Mining Interpretable AOG Representations from Convolutional Networks via Active Question Answering

Quanshi Zhang, Jie Ren, Ge Huang, Ruiming Cao, Ying Nian Wu, and Song-Chun Zhu

Abstract—In this paper, we present a method to mine object-part patterns from conv-layers of a pre-trained convolutional neural network (CNN). The mined object-part patterns are organized by an And-Or graph (AOG). This interpretable AOG representation consists of a four-layer semantic hierarchy, *i.e.* semantic parts, part templates, latent patterns, and neural units. The AOG associates each object part with certain neural units in feature maps of conv-layers. The AOG is constructed with very few annotations (*e.g.* 3–20) of object parts. We develop a question-answering (QA) method that uses active human-computer communications to mine patterns from a pre-trained CNN, in order to explain features in conv-layers incrementally. During the learning process, our QA method uses the current AOG for part localization. The QA method actively identifies objects, whose feature maps cannot be explained by the AOG. Then, our method asks people to annotate parts on the unexplained objects, and uses answers to discover CNN patterns corresponding to the newly labeled parts. In this way, our method gradually grows new branches and refines existing branches on the AOG to semanticize CNN representations. In experiments, our method exhibited a high learning efficiency. Our method used about $1/6$–$1/3$ of the part annotations for training, but achieved similar or better part-localization performance than fast-RCNN methods.

Index Terms—Convolutional Neural Networks, Hierarchical graphical model, Part localization

1 Introduction

Convolutional neural networks [23], [26], [29], [31] (CNNs) have achieved superior performance in many visual tasks, such as object detection and segmentation. However, in real-world applications, current neural networks still suffer from low interpretability of their middle-layer representations and data-hungry learning methods.

Thus, the objective of this study is to mine thousands of *latent patterns* from the mixed representations in conv-layers. Each latent pattern corresponds to a constituent region or a contextual region of an object part. We use an interpretable graphical model, namely an And-Or graph (AOG), to organize latent patterns hidden in conv-layers. The AOG maps implicit latent patterns to explicit object parts, thereby explaining the hierarchical representation of objects. We use very few (*e.g.* 3–20) part annotations to mine latent patterns and construct the AOG to ensure high learning efficiency.

We can use the AOG for part localization, *i.e.* inferring a parse graph within the AOG to localize object parts and their constituent regions. Another motivation is that the hierarchical structure of the AOG helps people understand feature representations of a CNN. Learning an interpretable model (*e.g.* the AOG) to explain CNN feature representations attracts an increasing attention in the scope of explainable AI.

As shown in Fig. 1, compared to ordinary CNN representations where each filter encodes a mixture of textures and parts (evaluated by [4]), we extract clear object-part representations from CNN features. Our learning method enables people to model objects or object parts on-the-fly without many human annotations, thereby ensuring broad applicability.

And-Or graph representations: As shown in Fig. 1, the AOG represents a semantic hierarchy on the top of conv-layers, which consists of four layers, *i.e.* the *semantic part*, *part templates*, *latent patterns*, to *CNN units*. In the AOG, AND nodes represent compositional regions of a part, and OR nodes represent a list of alternative template/deformation candidates for a local region.

- Layer 1: the top *semantic part* node is an OR node, whose children represent template candidates for the part.
- Layer 2: a *part template* in the second layer describes a certain part appearance with a specific pose, *e.g.* a black sheep head from a side view. A part template is an AND node, which uses its children latent patterns to encode its constituent regions.
- Layer 3: a *latent pattern* in the third layer represents a constituent region of a part (*e.g.* an eye in the head part) or a contextual region (*e.g.* the neck region *w.r.t.* the head). A latent pattern is an OR node, which naturally corresponds to a group

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

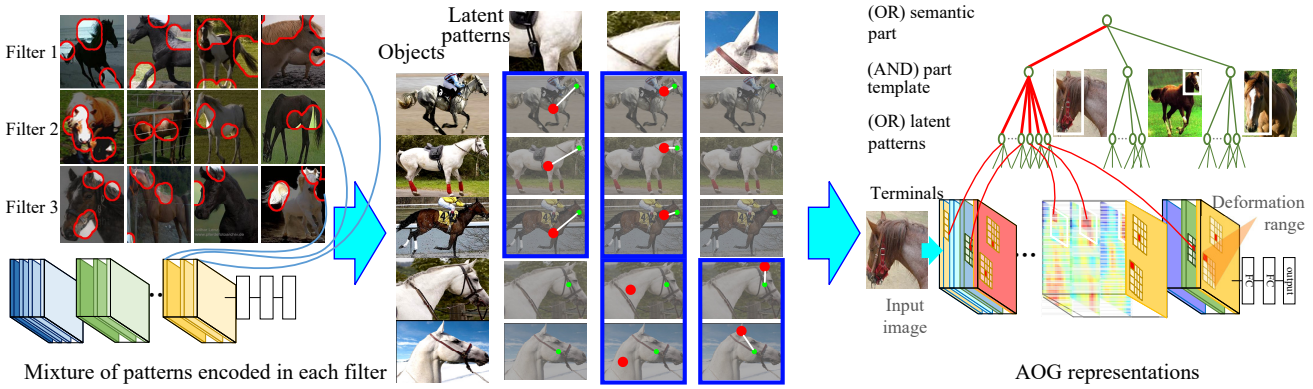


Fig. 1. Mining part-based AOG representations from CNN representations. (left) Each filter in a conv-layer usually encodes a mixture of patterns, which makes conv-layer representations a black box. The same filter may be activated by different parts of different objects. (middle) We disentangle CNN feature maps and mine latent patterns of object parts. White lines indicate the spatial relationship between a latent pattern's neural activation and the ground-truth position of an object part (head). (right) We grow an AOG on the CNN to associate CNN units with certain semantic parts (the horse head, here). Red lines in the AOG indicate a parse graph that associates certain CNN units with a semantic part.

of units within the feature map of a certain CNN filter. The latent pattern selects one of its children *CNN units* as the configuration of the geometric deformation.

- Layer 4: terminal nodes are *CNN units*, *i.e.* raw activation units on feature maps of a CNN filter.

For application, the AOG maps implicit latent patterns in raw CNN feature maps to explicit semantic parts in a bottom-up manner. As red lines in Fig. 1(right), the AOG infers a parse graph to localize object parts and their constituent regions for hierarchical object parsing. The AOG is interpretable and can be used for communications with human users.

Learning via active question-answering: We propose a new active learning strategy to build an AOG with a limited number of human annotations. As shown in Fig. 2, we use an active question-answering (QA) process to mine latent patterns from raw feature maps and gradually grow the AOG.

The input is a pre-trained CNN and its training samples (*i.e.* object images without part annotations). The QA method actively discovers the missing patterns in the current AOG and asks human users to label object parts for supervision.

In each step of the QA, we use the current AOG to localize a certain semantic part among all unannotated images. Our method actively identifies object images, which cannot fit well to the AOG. *I.e.* the current AOG cannot explain object parts in these images. Our method estimates the potential gain of asking about each of the unexplained objects, thereby determining an optimal sequence of questions for QA. Note that the QA is implemented based on pre-defined ontology, instead of using open-ended questions or answers. As in Fig. 2, the user is asked to provide five types of answers (*e.g.* labeling the correct part position when the AOG cannot accurately localize the part), in order to guide the growth of the AOG. Given each specific answer, our method may either refine the AOG branch of an existing part template or construct a new AOG branch for a new part template.

Based on human answers, we mine latent patterns for new AOG branches as follows. We require the new latent patterns

- to represent a region highly related to the annotated object parts,
- to frequently appear in unannotated objects,
- to consistently keep stable spatial relationships with other latent patterns.

Similar requirements were originally proposed in studies of pursuing AOGs, which mined hierarchical object structures from Gabor wavelets on edges [50] and HOG features [76]. We extend such ideas to feature maps of neural networks.

