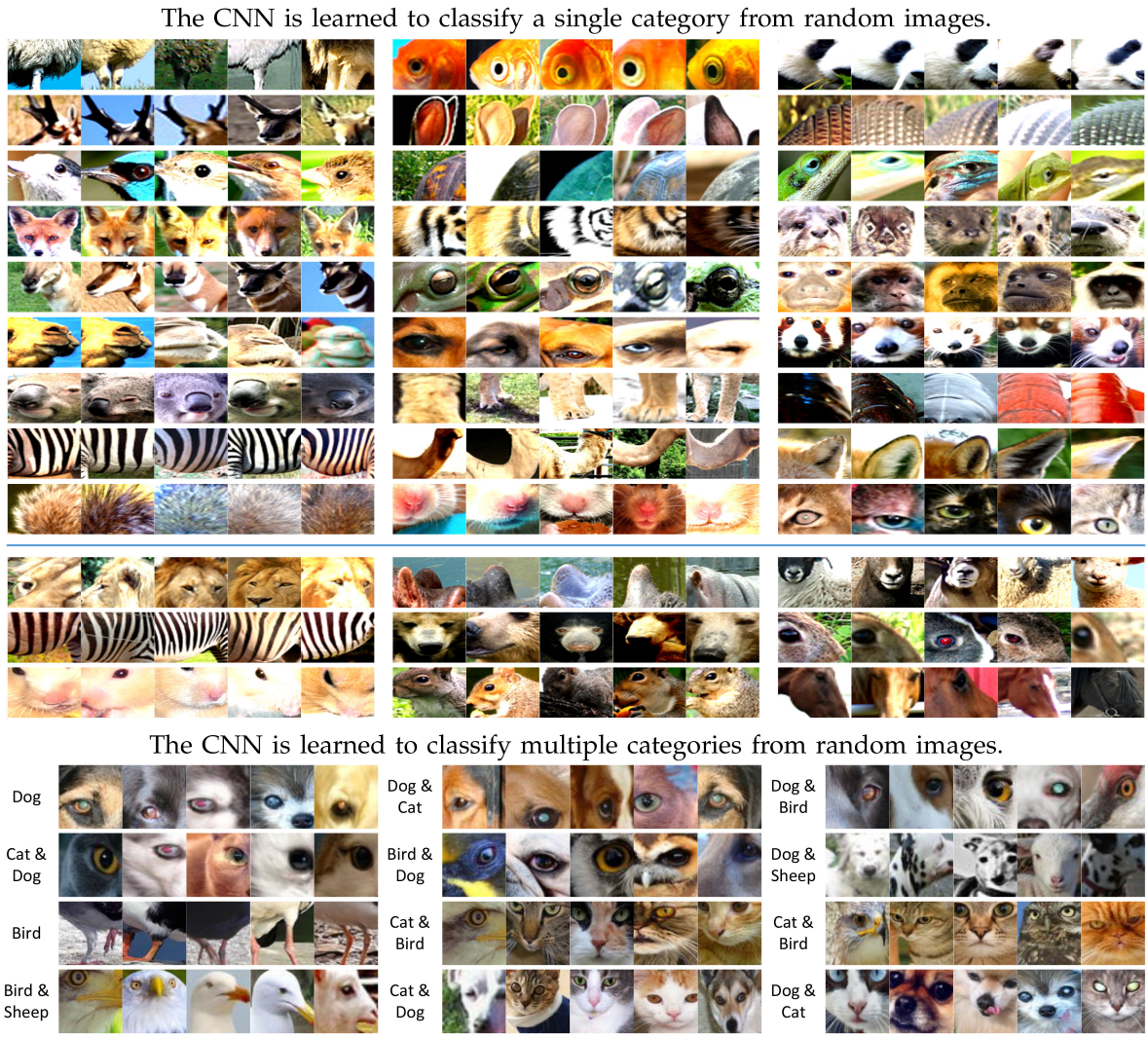

Fig. 5. Image patches corresponding to different nodes in explanatory graphs. We visualized nodes in explanatory graphs that were learned for two types of CNNs, i.e., CNNs learned for a single category and CNNs for multiple categories. (1) The top nine layers visualized nodes corresponding to CNNs, each learned for a single category. We used two methods to infer the image patch for each node. In top nine rows, part location was inferred as $\hat{x} = \text{argmax}_{x \in \mathbf{X}: d_x = d} S^I_{V \to x}$. In following three rows, parts were localized via $\hat{x} = \text{argmax}_{x \in \mathbf{X}: d_x = d} |f_x \cdot \frac{\partial y}{\partial f_x}|$. (2) The bottom four layers visualized image patches of graph nodes, when the CNN was learned to classify multiple categories. In this case, each node usually encoded parts shared by different categories. Texts before each group of image patches indicate their corresponding categories. Part location was inferred as $\hat{x} = \text{argmax}_{x \in \mathbf{X}: d_x = d} S^I_{V \to x}$. Please read texts for detailed explanations.

of $P(\mathbf{p}_x | \mathbf{p}_{V'}, \theta)$ values among all edges in an explanatory graph. In general, the distribution of $P(\mathbf{p}_x | \mathbf{p}_{V'}, \theta)$ satisfied the assumption of the Gaussian distribution. Fig. 13 demonstrates the convergence of our method.

We visualized object parts of graph nodes from the following three perspectives.

Top-Ranked Patches. For each image I, we performed the part localization on its feature maps. For a node V, we extracted a patch at the location of $\mathbf{p}_{\hat{x}_V}$[5] with a fixed scale of $70\,pixels \times 70\,pixels$ to represent V. Fig. 5 shows a part's image patches that had highest inference scores. In this figure, we used two different methods to infer the object part for each node. The first method was $\hat{x} = \text{argmax}_{x \in \mathbf{X}: d_x = d} S^I_{V \to x}$ as mentioned before. The second method incorporated gradients to localize parts, i.e., $\hat{x} = \text{argmax}_{x \in \mathbf{X}: d_x = d} |f_x \cdot \frac{\partial y}{\partial f_x}|$, where y and f_x denote the classification output of the target class and the activation value of the neural activation unit x, respectively. $|f_x \cdot \frac{\partial y}{\partial f_x}|$ is a classical evaluation of the numerical attribution of the neural activation unit x [47].

Note that in this study, we assumed that the CNN was learned to classify a single category from random images. However, it would be quite interesting if we visualized graph nodes corresponding to a CNN encoding parts of multiple categories. To this end, we learned a VGG-16 network to classify six animal categories (*bird*, *cat*, *cow*, *dog*, *horse*, *sheep*) from other fourteen categories in the VOC Part dataset [5] and built an explanatory graph for the CNN. Fig. 5 also visualizes nodes in this explanatory graph. Each node usually represented parts that were shared by multiple categories

Heatmaps of the Distribution of Object Parts. Given part localization results *w.r.t.* a cropped object image I, we drew heatmaps to show the spatial distribution of the inferred parts. We drew a heatmap for each layer L of the graph. Each part $V \in \Omega$ was visualized as a weighted Gaussian distribution $\alpha \cdot \mathcal{N}(\mu = \mathbf{p}_V, \sigma^2_V)$[5] on the heatmap, where $\alpha = S^I_{V \to \hat{x}}$.

