The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Interpreting CNN Knowledge via An Explanatory Graph

**Quanshi Zhang, Ruiming Cao, Feng Shi,
Ying Nian Wu, Song-Chun Zhu**
University of California, Los Angeles

Abstract

This paper learns a graphical model, namely an explanatory graph, which reveals the knowledge hierarchy hidden inside a pre-trained CNN. Considering that each filter[1] in a conv-layer of a pre-trained CNN usually represents a mixture of object parts, we propose a simple yet efficient method to automatically disentangles different part patterns from each filter, and construct an explanatory graph. In the explanatory graph, each node represents a part pattern, and each edge encodes co-activation relationships and spatial relationships between patterns. More importantly, we learn the explanatory graph for a pre-trained CNN in an unsupervised manner, *i.e.* without a need of annotating object parts. Experiments show that each graph node consistently represents the same object part through different images. We transfer part patterns in the explanatory graph to the task of part localization, and our method significantly outperforms other approaches.

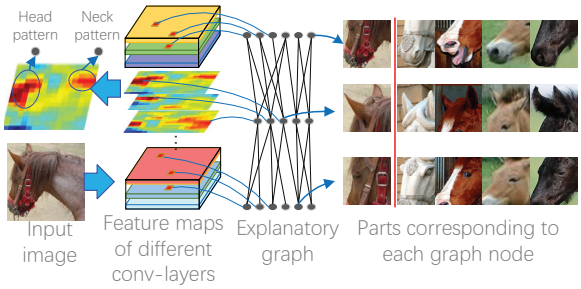

Figure 1: An explanatory graph represents knowledge hierarchy hidden in conv-layers of a CNN. Each filter in a pre-trained CNN may be activated by different object parts. Our method disentangles part patterns from each filter in an unsupervised manner, thereby clarifying the knowledge representation.

Introduction

Convolutional neural networks (CNNs) (LeCun et al. 1998; Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Li et al. 2015) have achieved superior performance in object classification and detection. However, the end-to-end learning strategy makes the entire CNN a black box. When a CNN is trained for object classification, we believe that its conv-layers have encoded rich implicit patterns (*e.g.* patterns of object parts and patterns of textures). Therefore, in this research, we aim to provide a global view of how visual knowledge is organized in a pre-trained CNN, which presents considerable challenges. For example,

1 How many types of patterns are memorized by each convolutional filter of the CNN (here, a pattern may describe a specific object part or a certain texture)?

2 Which patterns are co-activated to describe an object part?

3 What is the spatial relationship between two patterns?

In this study, given a pre-trained CNN, we propose to mine mid-level object part patterns from conv-layers, and we organize these patterns in an explanatory graph in an unsupervised manner. As shown in Fig. 1, the explanatory graph explains the knowledge hierarchy hidden inside the CNN. The explanatory graph disentangles the mixture of part patterns in each filter's feature map[1] of a conv-layer, and uses each graph node to represent a part.

- **Representing knowledge hierarchy:** The explanatory graph has multiple layers, which correspond to different conv-layers of the CNN. Each graph layer has many nodes. We use these graph nodes to summarize the knowledge hidden in chaotic feature maps of the corresponding conv-layer. Because each filter in the conv-layer may potentially represent multiple parts of the object, we use graph nodes to represent patterns of all candidate parts. A graph edge connects two nodes in adjacent layers to encode co-activation logics and spatial relationships between them.

Note that we do **not** fix the location of each pattern (node) to a certain neural unit of a conv-layer's output. Instead, given different input images, a part pattern may appear on various positions of a filter's feature maps[1]. For example, the horse face pattern and the horse ear pattern in Fig. 1 can appear on different positions of different images, as long as they are co-activated and keep certain spatial relationships.

- **Disentangling object parts from a single filter:** As shown in Fig. 1, each filter in a conv-layer may be activated

[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.

by different object parts (*e.g.* the filter's feature map[1] may be activated by both the head and the neck of a horse). To clarify the knowledge representation, we hope to disentangle patterns of different object parts from the same filter in an unsupervised manner, which presents a big challenge for state-of-the-art algorithms.

In this study, we propose a simple yet effective method to automatically discover object parts from a filter's feature maps **without** ground-truth part annotations. In this way, we can filter out noisy activations from feature maps, and we ensure that each graph node consistently represents the same object part among different images.

Given a testing image to the CNN, the explanatory graph can tell 1) whether a node (part) is triggered and 2) the location of the part on the feature map.

- **Graph nodes with high transferability:** Just like a dictionary, the explanatory graph provides off-the-shelf patterns for object parts, which enables a probability of transferring knowledge from conv-layers to other tasks. Considering that all filters in the CNN are learned using numerous images, we can regard each graph node as a detector that has been sophisticatedly learned to detect a part among thousands of images. Compared to chaotic feature maps of conv-layers, our explanatory graph is a more concise and meaningful representation of the CNN knowledge.

To prove the above assertions, we learn explanatory graphs for different CNNs (including the VGG-16, residual networks, and the encoder of a VAE-GAN) and analyze the graphs from different perspectives as follows.

Visualization & reconstruction: Patterns in graph nodes can be directly visualized in two ways. First, for each graph node, we list object parts that trigger strong node activations. Second, we use activation states of graph nodes to reconstruct image regions related to the nodes.

Examining part interpretability of graph nodes: (Bau et al. 2017) defined different types of interpretability for a CNN. In this study, we evaluate the part-level interpretability of the graph nodes. *I.e.* given an explanatory graph, we check whether a node consistently represents the same part semantics among different objects. We follow ideas of (Bau et al. 2017; Zhou et al. 2015) to measure the part interpretability of each node.

Examining location instability of graph nodes: Besides the part interpretability, we also define a new metric, namely location instability, to evaluate the clarity of the semantic meaning of each node in the explanatory graph. We assume that if a graph node consistently represents the same object part, then the distance between the inferred part and some ground-truth semantic parts of the object should not change a lot among different images.

Testing transferability: We associate graph nodes with explicit part names for multi-shot part localization. The superior performance of our method shows the good transferability of our graph nodes.

In experiments, we demonstrate both the representation clarity and the high transferability of the explanatory graph.

Contributions of this paper are summarized as follows.