The active QA process mines object-part patterns from the CNN with limited human annotations. There are three mechanisms to ensure the learning stability with limited supervision.

- Instead of learning all representations from scratch, we transfer patterns in a pre-trained CNN to the target object part, which boosts the learning efficiency. Because the CNN has been trained using numerous images, latent patterns in the AOG are supposed to consistently describe the same part region of different object images, instead of over-fitting to part annotations obtained during the QA process. For example, we use the annotation of a specific tiger head to mine latent patterns. The mined patterns are not over-fitted to the head annotation, but represent generic appearances of different tiger heads. In this way, we use very few (*e.g.* 1–3) part annotations

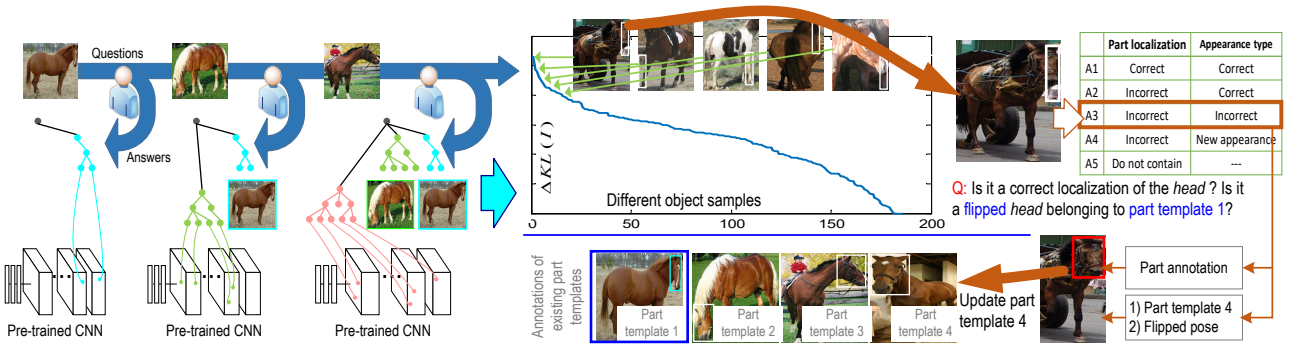


Fig. 2. Learning an AOG to explain a pre-trained CNN via active question-answering (QA). (left) We mine latent patterns of object parts from the CNN, and organize such patterns into a hierarchical AOG. During the learning process, our method automatically identifies objects whose parts cannot be well fit current part templates in the AOG, asks about the objects, and uses the answers to mine latent patterns and grow the AOG. (right) The learning algorithm sorts and selects objects for QA.

to extract latent patterns for each part template.

- It is important to maintain the generality of the pre-trained CNN during the learning procedure. *I.e.* we do not change/fine-tune the original convolutional weights within the CNN, when we grow new AOGs. This allows us to continuously learn new semantic parts from the same CNN, without the model drift.
- The active QA strategy reduces the excessive usage of the human labor of annotating object parts that have been well explained by the current AOG.

In addition, we use object-level annotations for pre-training, considering the following two facts: 1) Only a few datasets [8], [66] provide part annotations, and most benchmark datasets [16], [35], [45] mainly have annotations of object bounding boxes. 2) More crucially, real-world applications may focus on various object parts on-the-fly, and it is impractical to annotate a large number of parts for each specific task.

This paper makes the following three contributions. 1) From the perspective of object representations, we semanticize a pre-trained CNN by mining reliable latent patterns from noisy feature maps of the C-NN. We design an AOG to represent the semantic hierarchy inside conv-layers, which associates implicit neural patterns with explicit semantic parts. Learning an interpretable AOG model to explain CNN representations is important in the scope of explainable AI. 2) From the perspective of learning strategies, based on the clear semantic structure of the AOG, we present an active QA method to learn each part template of the object sequentially, thereby incrementally growing AOG branches on a CNN to enrich part representations in the AOG. 3) In experiments, our method exhibits superior performance to other baselines of part localization, in the scenario of learning models without many human annotations. For example, our methods with 11 part annotations outperformed fast-RCNNs with 60 annotations on the Pascal VOC Part dataset.

A preliminary version of this paper appeared in [71] and [72].

2 Related Work

In this section, we will discuss recent literature on interpreting deep neural networks and clarify relevant issues in the learning of compositional models.

2.1 Interpreting deep neural networks

Interpreting deep neural networks has attracted more and more attention in recent years. In general, network visualization, the diagnosis of feature representations, and the semanticization of feature representations represent three typical research directions.

Network visualization: Visualization of filters in a CNN is a direct way of exploring the pattern hidden inside a neural unit. Lots of visualization methods have been used in the literature. The gradient-based visualization [38], [53], [69] and the inversion-based method [13] represent two typical methodologies. Gradient-based visualization estimates the input image that maximizes the activation score of a neural unit. Up-convolutional nets [13] inverts feature maps of conv-layers to images. Unlike gradient-based methods, up-convolutional nets cannot mathematically ensure the visualization result reflects actual neural representations. In recent years, [41] provided a reliable tool to visualize filters in different conv-layers of a CNN. Unlike network visualization, our mining part representations from conv-layers is another choice to interpret CNN representations.

Diagnosis of feature representations: Going beyond "passive" visualization, some methods "actively" diagnose a pre-trained CNN to obtain an insightful understanding of CNN representations. The most typical idea is to estimate the image regions that

TABLE 1
Comparisons with studies of the network dissection.

	Ours	Gonzalez-Garcia *et al.* [22]	Bau *et al.* [4]
Modeling parts	√	√	√
Modeling textures		√	√
Modeling detailed structures of objects	√		
Can be used for inference	√		
Annotation cost	mainly 10–20 part annotations for each category	Part-level annotations of 10103 images in the VOC Part dataset [8] (containing 105 parts of 16 object classes)	Pixel-level annotations of 21663 images with parts in the Borden dataset [4]

directly contribute to the network output. Gradient-based methods [17], [44], [47] propagate gradients of feature maps *w.r.t.* the CNN loss back to the image to estimate attribution maps. In particular, Zhou *et al.* [77] proposed a method to accurately compute 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. The accurate estimation of the receptive field is crucial to understand a filter's representations.

Other studies include the analysis of semantic meanings [59], transferability [68], feature distributions [1], [37] of convolutional filters in intermediate layers. Network-attack methods [28], [57], [59] diagnosed network representations by computing adversarial samples for a CNN. In particular, influence functions [28] were proposed to compute adversarial samples, provide plausible ways to create training samples to attack the learning of CNNs, fix the training set, and further debug representations of a CNN. [30] discovered knowledge of blind spots (unknown patterns) of a pre-trained CNN in a weakly-supervised manner. Zhang *et al.* [75] developed a method to examine representations of conv-layers and automatically discover potential, biased representations of a CNN due to the dataset bias.

Semanticization of feature representations: Compared to the diagnosis of CNN representations, semanticization of CNN representations is closer to the spirit of building interpretable representations. Hu *et al.* [25] designed logic rules for network outputs, and used these rules to regularize neural networks and learn meaningful representations. However, this study has not obtained semantic representations in intermediate layers. Some studies extracted neural units with certain semantics from CNNs for different applications. Given feature maps of conv-layers, Zhou *et al.* [77], [78] extracted scene semantics. Simon *et al.* mined objects from feature maps of conv-layers [51], and learned explicit object parts [52].

Table 1 compares our work with previous studies of mining object parts from intermediate-layer features [4], [22].

Unlike the above research, we aim to explore the entire semantic hierarchy hidden inside conv-layers of a CNN. Because the AOG structure [50], [82] is suitable for representing the semantic hierarchy of objects, our method uses an AOG to represent the CNN. In our study, we use semantic-level QA to incrementally mine object parts from the CNN and grow the AOG. Such a "white-box" representation of the CNN also guided further active QA. With clear semantic structures, the AOG makes it easier to transfer CNN patterns to other part-based tasks.

