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 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. Each graph node represents a part pattern, and graph edges encode co-activation relationships and spatial relationships between patterns. 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 among different images. We transfer part patterns in the explanatory graph to part localization, and our method greatly outperforms other approaches.

1. Introduction

Convolutional neural networks (CNNs) [12, 10, 9, 14] 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 contain rich implicit patterns (e.g. patterns of object parts and/or textures). Therefore, instead of learning a more powerful network, in this research, we aim to provide a global view of how visual knowledge is organized in a pre-trained CNN, which presents great challenges. For example,

1 How many types of visual patterns are memorized by each convolutional filter of the CNN (here, a visual pattern may describe a certain object part or a texture)?
2 Which patterns are co-activated to describe an object part?
3 What is the spatial relationship between two patterns?

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.

In this study, given a pre-trained CNN, we propose to mine mid-level object part patterns from conv-layers, and 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 map¹ 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. For each conv-layer of the CNN, we extract a number of patterns from each filter’s feature map of the conv-layer. Then, we represent patterns in all filters as nodes in the corresponding layer of the graph. Graph edges connect nodes in neighboring layers to encode co-activation logics and spatial relationships between them.

Note that the location of each pattern (node) is not fixed to a certain neural unit of a conv-layer’s output. Instead, given different input images, a part pattern may appear on different positions of a filter’s feature maps¹. 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 by different object parts (e.g. the filter’s feature map¹ may
be activated by both the head and the neck of a horse). In or-
der to clarify the knowledge representation, we hope to dis-
entangle patterns of different object parts from the same fil-
ter in an unsupervised manner, which presents a great chal-
lenge 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 fea-
ture maps without ground-truth part annotations. Thus,
each node in the graph consistently represents the same 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 lo-
cation of the part on the feature map.

• With great transferability: Just like a dictionary, the
explanatory graph provides a number of off-the-shelf pat-
terns for object parts, which enables a probability of trans-
ferring knowledge from conv-layers to other tasks. Consid-
ering that all filters in the CNN are learned using numer-
ous 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.

In order to prove the above assertions, we learn explana-
tory graphs for different CNNs (including the VGG-16,
residual networks, and the encoder of a VAE-GAN) and
analyse the graphs from different perspectives.

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

Examining interpretability of graph nodes: [3] 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 [3,28] to measure the
interpretability of each node. We also evaluate the stability
of each node’s position inference.

Testing transferability: We associated graph nodes with
explicit part names for multi-shot part localization. The su-
perior performance of our method shows the good trans-
ferability of our graph nodes.

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

Contributions of this paper are summarized as follows.
1) In this paper, we, for the first time, propose to repre-
sent knowledge hierarchy hidden inside a pre-trained CNN
using an explanatory graph. The graph disentangles part
patterns from each filter of the CNN and organizes the pat-
terns in a deep hierarchy. Experiments show that the pattern
of each graph node consistently represents a certain object
part among different images.
2) Our method can be broadly applied to different CNNs,
e.g. the VGG-16, two residual networks, and the encoder of
a VAE-GAN.
3) The mined patterns have good transferability. We applied
our patterns to multi-shot part localization. Although the
patterns are learned without part annotations, our method
outperformed approaches that learned parts in a supervised
manner.

2. Related work

2.1. Semantics in CNNs

Interpretability and discriminability are two important
aspects of a model [3]. In recent years, different meth-
ods are developed to explore the semantics hidden inside
a CNN. Many statistical methods [21,24] have been pro-
posed to analyze the characteristics of CNN features.

Visualization of CNNs: Visualization of filters in a
CNN is the most direct way of exploring the pattern hidden
inside a neural unit. [24] showed the appearance
that maximized the score of a given unit. A method used to invert feature maps to images. How-
ever, passive visualization of a certain unit is not equivalent
to “active” mining of CNN patterns.

Pattern retrieval: Going beyond passive visualization,
some studies actively retrieve certain units from CNNs for
different applications. Like middle-level feature extrac-
tion [20] in images, pattern retrieval mainly learns mid-level
representations of CNN knowledge. Zhou et al. [26] selected units from feature maps to describe “scenes”. Si-
mon et al. discovered objects from feature maps of unla-
beled images [16], and selected a certain filter to describe
each semantic part in a supervised fashion [17]. However,
most methods simply assumed that each filter of a conv-
layer mainly encoded a single visual concept, and ignored
the case that a filter of high conv-layers encoded a mix-
ture of patterns. [25] extracted certain neurons from a fil-
ter’s feature map to describe an object part in a weakly-
supervised manner.