1) In this paper, we, for the first time, propose a simple yet effective method to clarify the chaotic knowledge hidden inside a pre-trained CNN and to summarize such a deep knowledge hierarchy using an explanatory graph. The graph disentangles part patterns from each filter of the CNN. Experiments show that each graph node consistently represents the same object part among different images.

2) Our method can be applied to different CNNs, *e.g.* VGGs, residual networks, and the encoder of a VAE-GAN.

3) The mined patterns have good transferability, especially in multi-shot part localization. Although our patterns were pre-trained without part annotations, our transfer-learning-based part localization outperformed approaches that learned part representations with part annotations.

Related work

Semantics in CNNs

The interpretability and the discrimination power are two crucial aspects of a CNN (Bau et al. 2017). In recent years, different methods are developed to explore the semantics hidden inside a CNN. Many statistical methods (Szegedy et al. 2014; Yosinski et al. 2014; Aubry and Russell 2015) have been proposed to analyze the characteristics of CNN features. In particular, (Zhang, Wang, and Zhu 2018) has demonstrated that in spite of the good classification performance, a CNN may encode biased knowledge representations due to dataset bias. Instead, 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, we need methods to visualize the knowledge hierarchy hidden inside a CNN.

Visualization & interpretability of CNN filters: Visualization of filters in a CNN is the most direct way of exploring the pattern hidden inside a neural unit. Up-convolutional nets (Dosovitskiy and Brox 2016) were developed to invert feature maps to images. Comparatively, gradient-based visualization (Zeiler and Fergus 2014; Mahendran and Vedaldi 2015; Simonyan, Vedaldi, and Zisserman 2013) showed the appearance that maximized the score of a given unit, which is more close to the spirit of understanding CNN knowledge. Furthermore, Bau *et al.* (Bau et al. 2017) defined and analyzed the interpretability of each filter.

Although these studies achieved clear visualization results, theoretically, gradient-based visualization methods visualize one of the local minimums contained in a high-layer filter. *I.e.* when a filter represents multiple patterns, these methods selectively illustrated one of the patterns; otherwise, the visualization result will be chaotic. Similarly, (Bau et al. 2017) selectively analyzed the semantics among the highest 0.5% activations of each filter. In contrast, our method provides a solution to explaining both strong and weak activations of each filter and discovering all possible patterns from each filter.

Pattern retrieval: Some studies go beyond passive visualization and actively retrieve units from CNNs for different applications. Like middle-level feature extraction (Singh, Gupta, and Efros 2012), pattern retrieval mainly learns mid-level representations of CNN knowledge. Zhou *et al.* (Zhou et al. 2015; 2016) selected units from feature maps to describe "scenes". Simon *et al.* discovered

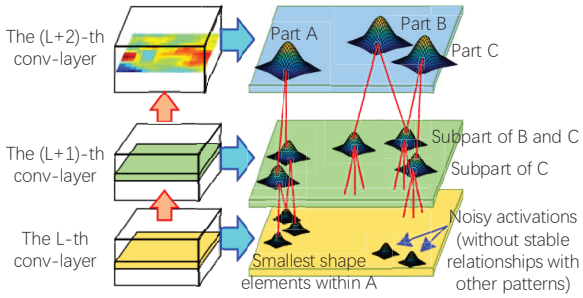

Figure 2: Spatial and co-activation relationships between part patterns in the explanatory graph. High-layer patterns filter out noises and disentangle low-layer patterns. From another perspective, we can regard low-layer patterns as components of high-layer patterns.

objects from feature maps of unlabeled images (Simon and Rodner 2015), and selected a filter to describe each part in a supervised fashion (Simon, Rodner, and Denzler 2014). 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 patterns. (Zhang et al. 2017a; 2017c; 2017b) extracted certain neurons from a filter's feature map to describe an object part in a weakly-supervised manner (*e.g.* learning from active question answering and human interactions).

In this study, the explanatory graph disentangles patterns different parts in the CNN without a need of part annotations. Compared to raw feature maps, patterns in graph nodes are more interpretable.

Weakly-supervised knowledge transferring

Knowledge transferring ideas have been widely used in deep learning. Typical research includes end-to-end fine-tuning and transferring CNN knowledge between different categories (Yosinski et al. 2014) or different datasets (Ganin and Lempitsky 2015). In contrast, we believe that a transparent representation of part knowledge will create a new possibility of transferring part knowledge to other applications. Therefore, we build an explanatory graph to represent part patterns hidden inside a CNN, which enables transfer part patterns to other tasks. Experiments have demonstrated the efficiency of our method in multi-shot part localization.

Algorithm

Intuitive understanding of the pattern hierarchy

As shown in Fig. 2, the feature map of a filter can usually be activated by different object parts in various locations. Let us assume that a feature map is activated with N peaks. Some peaks represent common parts of the object, and we call such activation peaks *part patterns*. Whereas, other peaks may correspond to background noises.

Our task is to discover activation peaks of part patterns out of noisy peaks from a filter's feature map. We assume that if a peak corresponds to an object part, then some patterns of other filters must be activated in similar map positions; vice versa. These patterns represent sub-regions of the same part and keep certain spatial relationships. Thus, in the explanatory graph, we connect each pattern in a low conv-layer to some patterns in the neighboring upper conv-layer. We mine part patterns layer by layer. Given patterns mined from the upper conv-layer, we select activation peaks, which keep stable spatial relationships with specific upper-layer patterns among different images, as part patterns in the current conv-layer.

As shown in Fig. 2, patterns in high conv-layers usually represent large-scale object parts. Whereas, patterns in low conv-layers mainly describes relatively simple shapes, which are less distinguishable in semantics. We use high-layer patterns to filter out noises and disentangle low-layer patterns. From another perspective, we can regard low-layer patterns as components of high-layer patterns.

Learning

Notations: We are given a CNN pre-trained using its own set of training samples $\mathbf{I}$. Let G denote the target explanatory graph. G contains several layers, which corresponds to conv-layers in the CNN. We disentangles the d-th filter of the L-th conv-layer into $N_{L,d}$ different part patterns, which are modeled as a set of $N_{L,d}$ nodes in the L-th layer of G, denoted by Ω_L. $\Omega_{L,d} \subset \Omega_L$ denotes the node set for the d-th filter. Parameters of these nodes in the L-th layer are given as $\boldsymbol{\theta}_L$, which mainly encode spatial relationships between these nodes and the nodes in the $(L+1)$-th layer.