The hierarchical representation of CNN patterns also boosts the capacity of knowledge transferring. Previous research mainly end-to-end fine-tuned and transferred CNN representations between different categories [19], [68], and datasets [20]. In contrast, we believe that a good explanation and transparent representation of parts will create a new possibility of transferring part features. As in [50], [81], the AOG is suitable to represent the semantic hierarchy, which enables semantic-level interactions between human and neural networks.

In addition, the semanticization of feature representations also makes our study from conventional part-detection approaches. First, most detection methods deal with classification problems, but inspired by graph mining [73], [74], [76], we mainly focus on a mining problem, *i.e.* discovering meaningful latent patterns to clarify CNN representations. Second, instead of summarizing common knowledge from massive annotations, our method requires very limited supervision to mine latent patterns.

2.2 Learning of compositional models with limited annotations

The labeling cost is an important issue of learning compositional models. Many methods have been developed to learn object models in an unsupervised or weakly supervised manner. Methods of [7], [51], [55], [76] learned with image-level annotations without labeling object bounding boxes. [9], [14] did not require any annotations during the learning process. [10] collected training data online from videos to incrementally learn models. [15], [56] discovered objects and identified actions from language Instructions and videos. Inspired by active learning [36], [58], [64], the idea of learning from question-answering has been used to learn object models [12], [46], [62]. Branson *et al.* [5] used human-computer interactions to label object parts to learn part models. Instead of directly

building new models from active QA, our method uses the QA to mine AOG part representations from CNN representations.

Modeling "objects" vs. modeling "parts": Generally speaking, it is usually more difficult to model object parts than to represent entire objects, when people limit the number of human annotations. For example, object discovery [42], [43], [51] and co-segmentation [3] only require image-level labels without object bounding boxes. Object discovery is mainly implemented by identifying common foreground patterns from the noisy background. People usually consider closed boundaries and common object structure as a strong prior for object discovery.

In contrast to objects, it is difficult to mine correct part parsing of objects without sufficient supervision. Up to now, there is no reliable solution to distinguishing semantically meaningful parts from other potential divisions of object parts in an unsupervised manner. In particular, some parts (*e.g.* the abdomen) do not have shape boundaries to determine their shape extent.

2.3 Active learning

Active learning aims to solve the following problem: how to train an accurate model with minimum cost of labeling [18]. Thus, the goal of active learning is to select the minimum instances from the unlabeled sample set to query from an oracle (e.g., a human). The two widely used selection criteria are representativeness and informativeness [80].

Representativeness-based approaches aim to exploit the cluster structure or the similarity of unlabeled data [80]. [11], [40] applied the clustering method and selected samples based on the clusters. [79] introduced some methods which focused on the measures of similarity among samples.

Approaches based on informativeness usually choose the "most confused" samples with respect to the classifier being used [60], [61]. [39], [48] generated multiple classifiers and selected the unlabeled instance on which the classifiers disagreed to the most. Method of [32] selected the instance on which the classifier was with the least confidence. [24] suggested choosing samples based on the entropy of unlabeled samples. They selected the sample that produced the maximum reduction in entropy once it was labeled. [6], [65] proposed to ask questions on each image and used the user responses to update predictions. These questions were also selected based on the information gain.

Unlike the above methods, we focus on learning an interpretable hierarchical model, and developing the active QA to explain more features incrementally. In the active QA process, samples are selected based on the uncertainty of part localization.

3 METHOD

The overall objective is to sequentially minimize the following three loss terms.

$$Loss = Loss^{\text{CNN}} + Loss^{\text{QA}} + Loss^{\text{AOG}} \quad (1)$$

$Loss^{\text{CNN}}$ denotes the classification loss of the CNN.

$Loss^{\text{QA}}$ is referred as to the loss for active QA. Given the current AOG, we use $Loss^{\text{QA}}$ to actively determine a sequence of questions about objects that cannot be explained by the current AOG, and require people to annotate bounding boxes of new object parts for supervision.

$Loss^{\text{AOG}}$ is designed to learn an AOG for the CNN. $Loss^{\text{AOG}}$ penalizes 1) the incompatibility between the AOG and CNN feature maps of unannotated objects and 2) part-location errors *w.r.t.* the annotated ground-truth part locations.

It is essential to determine the optimization sequence for the three losses in the above equation. We propose to first learn the CNN by minimizing $Loss^{\text{CNN}}$ and then build an AOG based on the learned CNN. We use the active QA to obtain new part annotations and use new part annotations to grow the AOG by optimizing $Loss^{\text{QA}}$ and $Loss^{\text{AOG}}$ alternatively.

Processes of learning CNNs, active QA, and learning AOGs are conducted recursively, so three terms in Equation (1) are not optimized jointly. We introduce details of the three losses in the following subsections.

3.1 Learning convolutional neural networks

To simplify the story, in this research, we just consider a CNN for single-category classification, *i.e.* identifying object images of a specific category from random images. We use the log-logistic loss to learn the CNN.

$$Loss^{\text{CNN}} = \mathbb{E}_{I\in\mathbf{I}}\big[Loss(\hat{y}_I, y_I^*)\big] \quad (2)$$

where $\hat{y}_I$ and y_I^* denote the predicted and ground-truth labels of an image I. If the image I belongs to the target category, then $y_I^* = +1$; otherwise $y_I^* = -1$.

3.2 Learning And-Or graphs

We are given a pre-trained CNN and its training images without part annotations. We use an active QA process to obtain a small number of annotations of object-part bounding boxes, which will be introduced in Section 3.3. Based on these inputs, in this subsection, we focus on the approach for learning an AOG to represent the object part.

3.2.1 And-Or graph representations

Before the introduction of learning AOGs, we first briefly overview the structure of the AOG and the part parsing (inference) based on the AOG.

As shown in Fig. 1, an AOG represents the semantic structure of a part at four layers.

Layer	Name	Node type
1	semantic part	OR node
2	part template	AND node
3	latent pattern	OR node
4	neural unit	Terminal node

In the AOG, each OR node encodes a list of alternative appearance (or deformation) candidates as children. Each AND node uses its children to represent its constituent regions.

More specifically, the top node is an OR node, which represents a certain semantic part, *e.g.* the head or the tail. The semantic part node encodes some part templates as children. Each part template corresponds to a specific part appearance from a certain perspective. During the inference process, the semantic part (an OR node) selects the best part template among all template candidates to represent the object.

The part template in the second layer is an AND node, which uses its children latent patterns to represent a constituent region or a contextual region *w.r.t.* the part template. The part template encodes spatial relationships between its children.

The latent pattern in the third layer is an OR node, whose receptive field is a square block within the feature map of a specific convolutional filter. The latent pattern takes neural units inside its receptive field as children. Because the latent pattern may appear at different locations in the feature map, the latent pattern uses these neural units to represent its deformation candidates. During the inference process, the latent pattern selects the strongest activated child unit as its deformation configuration.

Given an image I[1], we use the CNN to compute feature maps of all conv-layers on image I. Then, we use the AOG for hierarchical part parsing. *I.e.* the AOG semanticizes the feature maps and localizes the target part and its constituent regions in different layers.

The parsing result is illustrated as red lines in Fig. 1. From a top-down perspective, the parsing procedure 1) identifies a part template for the semantic part; 2) parses an image region for the selected part template; 3) for each latent pattern under the part template, it selects a neural unit within a specific deformation range to represent this pattern.

OR nodes: Both the top semantic-part node and latent-pattern nodes in the third layer are OR nodes. The parsing process assigns each OR node u with an image region Λ_u and an inference score S_u. S_u measures the fitness between the parsed region Λ_u and the sub-AOG under u. The computation of Λ_u and S_u for all OR nodes shares the same paradigm.

$$S_u = \max_{v \in Child(u)} S_v, \qquad \Lambda_u = \Lambda_{\hat{v}} \tag{3}$$

where let u have m children nodes $Child(u) = \{v_1, v_2, \ldots, v_m\}$. S_v denotes the inference score of the child v, and Λ_v is referred to as the image region assigned to v. The OR node selects the child with the highest score $\hat{v} = \text{argmax}_{v \in Child(u)} S_v$ as the true parsing configuration. Node $\hat{v}$ propagates its image region to the parent u.