5. We projected the unit to the image to compute its position.

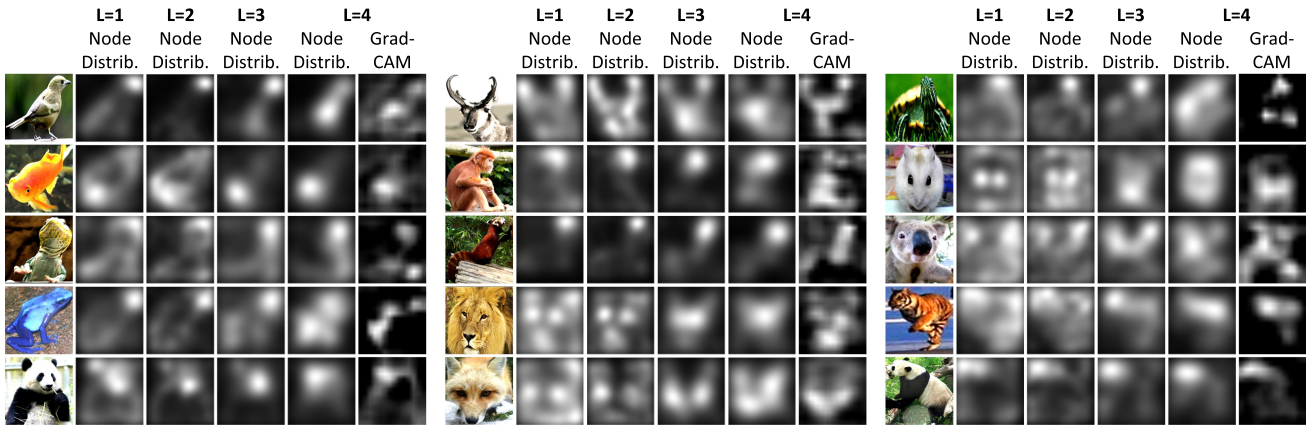

Fig. 6. Heatmaps of the distribution of object parts. We use a heatmap to visualize the spatial distribution of the top-50 percent object parts in the Lth layer of the explanatory graph with the highest inference scores. We also compare heatmaps with the grad-CAM [27] of the feature map. Unlike the grad-CAM, our heatmaps mainly focus on the foreground of an object and uniformly pay attention to all parts, rather than only focus on most discriminative parts.

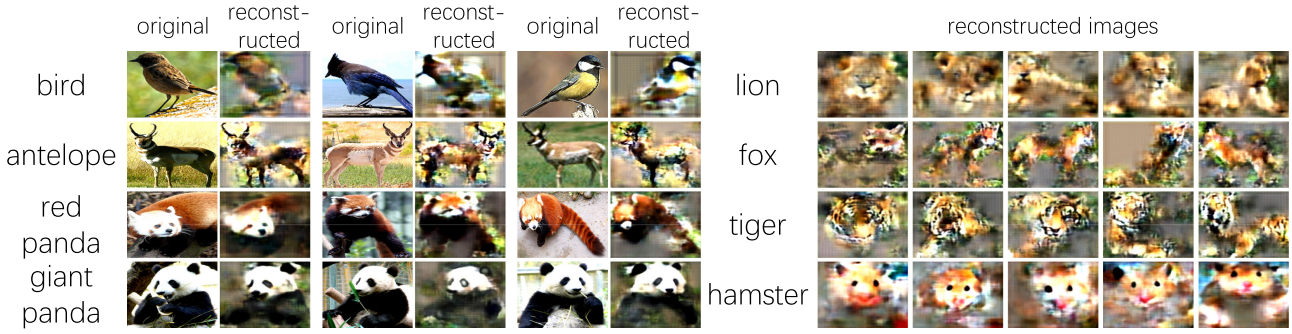

Fig. 7. Image synthesis based on the activation of nodes on an image. The explanatory graph only encodes major parts encoded in conv-layers with considerable information loss. Synthesis results demonstrate that the nodes are automatically learned to represent foreground appearance, and ignore background noises and trivial details of objects.

Fig. 6 shows heatmaps of the top-50 percent parts with the highest scores of $S^I_{V\to\hat{x}}$. Due to the lack of the ground truth for explanations, it is difficult to evaluate the attribution/attention/saliency map of a neural network. In general, two terms have to be considered in the evaluation, i.e., (1) whether or not the attribution map fits the human cognition and (2) whether or not the attribution map objectively reflects true reasons for the network prediction. From this perspective, in Fig. 6, results of the Grad-CAM better fit human cognition than our method. On the other hand, Fig. 6 visualizes the distribution of graph nodes, whose semantic meanings were verified in experiments. Therefore, the explanatory graph can better show object parts encoded in the CNN than the Grad-CAM method.

Node-Based Image Synthesis. We used the up-convolutional network [6] to visualize parts of graph nodes. Given an object image I, we used the explanatory graph for part localization, i.e., assigning each node V with a certain neural activation unit $\hat{x}_V$ as its position inference[5]. We considered the top-10 percent nodes with highest scores of $S^I_{V\to\hat{x}}$ as valid ones. We filtered out all neural responses of units, which were not assigned to valid nodes, from feature maps (setting these responses to zero). We selected the filtered feature map corresponding to the second graph layer and used the up-convolutional network to synthesize the filtered feature map to the input image. Fig. 7 shows image-synthesis results, which can be regarded as the visualization of the inferred nodes.

4.3 Experiment 2: Semantic Interpretability of Nodes

In this experiment, we evaluated whether or not each node consistently represented the same object part through different images. Four explanatory graphs were built for a VGG-16 network, two residual networks, and a VAE-GAN. These networks were learned using the CUB200-2011 dataset [38]. We used the following two metrics to measure the interpretability of nodes.

Part Interpretability of Nodes. The evaluation metric was inspired by Zhou *et al*. [52]. For each given node V, we used V to localize object parts among all images. We regarded inference results with the top-K inference scores $S^{I_i}_V$ among all images as valid representations of V. We required the K highest inference scores $S^{I_i}_V$ on images $\{I_1,\ldots,I_k\}$ to take about 30 percent of the inference energy, i.e., we use $\sum_{i=1}^{K} S^{I_i}_V = 0.3\sum_{i\in\mathbf{I}} S^I_V$ to compute K. We asked human raters to count the number of inference results, which described the same object part, among the top K, in order to compute the purity of part semantics of node V. In addition, as mentioned before, $|f_x \cdot \frac{\partial y}{\partial f_x}|$ is a classical evaluation of the numerical attribution of the neural activation unit x [47]. Thus, we designed a baseline method, namely *Ours with top-ranked* $|f_x \cdot \frac{\partial y}{\partial f_x}|$, to select inference results with top-ranked $|f_x \cdot \frac{\partial y}{\partial f_x}|$ values that took 30 percent of the total $|f_x \cdot \frac{\partial y}{\partial f_x}|$ score of all images.