Given a training image $I \in \mathbf{I}$, the CNN generates a feature map[1] of the L-th conv-layer, denoted by $\mathbf{X}_L^I$. Then, for each node $V \in \Omega_{L,d}$, we can use the explanatory graph to infer whether V's part pattern appears on the d-th channel[1] of $\mathbf{X}_L^I$, as well as the position of the part pattern (if the pattern appears). We use $\mathbf{R}_L^I$ to represent position inference results for all nodes in the L-th layer.

Objective function: We build the explanatory graph in a top-down manner. Given all training samples $\mathbf{I}$, we first disentangle patterns from the top conv-layer of the CNN, and built the top graph layer. Then, we use inference results of the patterns/nodes on the top layer to help disentangle patterns from the neighboring lower conv-layer. In this way, the construction of G is implemented layer by layer. Given inference results for the $(L+1)$-th layer $\{\mathbf{R}_{L+1}^I\}_{I\in\mathbf{I}}$, we expect that all patterns to simultaneously 1) be well fit to $\mathbf{X}_L^I$ and 2) keep consistent spatial relationships with upper-layer patterns $\mathbf{R}_{L+1}^I$ among different images. The objective of learning for the L-th layer is given as

$$\text{argmax}_{\boldsymbol{\theta}_L} \prod\nolimits_{I\in\mathbf{I}} P(\mathbf{X}_L^I | \mathbf{R}_{L+1}^I, \boldsymbol{\theta}_L) \quad (1)$$

I.e. we learn node parameters $\boldsymbol{\theta}_L$ that best fit feature maps of training images.

Let us focus on a conv-layer's feature map $\mathbf{X}_L^I$ of image I. Without ambiguity, we ignore the superscript I to simplify notations in following paragraphs. We can regard $\mathbf{X}_L$ as a distribution of "neural activation entities." We consider the neural response of each unit $x \in \mathbf{X}_L$ as the number of "activation entities." In other words, each unit x localizes

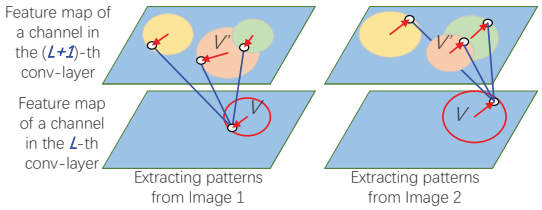

Figure 3: Related patterns V and V' keep similar spatial relationships among different images. Circle centers represent the prior pattern positions, *e.g.* μ_V and $\mu_{V'}$. Red arrows denote relative displacements between the inferred positions and prior positions, *e.g.* $\mathbf{p}_V - \mu_V$.

at the position of $\mathbf{p}_x$[2] in the d_x-th channel of $\mathbf{X}_L$. We use $F(x)=\beta\cdot\max\{f_x,0\}$ to denote the number of activation entities at the position $\mathbf{p}_x$, where f_x is the normalized response value of x; β is a constant.

Therefore, just like a Gaussian mixture model, we use all patterns in $\Omega_{L,d}$ as a mixture model to jointly explain the distribution of activation entities on the d-th channel of $\mathbf{X}_L$:

$$P(\mathbf{X}_L|\mathbf{R}_{L+1},\boldsymbol{\theta}_L)=\textstyle\prod_{x\in\mathbf{X}_L}P(\mathbf{p}_x|\mathbf{R}_{L+1},\boldsymbol{\theta}_L)^{F(x)} \quad (2)$$
$$=\textstyle\prod_{x\in\mathbf{X}_L}\Big\{\sum_{V\in\Omega_{L,d}\cup\{V_{\text{none}}\}}P(V)P(\mathbf{p}_x|V,\mathbf{R}_{L+1},\boldsymbol{\theta}_L)\Big\}_{d=d_x}^{F(x)}$$

where we consider each node $V\in\Omega_{L,d}$ as a hidden variable or an alternative component in the mixture model to describe activation entities. $P(V)=\frac{1}{N_{L,d}+1}$ is a constant prior probability. $P(\mathbf{p}_x|V,\mathbf{R}_{L+1},\boldsymbol{\theta}_L)$ measures the compatibility of using node V to describe an activation entity at $\mathbf{p}_x$. In addition, because noisy activations cannot be explained by any patterns, we add a dummy component V_{none} to the mixture model for noisy activations. Thus, the compatibility between V and $\mathbf{p}_x$ is computed based on spatial relationship between V and other nodes in G, which is roughly formulated as

$$P(\mathbf{p}_x|V,\mathbf{R}_{L+1},\boldsymbol{\theta}_L)=\begin{cases}\gamma\prod_{V'\in E_V}P(\mathbf{p}_x|\mathbf{p}_{V'},\boldsymbol{\theta}_L)^{\lambda} & V\in\Omega_{L,d_x}\\ \gamma\tau, & V=V_{\text{none}}\end{cases} \quad (3)$$

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

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

As shown in Fig. 3, an intuitive idea is that the relative displacement between V and V' should not change a lot among different images. Let $\mu_V\in\boldsymbol{\theta}_L$ and $\mu_{V'}\in\boldsymbol{\theta}_{L+1}$ denote the prior positions of V and V', respectively. Then, $\mathbf{p}_x-\mathbf{p}_{V'}$ will approximate to $\mu_V-\mu_{V'}$, if node V can well fit activation entities at $\mathbf{p}_x$. Therefore, given E_V and $\mathbf{R}_{L+1}$, we assume the spatial relationship between V and V' follows a Gaussian distribution in Eqn. 4, where $\mu_{V'\to V}=\mu_V-\mu_{V'}+\mathbf{p}_{V'}$ denotes the prior position of V given V'. $\sigma^2_{V'}$ denotes the variation, which can be estimated from data[3].

In this way, the core of learning is to determine an optimal set of neighboring patterns $E_V\in\boldsymbol{\theta}_L$ and estimate the prior position $\mu_V\in\boldsymbol{\theta}_L$. Note that our method only models the relative displacement $\mu_V-\mu_{V'}$.