In this study, we propose to summarize the middle-layer
knowledge in a CNN using an explanatory graph in an unsu-
pervised manner. The explanatory graph disentangles pat-
tterns different parts in the CNN without a need of part an-
notations. Experiments show that compared to raw feature
maps, patterns in graph nodes are more interpretable.

2.2. 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 [23] and/or datasets [7]. In contrast, we believe
that a transparent representation of part knowledge will cre-
ate 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. Inspired by [23, 25], we add an hierarchical And-Or graph to further associate implicit part patterns with explicit part names. Experiments have demonstrated the effectiveness and efficiency of our method in multi-shot part localization.

3. Algorithm

3.1. Intuitive understanding of the pattern hierarchy

As shown in Fig. 2, the feature map of a filter is usually activated by different object parts in different locations. Let the feature map be 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 a number of 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 a number of patterns in the neighboring upper conv-layer. We mine part patterns layer by layer. Given patterns mined from the the upper conv-layer, we select activation peaks, which keep stable spatial relationships with certain 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.

3.2. Learning

Notations: We are given a CNN pre-trained using its own set of training samples $I$. 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 $\Omega_L$. $\Omega_{L,d} \subseteq \Omega_L$ denotes the node set for the $d$-th filter. Parameters of these nodes in the $L$-th layer are given as $\theta_{L,d}$, which mainly encode spatial relationships between these nodes and the nodes in the $(L + 1)$-th layer.

Given a training image $I \in I$, we use the explanatory graph to infer whether $V$’s part pattern appears on the $d$-th channel of $X_{L}^I$, as well as the position of the part pattern (if the pattern appears). We use $R_{L,d}^I$ 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 $I$, 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 ($R_{L+1,d}^I \forall d \in I$), we expect that all patterns to simultaneously 1) be well fit to $X_{L+1}^I$ and 2) keep consistent spatial relationships with upper-layer patterns $R_{L,d}^{I+1}$ among different images. The objective of learning for the $L$-th layer is given as

$$\arg\max_{\theta_L} \prod_{i \in I} P(X_{L+1}^i | R_{L+1}^i, \theta_L)$$  \hspace{1cm} (1)$$

I.e. we learn node parameters $\theta_L$ that best fit feature maps of training images.

Let us focus on a conv-layer’s feature map $X_{L}^I$ of image $I$. Without ambiguity, we ignore the superscript $I$ to simplify notations in following paragraphs. We can regard $X_{L}$ as a distribution of “neural activation entities.” We consider
the neural response of each unit \( x \in X_L \) as the number of “activation entities”\(^2\). In other words, each unit \( x \) localizes at the position of \( p_x \) in the \( d_x \)-th channel of \( X_L \). We use \( F(x) = \beta \max\{f_x, 0\} \) to denote the number of activation entities at the position \( p_x \), where \( f_x \) is the normalized response value of \( x \); \( \beta \) is a constant.

Therefore, just like a Gaussian mixture model, we use all patterns in \( \Omega_{x,d} \) as a mixture model to jointly explain the distribution of activation entities on the \( d \)-th channel of \( X_L \):

\[
P(X_L|R_{L+1}, \theta_L) = \prod_{d \in X_L} P(p_x|X_{L+1}, \theta_L)^{F(x)}
\]

\[
= \prod_{d \in X_L} \left\{ \sum_{V \in \Omega_{x,d}} P(V)p_x|V, R_{L+1}, \theta_L \right\}^{F(x)}
\]

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

\[
P(p_x|V, R_{L+1}, \theta_L) = \left\{ \begin{array}{ll}
\gamma \prod_{V \in \hat{E}_V} P(p_x|p_{V'}, \theta_L)^{\lambda V \in \Omega_{d_x}} V = V_{\text{none}} \\
\gamma \prod_{V \in \hat{E}_V} P(p_x|p_{V'}, \theta_L)^{\lambda V \in \Omega_{d_x}} V = V_{\text{none}}
\end{array} \right.
\]

\[
P(p_x|p_{V'}, \theta_L) = N(p_x|\mu_{V' \rightarrow V}, \sigma_2^{V'})
\]