More specifically, we introduce detailed settings for different OR nodes.

- The OR node of the top semantic part contains a list of alternative part templates. We use *top* to denote the top node of the semantic part. The semantic part chooses a part template to describe each input image I.
- The OR node of each latent pattern u in the third layer naturally corresponds to a square deformation range within the feature map of a convolutional filter of a conv-layer. All neural units within the square are used as deformation candidates of the latent pattern. For simplification, we set a constant deformation range (with a center $\overline{\mathbf{p}}_u$ and a scale of $\frac{h}{3} \times \frac{w}{3}$ in the feature map where h and w ($h = w$) denote the height and width of the feature map) for each latent pattern. $\overline{\mathbf{p}}_u$ is a parameter that needs to be learned. Deformation ranges of different patterns in the same feature map may overlap. Given parsing configurations of children neural units as input, the latent pattern selects the child with the highest inference score as the true deformation configuration.

AND nodes: Each part template is an AND node, which uses its children (latent patterns) to represent its constituent or contextual regions. We use v and $Child(v) = \{u_1, u_2, \ldots, u_m\}$ to denote the part template and its children latent patterns. We learn the average displacement from Λ_u to Λ_v over different images, denoted by $\Delta\mathbf{p}_u$, as a parameter of the AOG. Given parsing results of children latent patterns, we use the image region of each child node Λ_u to infer the region for the parent v based on its spatial relationships. Just like a deformable part model, the parsing of v is given as

$$S_v = \sum_{u \in Child(v)} \left[S_u + S^{\text{inf}}(\Lambda_u | \Lambda_v) \right], \quad \Lambda_v = f(\Lambda_{u_1}, \ldots, \Lambda_{u_m}) \tag{4}$$

where we use parsing results of children nodes to infer the parent part template v. $S^{\text{inf}}(\Lambda_u|\Lambda_v)$ denotes the spatial compatibility between Λ_u and Λ_v *w.r.t.* their average displacement $\Delta\mathbf{p}_u$. Please see the appendix for details of $S^{\text{inf}}(\Lambda_u|\Lambda_v)$.

For the region parsing of the part template v, we need to estimate two terms, *i.e.* the center position $\mathbf{p}_v$ and the scale $scale_v$ of Λ_v. We learn a fixed scale

1. Because the CNN has demonstrated its superior performance in object detection, we assume that the target object is well detected by the pre-trained CNN. As in [8], we regard object detection and part localization as two separate processes for evaluation. Thus, to simplify the learning scenario, we crop I only to contain the object, resize it to the image size for CNN inputs, and just focus on the part localization task to simplify the scenario of learning for part localization.

for each part template, which will be introduced in Section 3.2.2. In this way, we simply implement region parsing by computing the region position that maximizes the inference score $\mathbf{p}_v = f(\Lambda_{u_1}, \Lambda_{u_2}, \ldots, \Lambda_{u_m}) = \text{argmax}_{\mathbf{p}_v} S_v$.

Terminal nodes (neural units): Each terminal node under a latent pattern represents a deformation candidate of the latent pattern. The terminal node has a fixed image region, *i.e.* we propagate the neural unit's receptive field back to the image plane as its image region. We compute a neural unit's inference score based on both its neural response value and its displacement *w.r.t.* its parent latent pattern. Please see the appendix for details.

Based on the above node definitions, we use the AOG to parse each given image I by dynamic programming in a bottom-up manner.

3.2.2 Learning And-Or graphs

The core of learning AOGs is to distinguish reliable latent patterns from noisy neural responses in conv-layers and select reliable latent patterns to construct the AOG.

Training data: Let $\mathbf{I}^{\text{obj}} \subset \mathbf{I}$ denote the set of object images of a target category. During the active question-answering, we obtain bounding boxes of the target object part in a small number of images, $\mathbf{I}^{\text{ant}} = \{I_1, I_2, \ldots, I_M\} \subset \mathbf{I}^{\text{obj}}$ among all objects. The other images without part annotations are denoted by $\mathbf{I}^{\text{unant}} = \mathbf{I}^{\text{obj}} \setminus \mathbf{I}^{\text{ant}}$. In addition, the question-answering process collects a number of part templates. Thus, for each image $I \in \mathbf{I}^{\text{ant}}$, we annotate (Λ^*_{top}, v^*), where Λ^*_{top} denotes the ground-truth bounding box of the part in I, and $v^* \in Child(top)$ specifies the ground-truth template for the part.

Which AOG parameters to learn: We use human annotations to define the first two layers of the AOG. If human annotators specify a total of m different part templates during the annotation process, correspondingly, we directly connect the top node with m part templates as children. For each part template $v \in Child(top)$, we fix a constant scale for its region Λ_v. *I.e.* if there are n ground-truth part boxes that are labeled for v, we compute the average scale among the n part boxes as the constant scale $scale_v$.

Thus, the key to AOG construction is to mine children latent patterns for each part template v. We need to mine latent patterns from a total of K conv-layers. We select n_k latent patterns from the k-th ($k = 1, 2, \ldots, K$) conv-layer, where K and $\{n_k\}$ are hyper-parameters. Let each latent pattern u in the k-th conv-layer correspond to a square deformation range, which is located in the D_u-th slice of the conv-layer's feature map. $\overline{\mathbf{p}}_u$ denotes the center of the range. As analyzed in the appendix, we only need to estimate the parameters of $D_u, \overline{\mathbf{p}}_u$ for u.

How to learn: Just like the pattern pursuing in Fig. 1, we mine the latent patterns by estimating their best locations $D_u, \overline{\mathbf{p}}_u \in \boldsymbol{\theta}$ that minimize the following objective function, where $\boldsymbol{\theta}$ denotes the parameter set of the AOG.

$$\begin{aligned} Loss^{\text{AOG}} = \mathbb{E}_{I\in\mathbf{I}^{\text{ant}}}\big[- S_{top} + L(\Lambda_{top}, \Lambda^*_{top})\big] \\ +\lambda^{\text{unant}}\mathbb{E}_{I\in\mathbf{I}^{\text{obj}}}\big[- S^{\text{unant}}_{\text{AOG}} + L^{\text{unant}}(\mathbf{\Lambda}_{\text{AOG}})\big] \end{aligned} \tag{5}$$

First, let us focus on the first half of the equation, which learns from part annotations. S_{top} and $L(\Lambda_{top}, \Lambda^*_{top})$ denote the final inference score of the AOG on image I and the loss of part localization, respectively. Given annotations (Λ^*_{top}, v^*) on I, we get

$$\begin{aligned} S_{top} &= \max_{v\in Child(top)} S_v \approx S_{v^*} \\ L(\Lambda_{top}, \Lambda^*_{top}) &= -\lambda_{v^*}\|\mathbf{p}_{top} - \mathbf{p}^*_{top}\| \end{aligned} \tag{6}$$

where we approximate the ground-truth part template v^* as the selected part template. We ignore the small probability of the AOG assigning an annotated image with an incorrect part template to simplify the computation. The part-localization loss $L(\Lambda_{top}, \Lambda^*_{top})$ measures the localization error between the parsed part region $\mathbf{p}_{top}$ and the ground truth $\mathbf{p}^*_{top} = \mathbf{p}(\Lambda^*_{top})$.