Inference of pattern positions: Given the d-th filter's feature map, we simply assign node $V\in\Omega_{L,d}$ with a certain unit $\hat{x}=\text{argmax}_{x\in\mathbf{X}_L:d_x=d}S^I_{V\to x}$ on the feature map as the true inference of V, where $S^I_{V\to x}=F(x)P(\mathbf{p}_x|V,\mathbf{R}_{L+1},\boldsymbol{\theta}_L)$ denotes the score of assigning V to x. Accordingly, $\mathbf{p}_{V'}=$

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

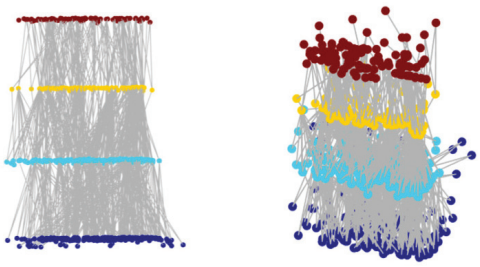

Figure 4: A four-layer explanatory graph. For clarity, we put all nodes of different filters in the same conv-layer on the same plan and only show 1% of the nodes with 10% of their edges from two perspectives.

[2]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.

[3]We can prove that for each $V\in\Omega_{L,d}$, $P(\mathbf{p}_x|V,\mathbf{R}_{L+1},\boldsymbol{\theta}_L)\propto\mathcal{N}(\mathbf{p}_x|\mu_V+\Delta_{I,V},\tilde{\sigma}^2_V)$, 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.

Figure 5: Image patches corresponding to different nodes in the explanatory graph.

$\mathbf{p}_{\hat{x}}$ represents the inferred position of V. In particular, in Eqn. (1), we define $\mathbf{R}_{L+1}=\{\mathbf{p}_{V'}\}_{V'\in\Omega_{L+1}}$.

Top-down EM-based Learning: For each node V, we need to learn the parameter $\mu_V \in \boldsymbol{\theta}_L$ and a set of patterns in the upper layer that are related to V, $E_V \in \boldsymbol{\theta}_L$. We learn the model in a top-down manner. We first learn nodes in the top-layer of G, and then learn for the neighboring lower layer. For the sub-graph in the L-th layer, we iteratively estimate parameters of μ_V and E_V for nodes in the sub-graph. We can use the Expectation-Maximization (EM) algorithm for learning. Please see Algorithm 1 for details.

Note that for each pattern V in the top conv-layer, we simply define $E_V = \{V_{\text{dummy}}\}$, In $\mathbf{R}_{L+1}$, $\mu_{V_{\text{dummy}}} = \mathbf{p}_{V_{\text{dummy}}} = \mathbf{0}$. V_{dummy} is a dummy node. Based on Eqns. (3) and (4), we obtain $P(\mathbf{p}_x|V, \mathbf{R}_{L+1}, \boldsymbol{\theta}_L)=\mathcal{N}(\mathbf{p}_x|\mu_V, \sigma_V^2)$.

Experiments

Overview of experiments

Four types of CNNs: To demonstrate the broad applicability of our method, we applied our method to four types of CNNs, *i.e.* the VGG-16 (Simonyan and Zisserman 2015), the 50-layer and 152-layer Residual Networks (He et al. 2016), and the encoder of the VAE-GAN (Larsen, Sønderby, and Winther 2016).

Three experiments and thirteen baselines: We designed three experiments to evaluate the explanatory graph. The first experiment is to visualize patterns in the graph. The second experiment is to evaluate the semantic interpretability of the part patterns, *i.e.* checking whether a pattern consistently represents the same object region among different images. We compared our patterns with three types of middle-level features and neural patterns. The third experiment is multi-shot learning for part localization, in order to test the transferability of patterns in the graph. In this experiment, we associated part patterns with explicit part names for part localization. We compared our method with ten baselines.

Three benchmark datasets: We built explanatory graphs to describe CNNs learned using a total of 37 animal categories in three datasets: the ILSVRC 2013 DET Animal-Part dataset (Zhang et al. 2017a), the CUB200-2011 dataset (Wah et al. 2011), and the Pascal VOC Part dataset (Chen et al. 2014). As discussed in (Chen et al. 2014; Zhang et al. 2017a), 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.

Implementation details

We first trained/fine-tuned a CNN using object images of a category, which were cropped using object bounding boxes. Then, we learned an explanatory graph to represent patterns of the category hidden inside the CNN. We set parameters $\tau=0.1$, $M=15$, $T=20$, and $\beta=1$.

VGG-16: Given a VGG-16 that was pre-trained using the 1.3M images in the ImageNet dataset (Deng et al. 2009), we fine-tuned all conv-layers of the VGG-16 using object images in a category. The loss for finetuning was that for classification between the target category and background images. In each VGG-16, there are 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 built a four-layer graph. We extracted $N_{L,d}$ patterns from the d-th filter of the L-th layer. We set $N_{L=1\text{ or }2,d}=40$ and $N_{L=3\text{ or }4,d}=20$.

Residual Networks: We chose two residual networks, *i.e.* the 50-layer and 152-layer ones. The finetuning 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 ouput, 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 $N_{L=1,d} = 40$, $N_{L=2,d} = 20$, and $N_{L=3,d}=10$.

VAE-GAN: For each category, we used the cropped object images in the category to train a VAE-GAN. We learned a three-layer graph based on the three conv-layers of the encoder of the VAE-GAN. We set $N_{L=1,d} = 52$, $N_{L=2,d} = 26$, and $N_{L=3,d}=13$.

Experiment 1: pattern visualization

Given an explanatory graph for a VGG-16 network, we visualize its structure in Fig. 4. Part patterns in the graph are visualized in the following three ways.

Top-ranked patches: We performed pattern inference on all object images. For each image I, we extracted an images patch in the position of $\mathbf{p}_{\hat{x}_V}$[4] with a fixed scale of

[4]We projected the unit to the image to compute its position.

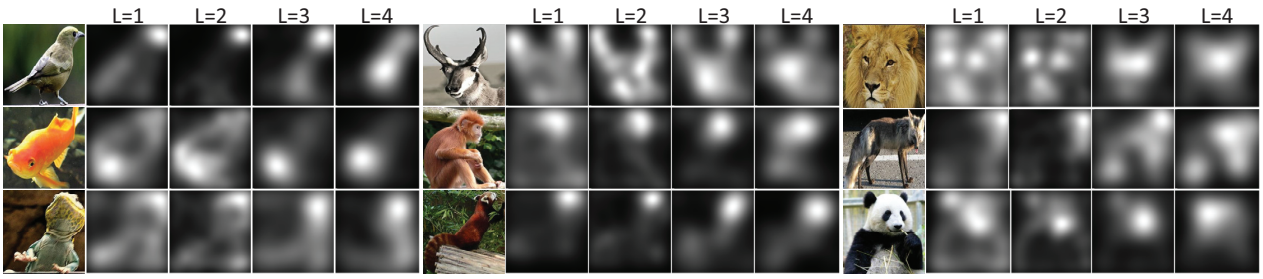

Figure 6: Heat maps of patterns. We use a heat map to visualize the spatial distribution of the top-50% patterns in the L-th layer of the explanatory graph with the highest inference scores.