The table in Fig. 8(top-left) shows the semantic purity of the nodes in the second layer of the graph. Let the second

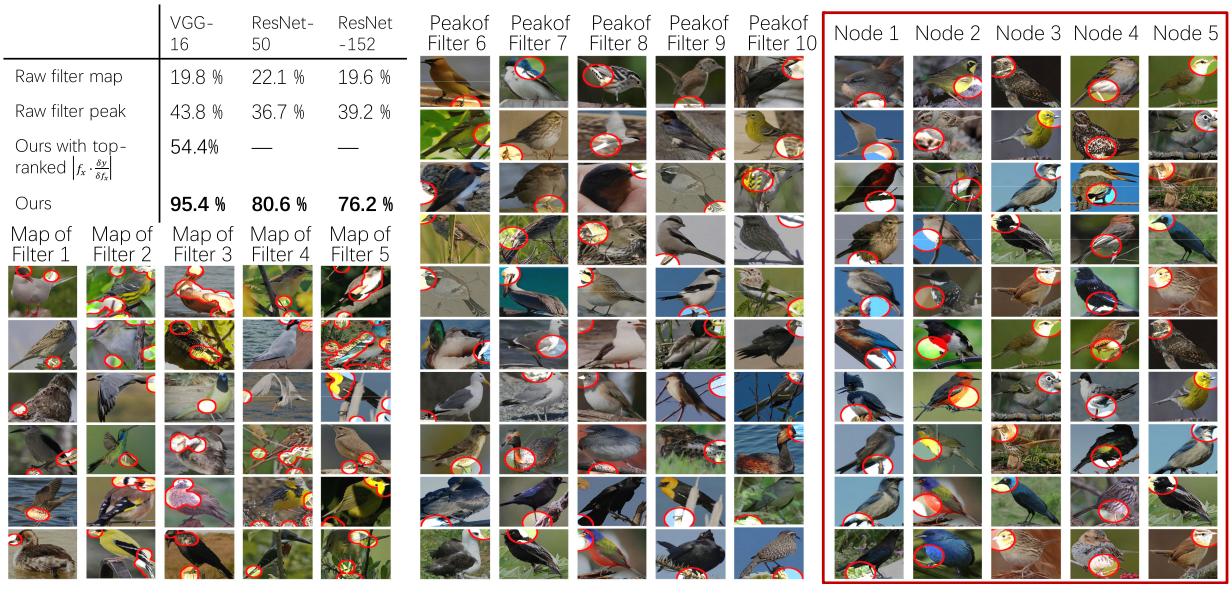

Fig. 8. Purity of part semantics (top-left). We compared object parts corresponding to nodes in the explanatory graph with features of raw filters. We draw raw feature maps of filters (left), the highest activation peaks on feature maps of filters (middle), and image regions corresponding to each node in the explanatory graph (right). Based on such visualization results, we use human users to annotate the semantic purity of each node/filter.

graph layer correspond to the Lth conv-layer with D filters. The *raw filter maps* baseline used all neural activation in the feature map of a filter to describe a part. The *raw filter peaks* baseline considered the highest peak on a filer's feature map as the part detection. Like our method, the two baselines also visualized top-K' part inferences (the K' feature maps' neural activations took 30 percent of activation energies over all images). We back-propagated the center of the receptive field of each neural activation to the image plane and draw the image region corresponding to each neural activation. Fig. 8 compares the image region corresponding to each graph node and image regions corresponding to feature maps of each filter. Our graph nodes represented explicit object parts, but raw filters encoded mixed semantics.

Because the baselines simply averaged the semantic purity among the D filters, we also computed average semantic purities using the top-D nodes with the highest scores of $\sum_{i\in\mathbf{I}} S_V^I$ to enable a fair comparison.

Location Instability of Inference Positions. We defined the location instability for each node as another evaluation metric of interpretability. Note that we used the localization of object parts, rather than the localization of entire objects, to evaluate the clarity of semantic meanings of each node. We assumed that if a node was always activated by the same object part through different images, then the distance between the node's inference position and a ground-truth landmark of the object part should not change a lot among various images.

As Fig. 9 shows, given a testing image I, d_I^{head}, d_I^{back}, and d_I^{tail} denote the distances between the inferred position of V and ground-truth landmark positions of *head*, *back*, and *tail* parts, respectively. These distances were normalized by the diagonal length of input images. Then, the node's location instability was given as $(\sqrt{var(d_I^{\text{head}})}+\sqrt{var(d_I^{\text{back}})}+\sqrt{var(d_I^{\text{tail}})})/3$, where $var(d_I^{\text{head}})$ denotes the variation of d_I^{head} over different images.

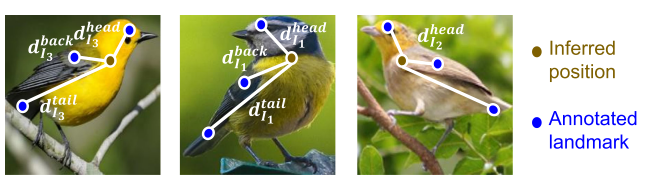

Fig. 9. Notation for the computation of location instability.

We compared its location instability of an explanatory graph with three baselines. The first baseline treated each filter in a CNN as a detector of a certain part. Thus, given the feature map of a filter (after the ReLu operation), we used the method of [52] to localize the unit with the highest response value as the part position. The other two baselines were typical methods to extract middle-level features from images [33] and extract parts from CNNs [30], respectively. For each baseline, we chose the top-500 parts, i.e., 500 nodes with top scores in the explanatory graph, 500 filters with strongest activations in the CNN, and the top-500 middle-level features. For each node, we selected position inferences on the top-20 images with highest scores to compute the location instability. Table 2 compares the location instability of different baselines. Nodes in the explanatory graph had significantly lower location instability than baselines.

4.4 Experiment 3: Few-Shot Part Localization

4.4.1 Hybrid And-Or Graph for Semantic Parts

The explanatory graph makes it plausible to transfer intermediate-layer features of a CNN to semantic object

TABLE 2
Location Instability of Nodes

	ResNet-50	ResNet-152	VGG-16	VAE-GAN
Raw filter [52]	0.1328	0.1346	0.1398	0.1944
Ours	0.0848	0.0858	0.0638	0.1066
[33]	0.1341			
[30]	0.2291			

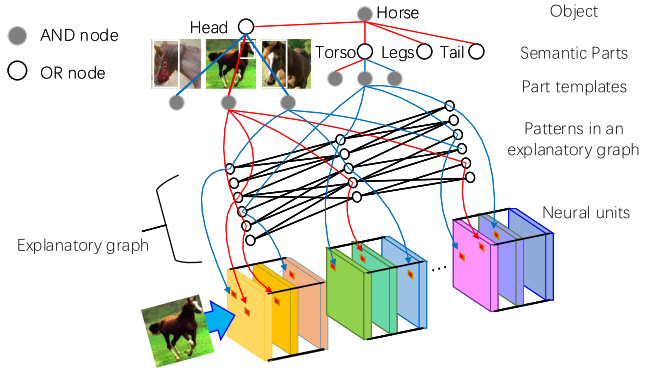

Fig. 10. Schematic illustration of an And-Or graph for semantic object parts. The AOG encodes a four-layer hierarchy for each semantic part, i.e., the semantic part (OR node), part templates (AND node), latent parts (OR nodes, those from the explanatory graph), and neural activation units (terminal nodes). In the AOG, the OR node of semantic part contains a number of alternative appearance candidates as children. Each OR node of a latent part encodes a list of neural activation units as alternative deformation candidates. Each AND node (e.g., a part template) uses a number of latent parts to describe its compositional regions.

parts. In this section, we further designed a hybrid And-Or graph (AOG) to connect the explanatory graph, and the AOG associated nodes in the explanatory graph with explicit part names.