The second half of Equation (5) learns from objects without part annotations.

$$\begin{aligned} S^{\text{unant}}_{\text{AOG}} &= \sum\nolimits_{u\in Child(v^*)} S^{\text{unant}}_u \\ L^{\text{unant}}(\mathbf{\Lambda}_{\text{AOG}}) &= \sum\nolimits_{u\in Child(v^*)} \lambda^{\text{close}}\|\Delta\mathbf{p}_u\|^2 \end{aligned} \tag{7}$$

where the first term $S^{\text{unant}}_{\text{AOG}}$ denotes the inference score at the level of latent patterns without ground-truth annotations of object parts. Please see the appendix for the computation of S^{unant}_u. The second term $L^{\text{unant}}(\mathbf{\Lambda}_{\text{AOG}})$ penalizes latent patterns that are far from their parent v^*. This loss encourages the assigned neural unit to be close to its parent latent pattern. We assume that 1) latent patterns that frequently appear among unannotated objects may potentially represent stable part appearance and should have higher priorities; and that 2) latent patterns spatially closer to their parent part templates are usually more reliable.

When we set λ_{v^*} to a constant $\lambda^{\text{inf}}\sum_{k=1}^{K} n_k$, we can transform the learning objective in Equation (5) as follows.

$$\forall v \in Child(top), \quad \min_{\boldsymbol{\theta}_v} L_v, \quad L_v = -\sum_{u\in Child(v)} Score(u) \tag{8}$$

where $Score(u) = \mathbb{E}_{I\in\mathbf{I}_v}[S_u + S^{\text{inf}}(\Lambda_u|\Lambda^*_v)] + \mathbb{E}_{I'\in\mathbf{I}^{\text{obj}}} \lambda^{\text{unant}}[S^{\text{unant}}_u - \lambda^{\text{close}}\|\Delta\mathbf{p}_u\|^2]$. $\boldsymbol{\theta}_v \subset \boldsymbol{\theta}$ denotes the parameters for the sub-AOG of the part template v. We use $\mathbf{I}_v \subset \mathbf{I}^{\text{ant}}$ to denote the subset of images that are annotated with v as the ground-truth part template.

Learning the sub-AOG for each part template: Based on Equation (8), we mine the sub-AOG for each part template v, which uses this template's annotations on images $I \in \mathbf{I}_v \subset \mathbf{I}^{\text{ant}}$, as follows.

1) We first enumerate all possible latent patterns corresponding to the k-th CNN conv-layer ($k = 1, \ldots, K$), by sampling all pattern locations *w.r.t.* D_u and $\overline{\mathbf{p}}_u$.
2) Then, we sequentially compute Λ_u and $Score(u)$ for each latent pattern.
3) Finally, we sequentially select a total of n_k latent patterns. In each step, we select $\hat{u} = \text{argmax}_{u \in Child(v)} \Delta L_v$. *I.e.* we select latent patterns with top-ranked values of $Score(u)$ as children of part template v.

Computational complexity: During the learning of the sub-AOG for each part template, we enumerate all possible latent patterns u and compute the $Score(u)$ for each u. Thus, the computational complexity is $O(MN \log M)$ considering the cost of selecting top-ranked $Score(u)$, where M denotes the number of all possible latent patterns, and N denotes the number of object images in $\mathbf{I}^{\text{obj}}$.

3.3 Learning via active question-answering

We propose a new learning strategy, *i.e.* active QA, which is more efficient than conventional batch learning. The QA-based learning algorithm actively detects blind spots in feature representations of the model and ask questions for supervision. In general, blind spots in the AOG include 1) neural-activation patterns in the CNN that have not been encoded in the AOG and 2) inaccurate latent patterns in the AOG. The unmodeled neural patterns potentially reflect new part templates, while inaccurate latent patterns correspond to sub-optimized part templates.

As an interpretable representation of object parts, the AOG represents blind spots using linguistic description. We design five types of answers to project these blind spots onto semantic details of objects. Our method selects and asks a series of questions. We then collect answers from human users, in order to incrementally grow new AOG branches to explain new part templates and refine existing AOG branches of part templates.

Our approach repeats the following QA process. As shown in Fig. 2, at first, we use the current AOG to localize object parts on all unannotated objects of a category. Based on localization results, the algorithm selects and asks about the object I, from which the AOG obtains the most information gain. A question $q = (I, \hat{v}, \Lambda_{\hat{v}})$ requires people to determine whether our approach predicts the correct part template $\hat{v}$ and parses a correct region $\Lambda_{top} = \Lambda_{\hat{v}}$ for the part. Our method expects one of the following answers.

Answer 1: the part detection is correct. **Answer 2:** the current AOG predicts the correct part template in the parse graph, but it does not accurately localize the part. **Answer 3:** neither the part template nor the part location is correctly estimated. **Answer 4:** the part belongs to a new part template. **Answer 5:** the target part does not appear in the image. In particular, in case of receiving Answers 2–4, our method will ask people to annotate the target part. In case of getting Answer 3, our method will require people to specify its part template and whether the object is flipped. Our method uses new part annotations to refine (for Answers 2–3) or create (for Answer 4) an AOG branch of the annotated part template based on Equation (5).

3.3.1 Question ranking

The core of the QA-based learning is to select a sequence of questions that reduce the uncertainty of part localization the most. Therefore, in this section, we design a loss function to measure the incompatibility between the AOG and real part appearances in object samples. Our approach predicts the potential gain (decrease of the loss) of asking about each object. Objects with large gains usually correspond to not well explained CNN neural activations. Note that annotating a part in an object may also help localize parts on other objects, thereby leading to a large gain. Thus, we use a greedy strategy to select a sequence of questions $\Omega = \{q_i | i = 1, 2, \ldots\}$, *i.e.* asking about the object that produces the most gain in each step.

For each object image I, we use $\mathbf{P}(y|I)$ and $\mathbf{Q}(y|I)$ to denote the prior distribution and the estimated distribution of an object part on I, respectively. A label $y \in \{+1, -1\}$ indicates whether I contains the target part. The AOG estimates the probability of object I containing the target part as $\mathbf{Q}(y = +1|I) = \frac{1}{Z} \exp[\beta S_{top}]$, where Z and β are parameters for scaling (see Section 4.1 for details); $\mathbf{Q}(y = -1|I) = 1 - \mathbf{Q}(y = +1|I)$. Let $\mathbf{I}^{\text{ant}}$ denote the set of objects that have been asked during previous QA. For each asked object $I \in \mathbf{I}^{\text{ant}}$, we set its prior distribution $\mathbf{P}(y = +1|I) = 1$ if I contains the target part; $\mathbf{P}(y = +1|I) = 0$ otherwise. For each un-asked object $I \in \mathbf{I}^{\text{unant}}$, we set its prior distribution based on statistics of previous answers, $\mathbf{P}(y = +1|I) = \mathbb{E}_{I' \in \mathbf{I}^{\text{ant}}} \mathbf{P}(y = +1|I')$. Therefore, we formulate the loss function as the KL divergence between the prior distribution $\mathbf{P}$ and the estimated distribution $\mathbf{Q}$.

$$\begin{aligned} Loss^{\text{QA}} = \mathbf{KL}(\mathbf{P} \| \mathbf{Q}) &= \sum_{I \in \mathbf{I}^{\text{obj}}} \sum_{y} \mathbf{P}(y, I) \log \frac{\mathbf{P}(y, I)}{\mathbf{Q}(y, I)} \\ &= \lambda \sum_{I \in \mathbf{I}^{\text{obj}}} \sum_{y} \mathbf{P}(y|I) \log \frac{\mathbf{P}(y|I)}{\mathbf{Q}(y|I)} \end{aligned} \tag{9}$$

where $\mathbf{P}(y, I) = \mathbf{P}(y|I)P(I)$; $\mathbf{Q}(y, I) = \mathbf{Q}(y|I)P(I)$; $\lambda = P(I) = 1/|\mathbf{I}^{\text{obj}}|$ is a constant prior probability for I.