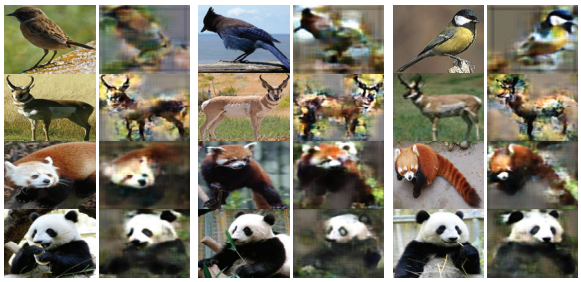

Figure 7: Image synthesis result (right) based on patterns activated on an image (left). The explanatory graph only encodes major part patterns hidden in conv-layers, rather than compress a CNN without information loss. Synthesis results demonstrate that the patterns are automatically learned to represent foreground appearance, and ignore background noises and trivial details of objects.

$70\,pixels \times 70\,pixels$ to represent pattern V. Fig. 5 shows a pattern's image patches that had highest inference scores.

Heat maps of patterns: Given a cropped object image I, we used the explanatory graph to infer its patterns on image I, and drew heat maps to show the spatial distribution of the inferred patterns. We drew a heat map for each layer L of the graph. Given inference results of patterns in the L-th layer, we drew each pattern $V \in \Omega_L$ as a weighted Gaussian distribution $\alpha \cdot \mathcal{N}(\mu = \mathbf{p}_V, \sigma_V^2)$[4] on the heat map, where $\alpha = S^I_{V \to \hat{x}}$. Please see Fig. 6 for heat maps of the top-50% patterns with the highest scores of $S^I_{V \to \hat{x}}$.

Pattern-based image synthesis: We used the up-convolutional network (Dosovitskiy and Brox 2016) to visualize the learned patterns. Up-convolutional networks were originally trained for image reconstruction. In this study, given an image's feature maps corresponding to the second graph layer, we estimated the appearance of the original image. Given an object image I, we used the explanatory graph for pattern inference, *i.e.* assigning each pattern V with a certain neural unit $\hat{x}_V$ as its position inference[4]. We considered the top-10% patterns 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 patterns, from feature maps (setting these responses to zero). We then used (Dosovitskiy and Brox 2016) to synthesize the appearance corresponding to the modified feature maps. We regard image synthesis in Fig. 7 as the visualization of the inferred patterns.

Experiment 2: semantic interpretability of patterns

In this experiment, we tested whether each pattern in an explanatory graph consistently represented the same object region among different images. We learned four explanatory graphs for a VGG-16 network, two residual networks, and a VAE-GAN that were trained/fine-tuned using the CUB200-2011 dataset (Wah et al. 2011). We used two methods to evaluate the semantic interpretability of patterns, as follows.

Part interpretability of patterns: We mainly extracted patterns from high conv-layers, and as discussed in (Bau et al. 2017), high conv-layers contain large-scale part patterns. We were inspired by Zhou *et al.* (Zhou et al. 2015) and measured the interpretability of part patterns. For the pattern of a given node V, we used people to manually evaluate the pattern's interpretability. When we used V to make inferences 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 require the K highest inference scores $S^{I_i}_V$ on images $\{I_1, \ldots, I_k\}$ to take about 30% of the inference energy, *i.e.* $\sum_{i=1}^{K} S^{I_i}_V = 0.3 \sum_{i \in \mathbf{I}} S^I_V$ (we use this equation to compute K). As shown in Fig.8, we asked human raters how many inference results among the top K described the same object part, in order to compute the purity of part semantics of pattern V.

The table in Fig. 8(top-left) shows the semantic purity of the patterns in the second layer of the graph. Let the second graph layer correspond to the L-th conv-layer with D filters. Like in (Zhou et al. 2015), the *raw filter maps* baseline used activated neurons 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 a part detection. Like our method, the two baselines only visualized top-K' part inferences (the K' feature maps' neural activations took 30% of activation energies among all images). We back-propagated the center of the receptive field of each neural activation to the image plane and simply used a fixed radius to draw the image region corresponding to each neural activation. Fig. 8 compares the image region corresponding to each node in the explanatory graph and image regions corresponding to feature maps of each filter. Our graph nodes encoded much more meaningful part representations than raw filters.