We used the AOG to test the transferability of nodes in the explanatory graph. It is because the AOG has been demonstrated as a classical model, which is suitable for representing the compositional hierarchy of objects [28], [54]. Adapting nodes in the explanatory graph enabled us to evaluate the clarify of compositional hierarchy that was encoded in a pre-trained CNN.

The structure of the AOG is inspired by [48], and the learning of the AOG was originally proposed in [44]. As Fig. 10 shows, the AOG encodes a four-layer hierarchy for each semantic part, i.e., the semantic part (OR node), part templates (AND node), latent parts (OR nodes, i.e., nodes in the explanatory graph), and neural activation units (terminal nodes).

Layer	Name	Node type	Notation
1	semantic part	OR node	V^{sem}
2	part template	AND node	$V^{\text{tmp}} \in \Omega^{\text{tmp}}$
3	latent part	OR node	$V^{\text{lat}} \in \Omega^{\text{lat}}$
4	neural unit	Terminal node	$x \in \Omega^{\text{unt}}$

where latent parts correspond to nodes from the explanatory graph.

In the AOG, each OR node (e.g., a semantic part or a latent part) contains a list of alternative appearance (or deformation) candidates. Each AND node (e.g., a part template) uses a number of latent parts to describe its compositional regions.

- The OR node of a semantic part contains a total of m part templates to represent alternative appearance or pose candidates of the part.
- Each part template (AND node) retrieve K latent parts from the explanatory graph as children. These latent parts describe compositional regions of the part.
- Each latent part (OR node) has all units in its corresponding filter's feature map as children, which represent its deformation candidates on image I.

Technical Details. Based on the AOG, we use the extracted latent parts to infer semantic parts in a bottom-up manner. We first compute inference scores of different units at the bottom layer *w.r.t.* different latent parts, and then we propagate inference scores up to the layers of part templates and the semantic part for part localization.

The top OR node of the semantic part V^{sem} contains a total of m part templates to represent alternative appearance or pose candidates of the part. We manually define the composition of the M part templates. During part-inference process, given an image I, V^{sem} selects its best child as the true part template:

$$\begin{aligned} S_{V^{\text{sem}}} &= \max_{V^{\text{tmp}} \in Child(V^{\text{sem}})} S_{V^{\text{tmp}}} \\ \mathbf{p}_{V^{\text{sem}}} &= \mathbf{p}_{\hat{V}^{\text{tmp}}}, \end{aligned} \tag{5}$$

where $S_{V^X}, X \in \{\text{sem}, \text{tmp}, \text{lat}, \text{unit}\}$ denotes the inference score of V^X.

Then, each part template V^{tmp} uses a number of latent parts to describe sub-regions of the part. In the scenario of one-shot learning, we only annotate one part sample belonging to the part template. Then, we retrieve latent parts (nodes) that are related to the annotated part from all nodes in the disentangling graph. Given the inference score $S_{V^{\text{lat}}}$ and inferred position $\mathbf{p}_{V^{\text{lat}}}$ of each latent part V^{lat} on I, we retrieve the top K latent parts with the highest scores of $S_{V^{\text{lat}}} \mathcal{N}(\mathbf{p}_{V^{\text{lat}}} | \mu = \mathbf{p}^*_{V^{\text{tmp}}}, \sigma^2)$ as children of V^{tmp}. $\mathbf{p}^*_{V^{\text{tmp}}}$ denotes the annotated position of the part V^{tmp}; $\sigma^2 = (0.3 \times image\ width)^2$ is a constant variation.

When we have extracted a set of latent parts for a part template, given a new image, we can use inference results of the latent parts to localize the part template:

$$\begin{aligned} S_{V^{\text{tmp}}} &= \sum_{V^{\text{lat}} \in Child(V^{\text{tmp}})} S_{V^{\text{lat}}} \\ \mathbf{p}_{V^{\text{tmp}}} &= \operatorname*{mean}_{V^{\text{lat}} \in Child(V^{\text{tmp}})} \left\{ \mathbf{p}_{V^{\text{lat}}} + \Delta \mathbf{p}_{V^{\text{lat}}, V^{\text{tmp}}} \right\} \end{aligned} \tag{6}$$

where $\Delta \mathbf{p}_{V^{\text{lat}}, V^{\text{tmp}}}$ denotes a constant displacement from V^{lat} to V^{tmp}.

Each latent part V^{lat} has a channel of units as children, which represent its deformation candidates on image I. The score of each unit x is given as $S_{V^{\text{lat}} \to x} = F(x) P(\mathbf{p}_x | V^{\text{lat}}, \mathbf{R}, \boldsymbol{\theta})$. The OR node of V^{lat} selects the unit with the maximum score as its deformation configuration:

$$\begin{aligned} S_{V^{\text{lat}}} &= \max_{x: V^{\text{lat}} \in \Omega_{d_x}} S_{V^{\text{lat}} \to x} \\ \mathbf{p}_{V^{\text{lat}}} &= \mathbf{p}_{\hat{x}}. \end{aligned} \tag{7}$$

Please see [44] for details of the AOG.

4.4.2 Experimental Settings of Three-Shot Learning

Given a fine-tuned VGG-16 network, we learned an explanatory graph and built the AOG upon the explanatory graph following the scenario of few-shot learning in [44]. For each category, we set three templates for the head part ($m = 3$) and used three part-box annotations for the three templates. Note that we used object images without part annotations to learn the explanatory graph, and we used three part annotations provided by [44] for each part to build the AOG. All these object-box annotations and part annotations

Fig. 11. Localization results based on AOGs that are learned using three annotations of the head part.

were equally provided to all baselines to enable fair comparisons (besides part annotations, all baselines also used object annotations contained in the datasets for learning). We set $K = 0.1\sum_{L,d} N_{L,d}$ to learn AOGs for categories in the ILSVRC Animal-Part and CUB200 datasets and set $K = 0.4\sum_{L,d} N_{L,d}$ for VOC Part categories. Then, we used the AOGs to localize semantic parts on objects.

Baselines. We compared AOGs with a total of fourteen baselines for part localization. The baselines included (i) approaches for object detection (i.e., directly detecting target parts from objects), (ii) graphical/part models for part localization, and (iii) the methods selecting CNN features to describe object parts.