We keep modifying both the prior distribution $\mathbf{P}$ and the estimated distribution $\mathbf{Q}$ during the QA process. Let the algorithm select an unannotated object $\tilde{I} \in \mathbf{I}^{\text{unant}} = \mathbf{I}^{\text{obj}} \setminus \mathbf{I}^{\text{ant}}$ and ask people to label its part. The annotation would encode part representations of $\tilde{I}$ into the AOG and significantly change the estimated distribution for objects that are similar to $\tilde{I}$. For each object $I' \in \mathbf{I}^{\text{obj}}$, we predict its estimated distribution

after a new part annotation as

$$\tilde{\mathbf{Q}}(y=+1|I') = \frac{1}{Z}\exp[\beta S^{\text{new}}_{top,I'}|_{\tilde{I}}]$$
$$S^{\text{new}}_{top,I'}|_{\tilde{I}} = S_{top,I'} + \Delta S_{top,\tilde{I}} e^{-\alpha \cdot dist(I',\tilde{I})} \quad (10)$$

where $S_{top,I'}$ indicates the current AOG's inference score of S_{top} on image I'. $S^{\text{new}}_{top,I'}|_{\tilde{I}}$ denotes the predicted inference score of I' when people annotate $\tilde{I}$. We assume that if object I' is similar to object $\tilde{I}$, the inference score of I' will have an increase similar to that of $\tilde{I}$. $\Delta S_{top,\tilde{I}} = \mathbb{E}_{I\in\mathbf{I}^{\text{ant}}} S_{top,I} - S_{top,\tilde{I}}$ denotes the score increase of $\tilde{I}$. α is a scalar weight. We formulate the appearance distance between I' and $\tilde{I}$ as $dist(I',\tilde{I}) = 1 - \frac{\phi(I')^T\phi(\tilde{I})}{|\phi(I')|\cdot|\phi(\tilde{I})|}$, where $\phi(I') = \mathbf{M}\,\mathbf{f}_{I'}$. $\mathbf{f}_{I'}$ denotes features of I' at the top conv-layer after ReLU operation, and $\mathbf{M}$ is a diagonal matrix representing the prior reliability for each feature dimension[2]. In addition, if I' and $\tilde{I}$ are assigned with different part templates by the current AOG, we set an infinite distance between I' and $\tilde{I}$ to achieve better performance. Based on Equation (10), we predict the changes of the KL divergence after the new annotation on $\tilde{I}$ as

$$\Delta\mathbf{KL}(\tilde{I}) = \lambda \sum\nolimits_{I\in\mathbf{I}^{\text{obj}}} \sum\nolimits_{y} \mathbf{P}(y|I)\log\frac{\tilde{\mathbf{Q}}(y|I)}{\mathbf{Q}(y|I)} \quad (11)$$

Thus, in each step, our method selects and asks about the object that decreases the KL divergence the most.

$$\hat{I} = \text{argmax}_{I\in\mathbf{I}^{\text{unant}}} \Delta\mathbf{KL}(I) \quad (12)$$

QA implementations: In the beginning, for each object I, we initialize $\mathbf{P}(y=+1|I)=1$ and $\mathbf{Q}(y=+1|I)=0$. Then, our approach selects and asks about an object $\hat{I}$ based on Equation (12). We use the answer to update $\mathbf{P}$. If a new object part is labeled during the QA process, we apply Equation (5) to update the AOG. More specifically, if people label a new part template, our method will grow a new AOG branch to encode this template. If people annotate a part for an old part template, our method will update its corresponding AOG branch. Then, we compute the new distribution $\mathbf{Q}$ based on the new AOG. In this way, the above QA procedure gradually grows the AOG.

Computational complexity: In each round of QA, we estimate the potential gain ($\Delta\mathbf{KL}$) of asking about each object I, where we consider the similarity between all pairs of objects. Thus, the cost of calculating the similarity between all pairs of objects is $O(N)$, and the computational complexity of updating the distribution after each QA step is $O(N)$, where N denotes the number of object images in $\mathbf{I}^{\text{obj}}$. Thus, the complexity of all QA process is $O(N^2)$.

2. $\mathbf{M}_{ii} \propto \exp[\mathbb{E}_{I\in\mathbf{I}} S_{v_i^{\text{unt}}}]$, where v_i^{unt} is the neural unit corresponding to the i-th element of $\mathbf{f}_{I'}$.

4 Experiments

4.1 Implementation details

We used a 16-layer VGG network (VGG-16) [54], which was pre-trained for object classification using 1.3M images in the ImageNet ILSVRC 2012 dataset [45]. Then, for each testing category, we further fine-tune the VGG-16 using object images in this category to classify target objects from random images. We selected the last nine conv-layers of VGG-16 as valid conv-layers. We extracted neural units from these conv-layers to build the AOG.

In addition to the VGG-16 network, we also used the residual network with 101 convolutional layers [23] (termed as ResNet-101) in experiments. The ResNet-101 was pre-trained using the ImageNet ILSVRC 2012 dataset. We further fine-tune the ResNet-101 to classify bird images in the CUB200-2011 dataset [66] from random images in the ImageNet dataset. We selected the highest three conv-layers that generated 14×14 feature maps, the highest three conv-layers that generated 28×28 feature maps, and the highest three conv-layers that generated 56×56 feature maps of the ResNet-101 as valid conv-layers. We extracted neural units from these conv-layers to build the AOG.

Active question-answering: Three parameters were involved in our active-QA method, *i.e.* α, β, and Z. Because most objects of the category contained the target part, we ignored the small probability of $\mathbf{P}(y=-1|I)$ in Equation (11) to simplify the computation. As a result, Z was eliminated in Equation (11), and the constant weight β did not affect object-selection results in Equation (12). We set $\alpha = 4.0$ in our experiments.

Learning AOGs: Multiple latent patterns corresponding to the same convolutional filter may have similar positions $\overline{\mathbf{p}}_u$, and their deformation ranges may highly overlap. Thus, we selected the latent pattern with the highest $Score(u)$ within a small range of $\epsilon\times\epsilon$ in the filter's feature map and removed other nearby patterns to obtain a spare AOG. Besides, for each part template v, we estimated n_k latent patterns in the k-th conv-layer. We assumed that scores of all latent patterns in the k-th conv-layer follow the distribution of $Score(u) \sim \alpha\exp[-(\xi\cdot rank)^{0.5}] + \gamma$, where $rank$ denotes the score rank of u. We set $n_k = \lceil 0.5/\xi \rceil$, which learned the best AOG.

4.2 Datasets

Because the evaluation of part localization requires ground-truth annotations of part positions, we used the following three benchmark datasets to test our method, *i.e.* the PASCAL VOC Part Dataset [8], the CUB200-2011 dataset [66], and the ILSVRC 2013 DET Animal-Part dataset [71]. Just like in [8], [71], we selected animal categories, which prevalently contain

TABLE 2
Average number of children of AOG nodes

Annotation number	Layer 1: semantic part	Layer 2: part template	Layer 3: latent pattern
05	3.15	3791.5	91.6
10	5.95	3804.8	93.9
15	8.52	3760.4	95.5
20	11.16	3778.3	96.3
25	13.55	3777.5	98.3
30	15.83	3837.3	99.2

non-rigid shape deformation, for testing. *I.e.* we selected six animal categories—*bird*, *cat*, *cow*, *dog*, *horse*, and *sheep*—from the PASCAL Part Dataset. The CUB200-2011 dataset contains 11.8K images of 200 bird species. We followed [5], [52], [71] and used all these images as a single bird category for learning. The ILSVRC 2013 DET Animal-Part dataset [71] contains part annotations of 30 animal categories among all the 200 categories in the ILSVRC 2013 DET dataset [45].

4.3 Baselines

We used the following thirteen baselines for comparison. The first two baselines were based on the Fast-RCNN [21]. We fine-tuned the fast-RCNN with a loss of detecting a single class/part for a fair comparison. The first baseline, namely *Fast-RCNN (1 ft)*, fine-tuned the VGG-16 using part annotations to detect parts on well-cropped objects. To enable a more fair comparison, we conducted the second baseline based on two-stage fine-tuning, namely *Fast-RCNN (2 fts)*. This baseline first fine-tuned the VGG-16 using numerous object-box annotations in the target category, and then fine-tuned the VGG-16 using a few part annotations.