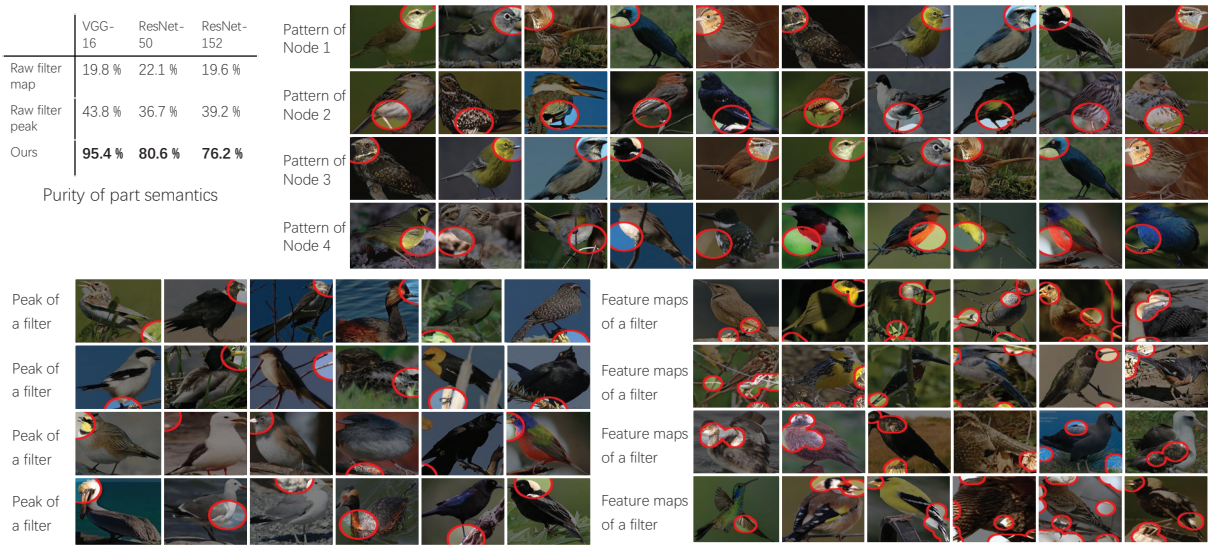

Figure 8: Purity of part semantics. We draw image regions corresponding to each node in an explanatory graph and image regions corresponding to each pattern learned by other methods (we show some examples on the right). We use human users to annotate the semantic purity of each node/pattern. Cyan boxes show inference results that do not describe the common part.

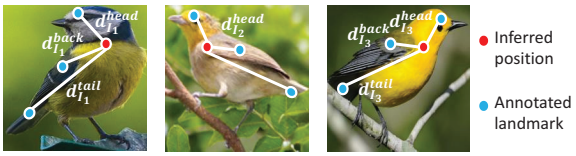

Figure 9: Notation for the computation of location instability.

	ResNet-50	ResNet-152	VGG-16	VAE-GAN
Raw filter (Zhou et al. 2015)	0.1328	0.1346	0.1398	0.1944
Ours	**0.0848**	**0.0858**	**0.0638**	**0.1066**
(Singh, Gupta, and Efros 2012)	0.1341			
(Simon, Rodner, and Denzler 2014)	0.2291			

Table 1: Location instability of patterns.

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

Location instability of inference positions: We also defined the location instability of inference positions for each pattern as an alternative evaluation of pattern interpretability. We assumed that if a pattern was always triggered by the same object part through different images, then the distance between the pattern's inference position and a ground-truth landmark of the object part should not change a lot among various images.

As shown in Fig. 9, for each testing image I, we computed the distances between the inferred position of V and ground-truth landmark positions of *head*, *back*, and *tail* parts, denoted by d_I^{head}, d_I^{back}, and d_I^{tail}. We normalized these distances by the diagonal length of input images. Then, we computed $(\sqrt{var(d_I^{\text{head}})} + \sqrt{var(d_I^{\text{back}})} + \sqrt{var(d_I^{\text{tail}})})/3$ as the location instability of the node for evaluation, where $var(d_I^{\text{head}})$ denotes the variation of d_I^{head} among different images.

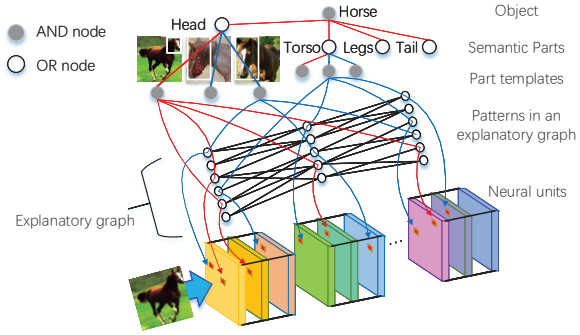

Figure 10: And-Or graph for semantic object parts. The AOG encodes a four-layer hierarchy for each semantic part, *i.e.* the semantic part (OR node), part templates (AND node), latent part patterns (OR nodes, those from the explanatory graph), and neural units (terminal nodes). In the AOG, the OR node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

Given an explanatory graph, we compared its location instability with three baselines. In the first baseline, we treated each filter in a CNN as a detector of a certain pattern. Thus, given the feature map of a filter (after the ReLu operation), we used the method of (Zhou et al. 2015) to localize the unit with the highest response value as the pattern position. The other two baselines were typical methods to extract middle-

	Method	obj.-box fine-tune	
no-RL	SS-DPM-Part (Azizpour and Laptev 2012)	N	0.3469
	PL-DPM-Part (Li et al. 2013)	N	0.3412
	Part-Graph (Chen et al. 2014)	N	0.4889
unsup[5]-RL	CNN-PDD (Simon, Rodner, and Denzler 2014)	N	0.2333
	CNN-PDD-ft (Simon, Rodner, and Denzler 2014)	Y	0.3269
	Ours	Y	**0.0862**
sup-RL	fc7+linearSVM	Y	0.3120
	fc7+sp+linearSVM	Y	0.3120
	Fast-RCNN (1 ft) (Girshick 2015)	N	0.4517
	Fast-RCNN (2 fts) (Girshick 2015)	Y	0.4131

Table 2: Normalized distance of part localization on the CUB200-2011 dataset (Wah et al. 2011). The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

level features from images (Singh, Gupta, and Efros 2012) and extract patterns from CNNs (Simon, Rodner, and Denzler 2014), respectively. For each baseline, we chose the top-500 patterns (*i.e.* 500 nodes with top scores in our explanatory graph, 500 filters with strongest activations in the CNN, and the top-500 middle-level features). For each pattern, we selected position inferences on the top-20 images with highest scores to compute the instability of its inferred positions. Table 1 compares the location instability of the patterns learned by different baselines, and our method exhibited significantly lower location instability.

Experiment 3: multi-shot part localization

And-Or graph for semantic parts The explanatory graph makes it plausible to transfer middle-layer patterns from CNNs to semantic object parts. In order to test the transferability of patterns, we build an additional And-Or graph (AOG) to associate certain implicit patterns with an explicit part name, in the scenario of multi-shot learning. We used the AOG to localize semantic parts of objects for evaluation. The structure of the AOG is inspired by (Zhang, Wu, and Zhu 2017), and the learning of the AOG was originally proposed in (Zhang et al. 2017a). We briefly introduce the AOG in (Zhang et al. 2017a) as follows.

As shown in Fig. 10, like the hierarchical model in (Li and Hua 2015), the AOG encodes a four-layer hierarchy for each semantic part, *i.e.* the semantic part (OR node), part templates (AND node), latent patterns (OR nodes, those from the explanatory graph), and neural units (terminal nodes). In the AOG, each OR node (*e.g.* a semantic part or a latent pattern) contains a list of alternative appearance (or deformation) candidates. Each AND node (*e.g.* a part template) uses a number of latent patterns to describe its compositional regions.