The first baseline was the standard fast-RCNN [9], namely *Fast-RCNN (1 ft)*, which directly fine-tuned a VGG-16 network based on part annotations. Then, the second baseline, namely *Fast-RCNN (2 fts)*, first used massive object-box annotations in the target category to fine-tune the VGG-16 network with the loss of object detection. Then, given part annotations, Fast-RCNN (2 fts) further fine-tuned the VGG-16 to detect object parts. We used [30] as the third baseline, namely *CNN-PDD*. CNN-PDD selected certain filters of a CNN to localize the target part. In CNN-PDD, the CNN was pre-trained using the ImageNet dataset [25]. Just like Fast-RCNN (2 ft), we extended [30] as the fourth baseline *CNN-PDD-ft*, which fine-tuned a VGG-16 network using object-box annotations before applying the technique of [30]. The fifth and sixth baselines were DPM-related methods, i.e., the strongly supervised DPM (*SS-DPM-Part*) [2] and the technique in [16] (*PL-DPM-Part*), respectively. Then, the seventh baseline, namely *Part-Graph*, used a graphical model for part localization [5]. For weakly supervised learning, "simple" methods are usually insensitive to model over-fitting. Thus, we designed six baselines as follows. First, we used object-box annotations in a category to fine-tune the VGG-16 network. Then, given a few well-cropped object images, we used the selective search [36] to collect image patches, and used the VGG-16 network to extract *fc7* features from these patches. The baselines *fc7+linearSVM*, *fc7+RBF-SVM*, *fc7+NN* used a linear SVM, an RBF-SVM, and the nearest-neighbor method (selecting the patch closest to the annotated part), respectively, to detect the target part. The other three baseline *fc7+sp+linearSVM*, *fc7+sp+RBF-SVM*, *fc7+sp+NN* combined both the *fc7* feature and the spatial position (x, y) $(-1 \leq x, y \leq 1)$ of each image patch as features for part detection. The last competing method is weakly supervised mining of parts from the CNN [44], namely *supervised-AOG*. Unlike our method (unsupervised), *supervised-AOG* used part annotations to extract parts.

Comparisons. We divided all baselines into three groups. The first group, namely *not-learn parts*, included traditional methods without using deep features, such as SS-DPM-Part, PL-DPM-Part, and Part-Graph. These methods did not learn deep features[6]. The second group, termed *super-learn parts*, contained Fast-RCNN (1 ft), Fast-RCNN (2 ft), CNN-PDD, CNN-PDD-ft, supervised-AOG, fc7+linearSVM, and fc7+sp+linearSVM. These methods learned deep features using part annotations, e.g., fast-RCNN methods used part annotations to learn features; supervised-AOG used part annotations to select filters from the CNN to localize parts. The third group (*unsuper-learn parts*) included CNN-PDD, CNN-PDD-ft, and our method. These methods learned deep features using object-level annotations, rather than part annotations.

Fig. 11 visualizes localization results based on AOGs, which were learned using three annotations of the head part of each category. We used the normalized distance (used in [30], [44]) and the traditional intersection-over-union (IoU) criterion to evaluate the localization performance. Tables 3, 4, 5, 6, and 7 show part-localization results on the CUB200-2011 dataset [38], the VOC Part dataset [5], and the ILSVRC 2013 DET Animal-Part dataset [44]. AOGs based on our graph nodes exhibited outperformed all baselines in few-shot learning. Note that our AOGs simply localized the

TABLE 3
Normalized Distance of Part Localization on the CUB200-2011 Dataset [38]

	Method	obj.-box fine-tune	
not learn parts	SS-DPM-Part [2]	N	0.3469
	PL-DPM-Part [16]	N	0.3412
	Part-Graph [5]	N	0.4889
unsuper-learn[6] parts	CNN-PDD [30]	N	0.2333
	CNN-PDD-ft [30]	Y	0.3269
	Ours	Y	**0.0862**
super-learn parts	fc7+linearSVM	Y	0.3120
	fc7+sp+linearSVM	Y	0.3120
	Fast-RCNN (1 ft) [9]	N	0.4517
	Fast-RCNN (2 fts) [9]	Y	0.4131

The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

6. Representation learning in these methods only used object-box annotations, which is independent to part annotations. A few part annotations were used to select off-the-shelf pre-trained features.

TABLE 4
Normalized Distance of Part Localization on the VOC Part Dataset [5]

	obj.-box fine-tune		bird	cat	cow	dog	horse	sheep	Avg.
not learn parts	SS-DPM-Part [2]	N	0.356	0.270	0.264	0.242	0.262	0.286	0.280
	PL-DPM-Part [16]	N	0.294	0.328	0.282	0.312	0.321	0.840	0.396
	Part-Graph [5]	N	0.360	0.208	0.263	0.205	0.386	0.500	0.320
unsuper-learn[6] parts	CNN-PDD [30]	N	0.301	0.246	0.220	0.248	0.292	0.254	0.260
	CNN-PDD-ft [30]	Y	0.358	0.268	0.220	0.200	0.302	0.269	0.269
	Ours	Y	**0.152**	**0.121**	0.303	**0.135**	**0.231**	**0.246**	**0.198**
super-learn parts	fc7+linearSVM	Y	0.247	0.174	0.251	0.217	0.261	0.317	0.244
	fc7+sp+linearSVM	Y	0.247	0.174	**0.249**	0.217	0.261	0.317	0.244
	Fast-RCNN (1 ft) [9]	N	0.324	0.324	0.325	0.272	0.347	0.314	0.318
	Fast-RCNN (2 fts) [9]	Y	0.350	0.295	0.255	0.293	0.367	0.260	0.303

The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

TABLE 5
Accuracy of Part Localization Evaluated by "$IoU \geq 0.5$" on the Pascal VOC Part dataset [5]

	obj.-box fine-tune		bird	cat	cow	dog	horse	sheep	Avg.
not learn parts	SS-DPM-Part [2]	N	0.0	1.3	1.6	1.9	1.1	3.3	1.5
	PL-DPM-Part [16]	N	0.5	1.1	4.4	0.4	0.0	0.0	1.1
	Part-Graph [5]	N	2.9	22.6	12.1	11.0	3.2	0.0	8.6
unsuper-learn[6] parts	**Ours**	Y	**20.2**	**34.9**	8.2	**33.8**	10.0	14.5	**20.3**
super-learn parts	fc7+linearSVM	Y	8.0	27.6	7.1	10.4	16.1	6.2	12.6
	fc7+sp+linearSVM	Y	8.0	27.6	7.1	10.4	**16.1**	6.2	12.6
	fc7+RBF-SVM	Y	5.3	26.0	7.7	8.9	14.7	8.3	11.8
	fc7+sp+RBF-SVM	Y	5.0	26.3	7.1	8.8	15.1	8.7	11.8
	fc7+NN	Y	1.9	21.0	3.8	4.7	3.6	5.0	6.7
	fc7+sp+NN	Y	1.9	21.0	3.8	4.7	3.6	5.0	6.7
	Fast-RCNN (1 ft) [9]	N	2.1	2.2	2.2	1.9	1.4	7.0	2.8
	Fast-RCNN (2 fts) [9]	Y	7.7	24.0	**18.7**	18.0	5.0	**19.4**	15.5

The second column indicates whether the baseline used all object annotations in the category to pre-finetune a CNN before learning the part.