In above equations, node \( V \) has a set of \( M \) neighboring patterns in the upper layer, denoted by \( E_V \in \theta_L \), which would be determined during the learning process. The overall compatibility \( P(p_x|V, R_{L+1}, \theta_L) \) is divided into the spatial compatibility between \( V \) and \( p_x \) and each neighboring node \( V' \), \( P(p_x|p_{V'}, \theta_L) \). \( V' \in \hat{E}_V \). \( p_{V'} \in R_{L+1} \) denotes the position reference of \( V' \), which have been provided. \( \lambda = \frac{1}{\gamma} \) is a constant for normalization. \( \gamma \) is a constant to roughly ensure \( \int P(p_x|V, R_{L+1}, \theta_L)dp_x = 1 \), 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 \( \mu_V \in \theta_L \) and \( \mu_{V'} \in \theta_{L+1} \) denote the prior positions of \( V \) and \( V' \), respectively. Then, \( p_x - p_{V'} \) will approximate to \( \mu_V - \mu_{V'} \), if node \( V \) will well fit activation entities at \( p_x \). Therefore, given \( E_V \) and \( R_{L+1} \), we assume the spatial relationship between \( V \) and \( V' \) follows a Gaussian distribution in Eqn. 4 where \( \mu_{V' \rightarrow V} = \mu_V - \mu_{V'} + \)

\(\text{Algorithm 1: Learning sub-graph in the } L \text{-th layer} \)

\(p_{V'}\) 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 \( E_V \in \theta_L \) and estimate the prior position \( \mu_V \in \theta_L \). 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 \( V \in \Omega_{x,d} \) with a certain unit \( \hat{x} = \arg\max_{x \in X_{L+1},d} S_{\hat{x}, L+1} \) on the feature map as the true inference of \( V \), where \( S_{\hat{x}, L+1} = F(x)P(p_x|V, R_{L+1}, \theta_L)^3 \) denotes the score of assigning \( V \) to \( \hat{x} \). Accordingly, \( p_{V'} = p_{\hat{x}} \) represents the inferred position of \( V \). In particular, in Eqn. 1, we define \( R_{L+1} = \left\{ p_{V'} \right\} V \in \Omega_{L+1} \).

**Top-down EM-based Learning:** For each node \( V \), we need to learn the parameter \( \mu_{V} \in \theta_{L} \) and a set of patterns in the upper layer that are related to \( V \), \( E_V \in \theta_L \). We learn the model in a top-down manner. We first learn nodes in the top-layer of \( G \), and then learn for the lower neighboring layer. For the sub-graph in the \( L \)-th layer, we iteratively estimate parameters of \( \mu_{V} \) 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:**

\(^2\)Please see supplementary materials for details.

\(^3\)In order to make unit positions in different conv-layers comparable with each other (e.g. \( \mu_{V' \rightarrow V} \) in Eq. 3, we normalize the position by projecting the position of unit \( x \) to the image plane. We define the coordinate \( p_x \) on the image plane, instead of on the feature-map plane.
In each category. The loss for finetuning was that for classification between the target category and background images. In each category, we used the cropped object images in a 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 $N_{L=3,d} = 13$.

### 4.3. Experiment 1: pattern visualization

Given an explanatory graph for a VGG-16 network, we visualize its structure in Fig.\ref{fig:exp1}. Part patterns in the graph are visualized in the following two ways.

- **Top-ranked patches**: We performed pattern inference on all object images. For each image $I$, we extracted an image patch in the position of $p\hat{V}$ with a fixed scale of $70\times70$ to represent pattern $\hat{V}$. Fig.\ref{fig:exp1} shows a pattern's image patches that had highest inference scores.

- **Pattern-based image synthesis**: We used the up-convolutional network \cite{kim2016accurate} 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 $\hat{V}$ with a certain neural unit $\hat{z}_V$ as its position inference. We considered the top-10% patterns with highest scores of $S_{V \rightarrow I}$ 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 \cite{kim2016accurate} to synthesize the appearance corresponding to the modified feature maps. We can regard results of image synthesis in Fig.\ref{fig:exp1} as the visualization of the inferred patterns.