The third baseline was proposed in [52], namely *CNN-PDD*. *CNN-PDD* selected a filter in a CNN (pre-trained using ImageNet ILSVRC 2012 dataset) to represent the part on well-cropped objects. Then, we slightly extended [52] as the fourth baseline *CNN-PDD-ft*. *CNN-PDD-ft* first fine-tuned the VGG-16 using object bounding boxes, and then applied [52] to learn object parts.

The strongly supervised DPM (*SS-DPM-Part*) [2] and the approach of [33] (*PL-DPM-Part*) were the fifth and sixth baselines. These methods learned DPMs for part localization. The graphical model proposed in [8] was selected as the seventh baseline, namely *Part-Graph*. The eighth baseline was the interactive learning for part localization [5] (*Interactive-DPM*).

Without lots of training samples, "simple" methods are usually insensitive to the over-fitting problem. Thus, we designed the last four baselines as follows. We first fine-tuned the VGG-16 using object bounding boxes, and collected image patches from cropped objects based on the selective search [63]. We used the VGG-16 to extract *fc7* features from image patches. The two baselines (*i.e. fc7+linearSVM* and *fc7+RBF-SVM*) used a linear SVM and an RBF-SVM, respectively, to detect object parts. The other baselines *VAE+linearSVM* and *CoopNet+linearSVM* used features of the VAE network [27] and the CoopNet [67], respectively, instead of *fc7* features, for part detection.

The last baseline [71] learned AOGs without QA (*AOG w/o QA*). We randomly selected objects and annotated their parts for training.

Both object annotations and part annotations are used to learn models in all the thirteen baselines (including those without fine-tuning). *Fast-RCNN (1 ft)* and *CNN-PDD* used the cropped objects as the input of the CNN; *SS-DPM-Part*, *PL-DPM-Part*, *Part-Graph*, and *Interactive-DPM* used object boxes and part boxes to learn models. *CNN-PDD-ft*, *Fast-RCNN (2 fts)*, and methods based on *fc7* features used object bounding boxes for fine-tuning.

TABLE 4
Part localization performance on the CUB200 dataset.

	Obj.-box finetune	Part Annot.	#Q	Normalizaed distance
SS-DPM-Part [2]	No	60	–	0.2504
PL-DPM-Part [33]	No	60	–	0.3215
Part-Graph [8]	No	60	–	0.3697
fc7+linearSVM	Yes	60	–	0.2786
fc7+RBF-SVM	Yes	60	–	0.3360
Interactive-DPM [5]	No	60	–	0.2011
CNN-PDD [52]	No	60	–	0.2446
CNN-PDD-ft [52]	Yes	60	–	0.2694
Fast-RCNN (1 ft) [21]	No	60	–	0.3105
Fast-RCNN (2 fts) [21]	Yes	60	–	0.1989
AOG w/o QA [71]	Yes	20	–	0.1084
Ours (VGG-16, setting $\lambda^{\text{unant}} = 0$)	Yes	10	46	0.0574
Ours (VGG-16, setting $\lambda^{\text{unant}} = 0$)	Yes	20	150	0.0464
Ours (based on ResNet-101)	Yes	10	–	0.2807
Ours (based on ResNet-101)	Yes	20	–	0.2510
Ours (based on VGG-16)	Yes	10	28	0.0570 ± 0.00449
Ours (based on VGG-16)	Yes	20	112	0.0455 ± 0.00439

See Table 3 for the introduction of the 2nd and 3rd columns. The 4th column shows the number of questions for training. The 4th column indicates whether the baseline used all object annotations (*more than part annotations*) in the category to pre-fine-tune a CNN before learning the part.

4.4 Evaluation metric

As discussed in [8], [71], a fair evaluation of part localization requires removing factors of object detection. Thus, we used ground-truth object bounding boxes to crop objects as testing images. Given an object image, some competing methods (*e.g. Fast-RCNN (1 ft)*, *Part-Graph*, and *SS-DPM-Part*) estimate several bounding boxes for the part with different confidences. We followed [8], [42], [52], [71] to take the most confident bounding box per image as the part-localization result. Given part-localization results of a category, we applied the *normalized distance* [52] and the *percentage of correctly localized parts* (PCP) [34], [49], [70] to evaluate the localization accuracy. We measured the distance between the predicted part center and the ground-truth part center, and then normalized the distance using the diagonal length of the object as the normalized distance. For the PCP, we used the typical metric of "$IoU \geq 0.5$" [21] to identify correct part localizations.

4.5 Experimental results

We learned AOGs for the head, the neck, and the nose/muzzle/beak parts of the six animal categories

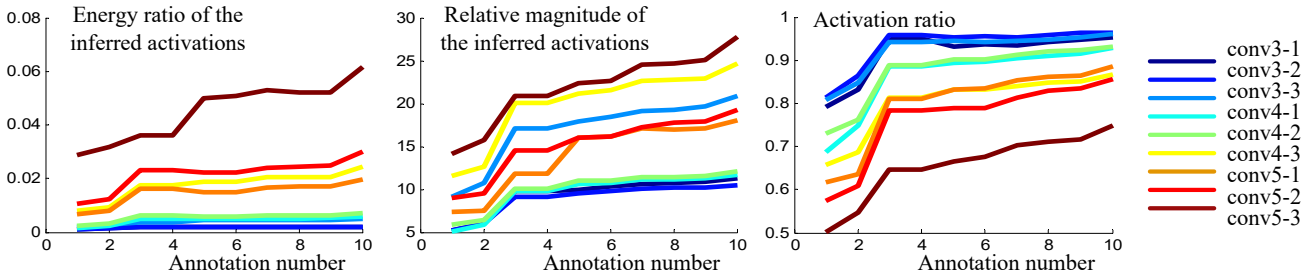


Fig. 3. Activation states of latent patterns under the selected part template. (left) The ratio of the inferred activation energy to all activation energy in feature maps. (middle) The relative magnitude of the inferred activations, which is normalized by the average activation value of all neural units on the feature map. (right) The ratio of latent patterns that are assigned with an activated neural unit. Different curves show scores computed based on latent patterns or neural activations in different conv-layers.

TABLE 3
Normalized distance of part localization on the ILSVRC 2013 DET Animal-Part dataset.