TABLE 6
Normalized Distance of Part Localization on the ILSVRC 2013 DET Animal-Part Dataset [44]

	obj.-box fine-tune		gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion
not learn parts	SS-DPM-Part	N	0.297	0.280	0.257	0.255	0.317	0.222	0.207	0.239	0.305	0.308	0.238
	PL-DPM-Part	N	0.273	0.256	0.271	0.321	0.327	0.242	0.194	0.238	0.619	0.215	0.239
	Part-Graph	N	0.363	0.316	0.241	0.322	0.419	0.205	0.218	0.218	0.343	0.242	0.162
unsuper-learn[6] parts	CNN-PDD	N	0.316	0.289	0.229	0.260	0.335	0.163	0.190	0.220	0.212	0.196	0.174
	CNN-PDD-ft	Y	0.302	0.236	0.261	0.231	0.350	0.168	0.170	0.177	0.264	0.270	0.206
	Ours	Y	**0.090**	**0.091**	**0.095**	0.167	**0.124**	**0.084**	**0.155**	0.147	**0.081**	**0.129**	**0.074**
super-learn parts	fc7+linearSVM	Y	0.150	0.318	0.186	0.150	0.257	0.156	0.196	0.136	0.101	0.138	0.132
	fc7+sp+linearSVM	Y	0.150	0.318	0.186	**0.150**	0.254	0.156	0.196	**0.136**	0.101	0.138	0.132
	Fast-RCNN (1 ft)	N	0.261	0.365	0.265	0.310	0.353	0.365	0.289	0.363	0.255	0.319	0.251
	Fast-RCNN (2 fts)	Y	0.340	0.351	0.388	0.327	0.411	0.119	0.330	0.368	0.206	0.170	0.144
			tiger	bear	rabb.	hams.	squi.	horse	zebra	swine	hippo	catt.	sheep
not learn parts	SS-DPM-Part	N	0.144	0.260	0.272	0.178	0.261	0.246	**0.206**	0.240	0.234	0.246	0.205
	PL-DPM-Part	N	0.136	0.323	0.228	0.186	0.281	0.322	0.267	0.297	0.273	0.271	0.413
	Part-Graph	N	0.127	0.224	0.188	0.131	0.208	0.296	0.315	0.306	0.378	0.333	0.230
unsuper-learn[6] parts	CNN-PDD	N	0.160	0.223	0.266	0.156	0.291	0.261	0.266	**0.189**	0.192	0.201	0.244
	CNN-PDD-ft	Y	0.256	0.178	0.167	0.286	0.237	0.310	0.321	0.216	0.257	0.220	0.179
	Ours	Y	**0.102**	**0.121**	**0.087**	**0.097**	**0.095**	**0.189**	0.212	0.212	0.151	**0.185**	**0.124**
super-learn parts	fc7+linearSVM	Y	0.163	0.122	0.139	0.110	0.262	0.205	0.258	0.201	0.140	0.256	0.236
	fc7+sp+linearSVM	Y	0.163	0.122	0.139	0.110	0.262	0.205	0.258	0.201	**0.140**	0.256	0.236
	Fast-RCNN (1 ft)	N	0.260	0.317	0.255	0.255	0.169	0.374	0.322	0.285	0.265	0.320	0.277
	Fast-RCNN (2 fts)	Y	0.160	0.230	0.230	0.178	0.205	0.346	0.303	0.212	0.223	0.228	0.195
			ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.			Avg.
not learn parts	SS-DPM-Part	N	0.224	0.277	0.253	0.283	0.206	0.219	0.256	0.129			0.242
	PL-DPM-Part	N	0.337	0.261	0.286	0.295	0.187	0.264	0.204	0.505			0.284
	Part-Graph	N	0.216	0.317	0.227	0.341	0.159	0.294	0.276	0.094			0.257
unsuper-learn[6] parts	CNN-PDD	N	0.208	0.193	0.174	0.299	0.236	0.214	0.222	0.179			0.225
	CNN-PDD-ft	Y	0.229	0.253	0.198	0.308	0.273	0.189	0.208	0.275			0.240
	Ours	Y	**0.093**	**0.120**	**0.102**	**0.188**	**0.086**	0.174	**0.104**	**0.073**			**0.125**
super-learn parts	fc7+linearSVM	Y	0.164	0.190	0.140	0.252	0.256	0.176	0.215	0.116			0.184
	fc7+sp+linearSVM	Y	0.164	0.190	0.140	0.250	0.256	0.176	0.215	0.116			0.184
	Fast-RCNN (1 ft)	N	0.255	0.351	0.340	0.324	0.334	0.256	0.336	0.274			0.299
	Fast-RCNN (2 fts)	Y	0.175	0.247	0.280	0.319	0.193	**0.125**	0.213	0.160			0.246

The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

TABLE 7
Accuracy of Part Localization Evaluated by "$IoU \geq 0.5$" on the ILSVRC 2013 DET Animal-Part Dataset [44]

obj.-box finetune		gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion	tiger	bear	rabb.	hams.	squi.
SS-DPM-Part [2]	N	1.5	0.0	1.2	2.6	0.7	8.8	1.4	5.2	0.0	10.9	13.4	20.4	7.0	0.5	6.5	0.5
PL-DPM-Part [16]	N	0.0	1.0	0.0	0.6	0.0	3.3	0.0	3.3	0.0	23.8	8.8	3.6	0.0	1.6	22.3	0.0
Part-Graph [5]	N	2.0	5.5	5.9	6.5	7.4	12.1	3.5	9.0	1.9	18.7	40.7	56.1	15.0	27.3	37.7	21.4
fc7+linearSVM	Y	20.0	2.0	13.5	20.8	7.4	30.2	1.4	27.5	55.9	39.4	43.3	27.0	46.5	44.3	60.5	8.8
fc7+RBF-SVM	Y	4.5	0.0	2.4	24.7	5.9	34.0	0.7	15.6	29.9	42.5	53.1	39.3	19.0	44.8	41.4	0.9
fc7+NN	Y	1.0	0.0	1.2	7.1	2.2	28.4	1.4	5.2	19.4	20.2	52.1	39.8	5.0	17.5	32.6	0.5
fc7+sp+linearSVM	Y	20.0	2.0	13.5	20.8	7.4	30.2	1.4	27.5	55.9	39.4	43.3	27.0	**46.5**	44.3	60.5	8.8
fc7+sp+RBF-SVM	Y	4.5	0.0	1.8	**24.7**	4.4	34.4	0.7	14.7	29.9	41.5	53.1	38.8	19.0	44.3	41.9	0.9
fc7+sp+NN	Y	1.0	0.0	1.2	7.1	2.2	28.4	1.4	5.2	19.4	20.2	52.1	39.8	5.0	17.5	32.6	0.5
Fast-RCNN (1 ft) [9]	N	5.0	0.5	1.8	2.6	3.7	3.3	0	0.5	28.9	11.4	22.2	11.7	2.5	20.2	27.9	36.3
Fast-RCNN (2 fts) [9]	Y	4.5	5.0	2.4	4.5	2.2	68.8	1.4	9.0	46.0	**50.8**	61.3	**65.8**	29.0	30.1	56.3	**40.9**
Ours	Y	**33.0**	**40.3**	**48.8**	18.2	**21.4**	**61.9**	**3.5**	**30.3**	**62.1**	26.4	**61.9**	49.5	36.0	**65.6**	**64.7**	25.6
		horse	zebra	swine	hippo	catt.	sheep	ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.		Avg.
SS-DPM-Part [2]	N	9.5	1.1	0.6	1.1	7.0	14.7	12.4	0.9	0.5	4.5	12.4	11.8	2.2	49.1		7.0
PL-DPM-Part [16]	N	5.8	0.0	0.6	0.5	0.5	0.0	0.0	0.0	0.0	0.0	9.1	2.6	28.1	0.0		3.9
Part-Graph [5]	N	10.0	13.0	4.9	4.3	7.0	19.0	23.0	5.6	18.2	6.6	18.3	2.6	16.2	58.6		15.9
fc7+linearSVM	Y	16.3	10.7	22.0	31.9	4.9	20.2	26.3	23.7	35.3	11.6	12.4	36.8	22.8	48.6		25.7
fc7+RBF-SVM	Y	7.9	27.1	7.3	14.4	2.7	14.1	25.3	16.3	37.4	13.6	10.8	22.4	26.8	54.5		21.3
fc7+NN	Y	2.1	22.6	1.2	1.1	2.2	6.1	2.3	8.8	40.6	10.6	7.0	5.3	21.1	55.9		14.0
fc7+sp+linearSVM	Y	16.3	10.7	22.0	31.9	4.9	20.2	26.3	**23.7**	35.3	12.1	12.4	36.8	22.4	48.6		25.7
fc7+sp+RBF-SVM	Y	7.9	27.1	7.3	14.4	2.7	14.1	19.4	16.3	37.4	13.6	9.1	22.4	27.6	55.0		21.0
fc7+sp+NN	Y	2.1	22.6	1.2	1.1	2.2	6.1	2.3	8.8	**40.6**	10.6	7.0	5.3	21.1	55.9		14.0
Fast-RCNN (1 ft) [9]	N	3.2	6.8	11.0	11.2	1.6	7.4	23.0	1.9	2.1	2.5	3.8	11.8	14.5	19.5		10.0
Fast-RCNN (2 fts) [9]	Y	6.3	15.3	**39.0**	34.6	**36.2**	**43.6**	46.5	20.5	26.7	13.1	36.6	**56.6**	47.8	57.3		31.9
Ours	Y	**37.9**	**35.6**	15.2	**41.0**	27.6	39.9	**53.5**	15.8	20.9	**28.3**	**55.4**	32.9	**51.8**	**67.3**		**39.1**