### 4.4. Experiment 2: semantic stability 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 explicative graphs for each 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 \cite{deng2009imagenet}, we finetuned 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

\begin{enumerate}
\item We projected the unit to the image plane to compute its position.
\end{enumerate}
planetary graphs for a VGG-16 network, two residual networks, and a VAE-GAN that were trained/finetuned using the CUB200-2011 dataset [22]. We used two methods to evaluate the semantic stability of patterns, as follows.

Pattern interpretability: We mainly extract patterns from high conv-layers, and as discussed in [3], high conv-layers contain large-scale part patterns. We followed Zhou et al. [26] to measure the interpretability of part patterns. For the pattern of a given node \( V \), we used the Amazon Mechanical Turk (AMT) to evaluate the pattern’s interpretability. When we used \( V \) to do inferences among all images, let the top-\( K \) images \( \{I_1, \ldots, I_k\} \) with highest inference scores \( S^V_i \) take about 30% of the inference energy, i.e. \( \sum_{i=1}^{K} S^V_i = 0.3 \sum_{i \in I} S^V_i \). We regarded the top-\( K \) inference results as valid representations of \( V \). As shown in Fig. 6, 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. 6(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 [26], 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 filter’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). 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 \( \sum_{i \in I} S^V_i \).

Stability of inference positions: We also defined stability of inference positions for each pattern as an alternative evaluation of pattern interpretability. For each testing image \( I \), we computed the distances between the inferred position of \( V \) and ground-truth positions of head, back, and tail parts, denoted by \( d^\text{head} \), \( d^\text{back} \), and \( d^\text{tail} \). We normalized the distance by the diagonal length of input images. Then, we computed \( \sqrt{(\text{var}(d^\text{head}) + \text{var}(d^\text{back}) + \text{var}(d^\text{tail}))/3} \) as the metric (namely, standard deviation of relative pattern positions) to evaluate pattern stability, where \( \text{var}(d^\text{head}) \) denotes the variation of \( d^\text{head} \) among different images.

Given an explanatory graph, we compared its pattern stability 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 [26] to localize the unit with the highest response value as the pattern position. The other two baselines were typical methods to extract middle-level features from images [20] and extract patterns from CNNs [17], respectively. For each baseline, we selected the top-500 patterns, and for each pattern, we selected position inferences on the top-20 images with highest scores to compute the standard deviation of its position. Table 1 compares the stability of the patterns learned by different baselines, and our method exhibited significantly better performance.

|                  | ResNet-50 | ResNet-152 | VGG-16 | VAE-GAN |
|------------------|-----------|------------|--------|---------|
| Raw filter [26]  | 0.1328    | 0.1346     | 0.1398 | 0.1944  |
| Ours             | 0.0848    | 0.0858     | 0.0638 | 0.1066  |

Table 1. Standard deviations of relative pattern positions. We used relative distances between the pattern and the head, back, and tail parts to compute the standard deviation among different images.

4.5. Experiment 3: multi-shot part localization

In order to test the transferability of patterns, we build an And-Or graph (AOG) to associate certain part patterns with an explicit part name. We use the AOG to localize semantic parts of objects for evaluation. The learning of the AOG
was proposed in [25]. Fig. 8 shows the semantic hierarchy of the AOG.

We learned the explanatory graph based on a pre-finetuned VGG-16 network and built the AOG following the scenario of multi-shot learning introduced in [25]. For each category, we used three annotations of the head part to learn three head templates in the AOG. Such part annotations were equally provided to all baselines for learning. To enable a fair comparison, all the object-box annotations and the three part annotations were provided by [25]. To enable a fair comparison, all the object-box annotations and the three part annotations were equally provided to all baselines for learning. Please see [25] and supplementary materials for details of the AOG.