1) The OR node of a semantic part contains a total of m part templates to represent alternative appearance or pose candidates of the part. 2) Each part template (AND node) retrieve K patterns from the explanatory graph as children. These patterns describe compositional regions of the part. 3) Each latent pattern (OR node) has all units in its corresponding filter's feature map as children, which represent its deformation candidates on image I.

Experimental settings of three-shot learning We learned the explanatory graph based on a fine-tuned VGG-16 network and built the AOG following the scenario of multi-shot learning introduced in (Zhang et al. 2017a). For each category, we used three annotations of the head part to learn three head templates in the AOG. Such part annotations were offered by (Zhang et al. 2017a). To enable a fair comparison, all the object-box annotations and the three part annotations were equally provided to all baselines for learning.

We learned the explanatory graph based on a fine-tuned VGG-16 network (Simonyan and Zisserman 2015) and built the AOG following the scenario of multi-shot learning introduced in (Zhang et al. 2017a). For each category, we set three templates for the head part ($m = 3$), and used a single part-box annotation for each template. 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 Pascal VOC Part categories. Then, we used the AOGs to localize semantic parts on objects. Note that we used object images without part annotations to learn the explanatory graph and we used three part annotations provided by (Zhang et al. 2017a) to build the AOG. All these training samples were equally provided to all baselines for learning (besides part annotations, all baselines also used object annotations contained in the datasets for learning).

Baselines: We compared AOGs with a total of ten baselines in part localization. The baselines included 1) state-of-the-art algorithms for object detection (*i.e.* directly detecting target parts from objects), 2) graphical/part models for part localization, and 3) the methods selecting CNN patterns to describe object parts.

The first baseline was the standard fast-RCNN (Girshick 2015), 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 (Simon, Rodner, and Denzler 2014) 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 (Deng et al. 2009). Just like Fast-RCNN (2 ft), we extended (Simon, Rodner, and Denzler 2014) as the fourth baseline *CNN-PDD-ft*, which fine-tuned a VGG-16 network using object-box annotations before applying the technique of (Simon, Rodner, and Denzler 2014). The fifth and sixth baselines were DPM-related methods, *i.e.* the strongly supervised DPM (*SS-DPM-Part*) (Azizpour and Laptev 2012) and the technique in (Li et al. 2013) (*PL-DPM-Part*), respectively. Then, the seventh baseline, namely *Part-Graph*, used a graphical model for part localization (Chen et al. 2014). For weakly supervised learning, "simple" methods are usually insensitive to model over-fitting. Thus, we designed two 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 (Uijlings et al. 2013) to collect im-

		obj.-box fine-tune	bird	cat	cow	dog	horse	sheep	**Avg.**
no-RL	SS-DPM-Part (Azizpour and Laptev 2012)	N	0.356	0.270	0.264	0.242	0.262	0.286	0.280
no-RL	PL-DPM-Part (Li et al. 2013)	N	0.294	0.328	0.282	0.312	0.321	0.840	0.396
no-RL	Part-Graph (Chen et al. 2014)	N	0.360	0.208	0.263	0.205	0.386	0.500	0.320
unsup[5]-RL	CNN-PDD (Simon, Rodner, and Denzler 2014)	N	0.301	0.246	0.220	0.248	0.292	0.254	0.260
unsup[5]-RL	CNN-PDD-ft (Simon, Rodner, and Denzler 2014)	Y	0.358	0.268	0.220	0.200	0.302	0.269	0.269
unsup[5]-RL	**Ours**	Y	**0.162**	**0.130**	0.258	**0.137**	**0.181**	**0.192**	**0.177**
sup-RL	fc7+linearSVM	Y	0.247	0.174	0.251	0.217	0.261	0.317	0.244
sup-RL	fc7+sp+linearSVM	Y	0.247	0.174	**0.249**	0.217	0.261	0.317	0.244
sup-RL	Fast-RCNN (1 ft) (Girshick 2015)	N	0.324	0.324	0.325	0.272	0.347	0.314	0.318
sup-RL	Fast-RCNN (2 fts) (Girshick 2015)	Y	0.350	0.295	0.255	0.293	0.367	0.260	0.303

Table 3: Normalized distance of part localization on the Pascal VOC Part dataset (Chen et al. 2014). The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

		obj.-box fine-tune	gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion	tiger	bear	rabb.	hams.	squi.
no-RL	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	0.144	0.260	0.272	0.178	0.261
no-RL	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	0.136	0.323	0.228	0.186	0.281
no-RL	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	0.127	0.224	0.188	0.131	0.208
unsup[5]-RL	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	0.160	0.223	0.266	0.156	0.291
unsup[5]-RL	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	0.256	0.178	0.167	0.286	0.237
unsup[5]-RL	**Ours**	Y	**0.090**	**0.091**	**0.095**	0.167	**0.124**	**0.084**	**0.155**	0.147	**0.081**	**0.129**	**0.074**	**0.102**	**0.121**	**0.087**	**0.097**	**0.095**
sup-RL	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	0.163	0.122	0.139	0.110	0.262
sup-RL	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	0.163	0.122	0.139	0.110	0.262
sup-RL	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	0.260	0.317	0.255	0.255	0.169
sup-RL	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	0.160	0.230	0.230	0.178	0.205
			horse	zebra	swine	hippo	catt.	sheep	ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.		**Avg.**
no-RL	SS-DPM-Part	N	0.246	**0.206**	0.240	0.234	0.246	0.205	0.224	0.277	0.253	0.283	0.206	0.219	0.256	0.129		0.242
no-RL	PL-DPM-Part	N	0.322	0.267	0.297	0.273	0.271	0.413	0.337	0.261	0.286	0.295	0.187	0.264	0.204	0.505		0.284
no-RL	Part-Graph	N	0.296	0.315	0.306	0.378	0.333	0.230	0.216	0.317	0.227	0.341	0.159	0.294	0.276	0.094		0.257
unsup[5]-RL	CNN-PDD	N	0.261	0.266	**0.189**	0.192	0.201	0.244	0.208	0.193	0.174	0.299	0.236	0.214	0.222	0.179		0.225
unsup[5]-RL	CNN-PDD-ft	Y	0.310	0.321	0.216	0.257	0.220	0.179	0.229	0.253	0.198	0.308	0.273	0.189	0.208	0.275		0.240
unsup[5]-RL	**Ours**	Y	**0.189**	0.212	0.212	0.151	**0.185**	**0.124**	**0.093**	**0.120**	**0.102**	**0.188**	**0.086**	0.174	**0.104**	**0.073**		**0.125**
sup-RL	fc7+linearSVM	Y	0.205	0.258	0.201	0.140	0.256	0.236	0.164	0.190	0.140	0.252	0.256	0.176	0.215	0.116		0.184
sup-RL	fc7+sp+linearSVM	Y	0.205	0.258	0.201	**0.140**	0.256	0.236	0.164	0.190	0.140	0.250	0.256	0.176	0.215	0.116		0.184
sup-RL	Fast-RCNN (1 ft)	N	0.374	0.322	0.285	0.265	0.320	0.277	0.255	0.351	0.340	0.324	0.334	0.256	0.336	0.274		0.299
sup-RL	Fast-RCNN (2 fts)	Y	0.346	0.303	0.212	0.223	0.228	0.195	0.175	0.247	0.280	0.319	0.193	**0.125**	0.213	0.160		0.246