The second column indicates whether the baseline used all object annotations in the category to pre-finetune a CNN before learning the part.

center of an object part without sophisticatedly modeling the scale of the part. Thus, detection-based methods, which also estimated the part scale, performed better in very few cases. Table 8 compares the unsupervised and supervised learning of parts. In the experiment, our method outperformed all baselines, even including approaches that learned part features using part annotations. Finally, Table 9 compares the part-localization performance when we set different edge numbers M for each node. It shows that explanatory graphs with each node containing 15 edges usually performed better in the perspective of the intersection-over-union (IoU) criterion, and explanatory graphs with each node containing 25 edges exhibited lower normalized distances of part localization.

Note that we tested the explanatory graph and its corresponding AOG from the perspective of part localization, instead of evaluating their performance of object recognition. It is because the explanatory graph was proposed to explain object-part semantics in intermediate layers of the CNN, and the AOG was designed for part localization (i.e., estimating the part location under the condition that the image contains the target part), instead of object recognition (i.e., identifying whether or not the target object appears). Moreover, theoretically, it was difficult for nodes in the explanatory graph to outperform the original CNN, because the explanatory graph selectively retrieved part-alike neural activations from high conv-layers, and ignored other activations, whereas fully-connected layers in the CNN used all information (including both object parts and textures) to recognize objects. I.e., the original CNN used much richer information than the explanatory graph.

TABLE 8
Normalized Distance of Part Localization

Dataset	ILSVRC DET Animal	VOC Part	CUB200 -2011
Supervised-AOG	0.1344	0.1767	0.0915
Ours (unsupervised)	**0.1250**	**0.1765**	**0.0862**

We compared supervised and unsupervised mining of parts.

TABLE 9
Effects of the Edge Number M

Normalized distance	bird	cow	cat	dog	horse	sheep	Avg.
$M=10$	0.148	0.118	0.309	0.132	0.229	0.240	0.196
$M=15$	0.152	0.121	0.303	0.135	0.231	0.246	0.198
$M=20$	0.145	0.119	0.288	0.132	0.220	0.227	0.189
$M=25$	0.152	0.121	0.283	0.133	0.220	0.218	**0.188**
Accuracy of part localization	bird	cow	cat	dog	horse	sheep	Avg.
$M=10$	20.7	33.2	8.2	33.5	11.1	13.2	20.0
$M=15$	20.2	34.9	8.2	33.8	10.0	14.5	**20.3**
$M=20$	19.9	33.5	8.2	32.8	9.7	13.6	19.6
$M=25$	18.6	34.2	8.2	33.1	9.7	13.6	19.6

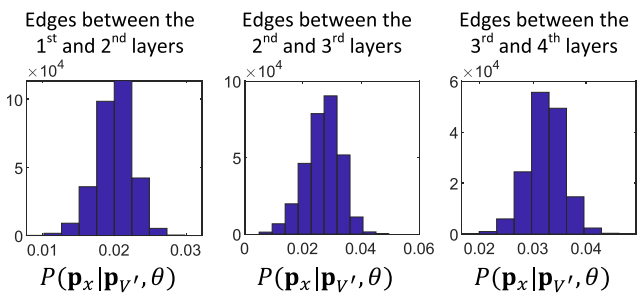

Fig. 12. Histogram of $P(\mathbf{p}_x|\mathbf{p}_{V'},\boldsymbol{\theta})$ values among all edges in an explanatory graph for the *cat* category.

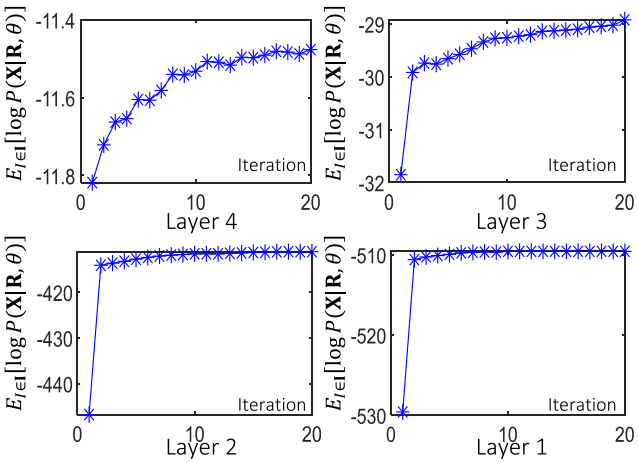

Fig. 13. Convergence of the learning process. We showed the average value of $\log P(\mathbf{X}^I|\mathbf{R}^I,\boldsymbol{\theta})$ after different iterations during the learning process.