**Baselines**: We compared AOGs with a total of nine 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 Fast-RCNN (1 ft) method was a standard fast-RCNN that was learned using part annotations. Fast-RCNN (2 ft) finetuned the fast-RCNN using annotations of both objects and parts. CNN-PDD [17] selected certain filters of a pre-trained CNN to localize the target part. Compared to CNN-PDD, CNN-PDD-ft finetuned the CNN using object-box annotations as a preprocessing. We also compared our method with two DPM approaches, i.e. SS-DPM-Part [2] and PL-DPM-Part [13] and a graphical model for part localization, namely Part-Graph [4]. The baselines of fc7+linearSVM was proposed in [25] as a competing method. fc7+sp+linearSVM further added the spatial position of each part proposal as features for part detection.

**Comparisons**: To enable a fair comparison, we classify all baselines into three groups, i.e. no representation learning (no-RL), unsupervised representation learning (sup-RL) 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, fc7+linearSVM, and fc7+sp+linearSVM. Fast-RCNN methods used part annotations to learn features. Unsup-RL methods includes CNN-PDD, CNN-PDD-ft, and our method. These methods did not use part annotations, and only used object boxes for learning/selection.

Table 2 shows part-localization results on the ILSVRC 2013 DET Animal-Part dataset [25]. In the experiment, the AOG outperformed all baselines, even methods that learned part features in a supervised manner.

### 5. 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 provides a plausible way to transfer CNN patterns to object parts. Part-localization experiments well demonstrated the advantages of the AOG method over conventional methods.
| Method            | Obj-box | gold. | bird | frog | turt. | liza. | koala | lobs. | dog | fox | cat | lion | tiger | bear | rabb. | hams. | squ. |
|-------------------|---------|-------|------|------|-------|-------|-------|-------|-----|-----|-----|-----|------|------|-----|------|------|-----|
| SS-DPM-Part [3]   | 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 | 0.144 | 0.260 | 0.272 | 0.178 | 0.261 |
| PL-DPM-Part [13]  | 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 |
| Part-Graph [4]    | 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 |
| 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 |
| 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 |
| 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 |
| CNN-PDD [17]     | N       | 0.316 | 0.289 | 0.229 | 0.260 | 0.335 | 0.163 | 0.190 | 0.220 | 0.212 | 0.196 | 0.174 | 0.228 | 0.186 | 0.228 | 0.186 | 0.281 |
| CNN-PDD-r [17]   | N       | 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 | 0.268 | 0.237 |
| Fast-RCNN (1 ft) | N       | 0.261 | 0.365 | 0.285 | 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 |
| Fast-RCNN (2 ft) | N       | 0.340 | 0.351 | 0.388 | 0.327 | 0.411 | 0.199 | 0.330 | 0.368 | 0.206 | 0.170 | 0.144 | 0.160 | 0.790 | 0.178 | 0.205 | 0.205 |

Table 2. Normalized distance of part localization on the ILSVRC 2013 DET Animal-Part dataset. The second column indicates whether the baseline used all object-box annotations in the category to finetune a ConvNet (in fact, object-box annotations are more than part annotations).

References

[1] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. In ICCV, 2015.
[2] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In ECCV, 2012.
[3] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In CVPR, 2017.
[4] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In CVPR, 2014.
[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009.
[6] A. Dosovitisky and T. Brox. Inverting visual representations with convolutional networks. In CVPR, 2016.
[7] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation in backpropagation. In ICML, 2015.
[8] R. Girshick. Fast r-cnn. In ICCV, 2015.
[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
[10] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
[11] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In ICML, 2016.
[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, 1998.
[13] B. Li, W. Hu, T. Wu, and S.-C. Zhu. Modeling occlusion by discriminative and-or structures. In ICCV, 2013.
[14] H. Li, Z. Lin, J. Brandt, X. Shen, and G. Hua. A convolutional neural network cascade for face detection. In CVPR, 2015.
[15] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In CVPR, 2015.
[16] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In ICCV, 2015.
[17] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. In ACCV, 2014.
[18] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. In arXiv:1312.6034, 2013.
[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for object recognition. In ICLR, 2015.
[20] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In ECCV, 2012.
[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In ICLR, 2014.
[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. In arXiv:1312.6034, 2013.
[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In NIPS, 2014.
[24] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.
[25] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable graphs on convnets via multi-shot learning. In AAAI, 2016.

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

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

[28] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille. Max-margin and/or graph learning for parsing the human body. In CVPR, 2008.