Table 4: Normalized distance of part localization on the ILSVRC 2013 DET Animal-Part dataset (Zhang et al. 2017a). The second column indicates whether the baseline used all object-box annotations in the category to fine-tune a CNN.

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

Table 5: Normalized distance of part localization. We compared supervised and unsupervised mining of part patterns.

age patches, and used the VGG-16 network to extract *fc7* features from these patches. The baseline *fc7+linearSVM* used a linear SVM to detect the target part. The other baseline *fc7+sp+linearSVM* 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 part patterns from CNNs (Zhang et al. 2017a), namely *supervised-AOG*. Unlike our method (unsupervised), *supervised-AOG* used part annotations to extract part patterns.

Comparisons: To enable a fair comparison, we classify all baselines into three groups, *i.e.* no representation learning (no-RL), unsupervised representation learning (unsup-RL)[5], and supervised representation learning (sup-RL). The No-RL group includes conventional methods without using deep features, such as SS-DPM-Part, PL-DPM-Part, and Part-Graph. Sup-RL methods are Fast-RCNN (1 ft), Fast-RCNN (2 ft), CNN-PDD, CNN-PDD-ft, supervised-AOG, fc7+linearSVM, and fc7+sp+linearSVM. Fast-RCNN methods used part annotations to learn features. Supervised-AOG used part annotations to select filters from CNNs to localize parts. Unsup-RL methods include CNN-PDD, CNN-PDD-ft, and our method. These methods did not use part annotations, and only used object boxes for learning/selection.

We use the normalized distance to evaluate localization accuracy, which has been used in (Zhang et al. 2017a; Simon, Rodner, and Denzler 2014) as a standard metric. Tables 2, 3, and 4 show part-localization results on the CUB200-2011 dataset (Wah et al. 2011), the Pascal VOC Part dataset (Chen et al. 2014), and the ILSVRC 2013 DET Animal-Part dataset (Zhang et al. 2017a), respectively. Table 5 compares the unsupervised and supervised learning of neural patterns. In the experiment, the AOG outperformed all baselines, even methods that learned part features in a supervised manner.

[5]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.

Conclusion and discussions

In this paper, we proposed a simple yet effective method to learn an explanatory graph that reveals knowledge hierarchy inside conv-layers of a pre-trained CNN (*e.g.* a VGG-16, a residual network, or a VAE-GAN). We regard the graph as a concise and meaningful representation, which 1) filters out noisy activations, 2) disentangles reliable part patterns from each filter of the CNN, and 3) encodes co-activation logics and spatial relationships between patterns. Experiments showed that our patterns had significantly higher stability than baselines.

The explanatory graph's transparent representation makes it plausible to transfer CNN patterns to object parts. Part-localization experiments well demonstrated the good transferability. Our method even outperformed supervised learning of part representations. Nevertheless, the explanatory graph is still a rough representation of the CNN, rather than an accurate reconstruction of the CNN knowledge.

Acknowledgement

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

References

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

Azizpour, H., and Laptev, I. 2012. Object detection using strongly-supervised deformable part models. *In ECCV*.

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*.

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

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. *In CVPR*.

Dosovitskiy, A., and Brox, T. 2016. Inverting visual representations with convolutional networks. *In CVPR*.

Ganin, Y., and Lempitsky, V. 2015. Unsupervised domain adaptation in backpropagation. *In ICML*.

Girshick, R. 2015. Fast r-cnn. *In ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *In CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. *In NIPS*.

Larsen, A. B. L.; Sønderby, S. K.; and Winther, O. 2016. Autoencoding beyond pixels using a learned similarity metric. *In ICML*.

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

Li, H., and Hua, G. 2015. Hierarchical-pep model for real-world face recognition. *In CVPR*.

Li, B.; Hu, W.; Wu, T.; and Zhu, S.-C. 2013. Modeling occlusion by discriminative and-or structures. *In ICCV*.

Li, H.; Lin, Z.; Brandt, J.; Shen, X.; and Hua, G. 2015. A convolutional neural network cascade for face detection. *In CVPR*.

Mahendran, A., and Vedaldi, A. 2015. Understanding deep image representations by inverting them. *In CVPR*.

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

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

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

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

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

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

Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; and Smeulders, A. W. M. 2013. Selective search for object recognition. *In IJCV* 104(2):154–171.

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

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

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. *In ECCV*.

Zhang, Q.; Cao, R.; Wu, Y. N.; and Zhu, S.-C. 2017a. Growing interpretable part graphs on convnets via multi-shot learning. *In AAAI*.

Zhang, Q.; Cao, R.; Zhang, S.; Edmonds, M.; Wu, Y.; and Zhu, S.-C. 2017b. Interactively transferring cnn patterns for part localization. *In arXiv:1708.01783*.

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

Zhang, Q.; Wang, W.; and Zhu, S.-C. 2018. Examing cnn representations with respect to dataset bias. *In AAAI*.

Zhang, Q.; Wu, Y.; and Zhu, S.-C. 2017. A cost-sensitive visual question-answer framework for mining a deep and-or object semantics from web images. *In arXiv:1708.03911*.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2015. Object detectors emerge in deep scene cnns. *In ICRL*.

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