5 Conclusion and Discussions

In this paper, we have developed a simple yet effective method to learn an explanatory graph that reveals the compositional hierarchy of object parts encoded inside conv-layers of a pre-trained CNN. The explanatory graph filters out noisy activations, disentangles object parts from each filter, and models co-activation relationships and spatial relationships between parts. Experiments showed that our graph nodes had significantly higher stability than baselines. More crucially, our method can be applied to different types of networks, including the VGG-16, residual networks, and the VAE-GAN, to explain their conv-layers.

The transparent representation of the explanatory graph boosts the transferability of CNN features. Part-localization experiments well demonstrated the good transferability of graph nodes. Our method even outperformed the supervised learning of part representations. Nevertheless, the explanatory graph is just a rough representation of the CNN. It is still difficult to well disentangle textural patterns from filters of the CNN.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (U19B2043 and 61906120), 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2875–2883.
[2] H. Azizpour and I. Laptev, "Object detection using strongly-supervised deformable part models," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012.
[3] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3319–3327.
[4] A. Binder, G. Montavon, S. Bach, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," in *Proc. 25th Int. Conf. Artif. Neural Netw. Mach. Learn.*, 2016, pp. 63–71.
[5] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1979–1986.
[6] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4829–4837.
[7] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3449–3457.
[8] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation in backpropagation," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1180–1189.
[9] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[11] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. P. Xing, "Harnessing deep neural networks with logic rules," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 2410–2420.
[12] P. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1885–1894.
[13] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz, "Identifying unknown unknowns in the open world: Representations and policies for guided exploration," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2124–2132.
[14] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nat. Commun.*, vol. 10, 2019, Art. no. 1096.
[15] A. B. L. Larsen, S. K. Sønderby, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1558–1566.
[16] B. Li, W. Hu, T. Wu, and S.-C. Zhu, "Modeling occlusion by discriminative AND-OR structures," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2560–2567.
[17] Y. Lu, "Unsupervised learning on neural network outputs with application in zero-shot learning," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 3432—3438.
[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.
[19] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5188–5196.
[20] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," in *Google Research Blog*, pp. 1–8, 2015. [Online]. Available: http://googleresearch.blogspot.it/2015/06/inceptionism-going-deeper-into-neural.html
[21] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3395–3403.
[22] A. Nguyen, J. Yosinski, and J. Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," in *Proc. ICML Vis. Deep Learn. Workshop*, 2016.
[23] C. Olah, A. Mordvintsev, and L. Schubert, "'Feature visualization," *Distill*, 2017. [Online]. Available: https://distill.pub/2017/feature-visualization
[24] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?" explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.
[25] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, 3, pp. 211–252, 2015.
[26] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3859–3869.
[27] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 618–626.
[28] Z. Si and S.-C. Zhu, "Learning and-or templates for object recognition and detection," *IEEE Trans. Pattern Anal. Mach Intell.*, vol. 35, no. 9, pp. 2189–2205, Sep. 2013.

[29] M. Simon and E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1143–1151.
[30] M. Simon, E. Rodner, and J. Denzler, "Part detector discovery in deep convolutional neural networks," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 162–177.
[31] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv preprint arXiv:1312.6034*.
[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[33] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 73–86.
[34] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.
[35] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014.
[36] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, 2, pp. 154–171, 2013.
[37] J. Vaughan, A. Sudjianto, E. Brahimi, J. Chen, and V. N. Nair, "Explainable neural networks based on additive index models," 2018, *arXiv preprint arXiv:1806.01933*.
[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200–2011 dataset," California Institute of Technology, Pasadena, CA, *Tech. Rep. CNS-TR-2011-001*, 2011.
[39] T. Wu and S.-C. Zhu, "A numerical study of the bottom-up and top-down inference processes in and-or graphs," *Int. J. Comput. Vis.*, vol. 93, no. 2, pp. 226–252, 2011.
[40] T.-F. Wu, G.-S. Xia, and S.-C. Zhu, "Compositional boosting for computing hierarchical image structures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[41] X. Yang, T. Wu, and S.-C. Zhu, "Evaluating information contributions of bottom-up and top-down processes," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 1042–1049.
[42] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[43] Q. Zhang, R. Cao, F. Shi, Y. Wu, and S.-C. Zhu, "Interpreting CNN knowledge via an explanatory graph," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2124–2132.
[44] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu, "Growing interpretable graphs on convnets via multi-shot learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2898–2906.
[45] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu, "Mining object parts from CNNs via active question-answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3890–3899.
[46] Q. Zhang, R. Cao, S. Zhang, M. Edmonds, Y. N. Wu, and S.-C. Zhu, "Interactively transferring CNN patterns for part localization," 2017, *arXiv preprint arXiv:1708.01783*.
[47] Q. Zhang, W. Wang, and S.-C. Zhu, "Examining CNN representations with respect to dataset bias," in *Proc. Assoc. Advancement Artif. Intell.*, 2018, pp. 4464–4473.
[48] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "A cost-sensitive visual question-answer framework for mining a deep and-or object semantics from web images," 2017, *arXiv preprint arXiv:1708.03911*.
[49] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8827–8836.
[50] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting CNNs via decision trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6254–6263.
[51] Q. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: A survey," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 1, pp. 27–39, 2018.
[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene CNNs," in *Proc. Int. Conf. Learn. Representations*, 2015.
[53] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2921–2929.
[54] S. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations Trends Comput. Graph. Vis.*, vol. 2, 4, pp. 259–362, 2006.

Quanshi Zhang received the BS degree in machine intelligence from Peking University, China, in 2009, and the MS and PhD degrees from the Center for Spatial Information Science, University of Tokyo, Japan, in 2011 and 2014, respectively. In 2014, he went to the University of California, Los Angeles, as a postdoctoral associate. Currently, he is working as an associate professor at Shanghai Jiao Tong University. His research interests include computer vision, machine learning, and robotics.

Xin Wang is currently working toward the PhD degree as an internship student at Shanghai Jiao Tong University. His research interests include machine learning and computer vision.

Ruiming Cao received the BS degree in computer science from the University of California, Los Angeles, in 2017. Currently, he is working toward the master's degree at the University of California, Los Angeles. His research interest includes computer vision.

Feng Shi is currently working toward the PhD degree at the University of California, Los Angeles. His research interests include computer vision and electric engineering.

Ying Nian Wu received the PhD degree from the Harvard University, in 1996. He was an assistant professor with the University of Michigan between 1997 and 1999 and an assistant professor with the University of California, Los Angeles between 1999 and 2001. He became an associate professor with the University of California, Los Angeles, in 2001. From 2006 till now, he is a professor with the University of California, Los Angeles. His research interests include statistics, machine learning, and computer vision.

Song-Chun Zhu received the PhD degree from Harvard University, in 1996, and is a professor with the Departments of Statistics and Computer Science at UCLA. He has published more than 300 papers in computer vision, statistical modeling and learning, cognition, Language, robotics, and AI. He received a number of honors, including the Marr Prize in 2003, the Aggarwal Prize from the Intal association of pattern recognition in 2008, the Holmholtz Test-of-Time Prize in 2013, twice Marr Prize honorary nominations in 1999 and 2007. A Sloan Fellowship, the US NSF Career Award, and ONR Young Investigator Award in 2001.