	Part Annot.	Obj.-box finetune	gold.	bird	frog	turt.	liza.	koala	lobs.	dog	fox	cat	lion	tiger	bear	rabb.	hams.	squi.
SS-DPM-Part [2]	60	No	0.1859	0.2747	0.2105	0.2316	0.2901	0.1755	0.1666	0.1948	0.1845	0.1944	0.1334	0.0929	0.1981	0.1355	0.1137	0.1717
PL-DPM-Part [33]	60	No	0.2867	0.2337	0.2169	0.2650	0.3079	0.1445	0.1526	0.1904	0.2252	0.1488	0.1450	0.1340	0.1838	0.1968	0.1389	0.2590
Part-Graph [8]	60	No	0.3385	0.3305	0.3853	0.2873	0.3813	0.0848	0.3467	0.1679	0.1736	0.3499	0.1551	0.1225	0.1906	0.2068	0.1622	0.3038
fc7+linearSVM	60	Yes	0.1359	0.2117	0.1681	0.1890	0.2557	0.1734	0.1845	0.1451	0.1374	0.1581	0.1528	0.1525	0.1354	0.1478	0.1287	0.1291
fc7+RBF-SVM	60	Yes	0.1818	0.2637	0.2035	0.2246	0.2538	0.1663	0.1660	0.1512	0.1670	0.1719	0.1176	0.1638	0.1325	0.1312	0.1410	0.1343
CNN-PDD [52]	60	No	0.1932	0.2015	0.2734	0.2195	0.2650	0.1432	0.1535	0.1657	0.1510	0.1787	0.1560	0.1756	0.1444	0.1320	0.1251	0.1776
CNN-PDD-ft [52]	60	Yes	0.2109	0.2531	0.1999	0.2144	0.2494	0.1577	0.1605	0.1847	0.1845	0.2127	0.1521	0.2066	0.1826	0.1595	0.1570	0.1608
Fast-RCNN (1 ft) [21]	30	No	0.0847	0.1520	0.1905	0.1696	0.1412	0.0754	0.2538	0.1471	0.0886	0.0944	0.1004	0.0585	0.1013	0.0821	0.0577	0.1005
Fast-RCNN (2 fts) [21]	30	Yes	0.0913	0.1043	0.1294	0.1632	0.1585	0.0730	0.2530	0.1148	0.0736	**0.0770**	0.0680	**0.0441**	0.1265	0.1017	0.0709	0.0834
Ours	10	Yes	**0.0796**	**0.0850**	**0.0906**	0.2077	**0.1260**	0.0759	**0.1212**	0.1476	**0.0584**	0.1107	0.0716	0.0637	0.1092	**0.0755**	0.0697	**0.0421**
Ours	20	Yes	**0.0638**	**0.0793**	**0.0765**	**0.1221**	**0.1174**	**0.0720**	**0.1201**	**0.1096**	**0.0517**	0.1006	0.0752	0.0624	0.1090	**0.0788**	0.0603	**0.0454**
Ours	30	Yes	**0.0642**	**0.0734**	**0.0971**	**0.0916**	**0.0948**	**0.0658**	**0.1355**	**0.1023**	**0.0474**	0.1011	**0.0625**	0.0632	**0.0964**	**0.0783**	**0.0540**	**0.0499**
			horse	zebra	swine	hippo	catt.	sheep	ante.	camel	otter	arma.	monk.	elep.	red pa.	gia.pa.		**Avg.**
SS-DPM-Part [2]	60	No	0.2346	0.1717	0.2262	0.2261	0.2371	0.2364	0.2026	0.2308	0.2088	0.2881	0.1859	0.1740	0.1619	0.0989		0.1946
PL-DPM-Part [33]	60	No	0.2657	0.2937	0.2164	0.2150	0.2320	0.2145	0.3119	0.2949	0.2468	0.3100	0.2113	0.1975	0.1835	0.1396		0.2187
Part-Graph [8]	60	No	0.2804	0.3376	0.2979	0.2964	0.2513	0.2321	0.3504	0.2179	0.2535	0.2778	0.2321	0.1961	0.1713	0.0759		0.2486
fc7+linearSVM	60	Yes	0.2003	0.2409	0.1632	0.1400	0.2043	0.2274	0.1479	0.2204	0.2498	0.2875	0.2261	0.1520	0.1557	0.1071		0.1776
fc7+RBF-SVM	60	Yes	0.2207	0.1550	0.1963	0.1536	0.2609	0.2295	0.1748	0.2080	0.2263	0.2613	0.2244	0.1806	0.1417	0.1095		0.1838
CNN-PDD [52]	60	No	0.2610	0.2363	0.1623	0.2018	0.1955	0.1350	0.1857	0.2499	0.2486	0.2656	0.1704	0.1765	0.1713	0.1638		0.1893
CNN-PDD-ft [52]	60	Yes	0.2417	0.2725	0.1943	0.2299	0.2104	0.1936	0.1712	0.2552	0.2110	0.2726	0.1463	0.1602	0.1868	0.1475		0.1980
Fast-RCNN (1 ft) [21]	30	No	0.2694	**0.0823**	0.1319	0.0976	0.1309	0.1276	0.1348	0.1609	0.1627	0.1889	0.1367	**0.1081**	0.0791	**0.0474**		0.1252
Fast-RCNN (2 fts) [21]	30	Yes	0.1629	0.0881	**0.1228**	**0.0889**	**0.0922**	**0.0622**	0.1000	0.1519	0.0969	**0.1485**	0.0855	0.1085	**0.0407**	0.0542		0.1045
Ours	10	Yes	**0.1297**	0.1413	0.2145	0.1377	0.1493	0.1415	0.1046	**0.1239**	0.1288	0.1964	**0.0524**	0.1507	0.1081	0.0640		0.1126
Ours	20	Yes	**0.1083**	0.1389	0.1475	0.1280	0.1490	0.1300	**0.0667**	**0.1033**	0.1103	0.1526	**0.0497**	0.1301	0.0802	0.0574		**0.0965**
Ours	30	Yes	**0.1129**	0.1066	0.1408	0.1204	0.1118	0.1260	**0.0825**	**0.0836**	**0.0901**	0.1685	**0.0490**	0.1224	0.0779	0.0577		**0.0909**

The 2nd column shows the number of part annotations for training. The 3rd column indicates whether the baseline used all object-box annotations in the category to pre-fine-tune a CNN before learning the part (*object-box annotations are more than part annotations*).

TABLE 5
Part localization on the Pascal VOC Part dataset.

	Method	Annot.	#Q	bird	cat	cow	dog	horse	sheep	Avg.
Head	Fast-RCNN (1 ft) [21]	10	–	0.326	0.238	0.283	0.286	0.319	0.354	0.301
	Fast-RCNN (2 fts) [21]	10	–	0.233	0.196	0.216	0.206	0.253	0.286	0.232
	Fast-RCNN (1 ft) [21]	20	–	0.352	**0.131**	0.275	0.189	0.293	0.252	0.249
	Fast-RCNN (2 fts) [21]	20	–	0.176	0.132	0.191	0.171	0.231	**0.189**	0.182
	Fast-RCNN (1 ft) [21]	30	–	0.285	0.146	0.228	0.141	0.250	0.220	0.212
	Fast-RCNN (2 fts) [21]	30	–	0.173	0.156	0.150	**0.137**	0.132	0.221	0.161
	Ours	10	14.7	**0.144**	0.146	**0.137**	0.145	**0.122**	0.193	**0.148**
Neck	Fast-RCNN (1 ft) [21]	10	–	0.251	0.333	0.310	0.248	0.267	0.242	0.275
	Fast-RCNN (2 fts) [21]	10	–	0.317	0.335	0.307	0.362	0.271	0.259	0.309
	Fast-RCNN (1 ft) [21]	20	–	0.255	0.359	0.241	0.281	0.268	0.235	0.273
	Fast-RCNN (2 fts) [21]	20	–	0.260	0.289	0.304	0.297	0.255	0.237	0.274
	Fast-RCNN (1 ft) [21]	30	–	0.288	0.324	0.247	0.262	0.210	0.220	0.258
	Fast-RCNN (2 fts) [21]	30	–	0.201	0.276	0.281	0.254	0.220	0.229	0.244
	Ours	10	24.5	**0.120**	**0.144**	**0.178**	**0.152**	**0.161**	**0.161**	**0.152**
Nose/Muzzle/Beak	Fast-RCNN (1 ft) [21]	10	–	0.446	0.389	0.301	0.326	0.385	0.328	0.363
	Fast-RCNN (2 fts) [21]	10	–	0.447	0.433	0.313	0.391	0.338	0.350	0.379
	Fast-RCNN (1 ft) [21]	20	–	0.425	0.372	0.260	0.303	0.334	0.279	0.329
	Fast-RCNN (2 fts) [21]	20	–	0.419	0.351	0.289	0.249	0.296	0.293	0.316
	Fast-RCNN (1 ft) [21]	30	–	0.462	0.336	0.242	0.260	0.247	0.257	0.301
	Fast-RCNN (2 fts) [21]	30	–	0.430	0.338	0.239	0.219	0.271	0.285	0.297
	Ours	10	23.8	**0.134**	**0.112**	**0.182**	**0.156**	**0.217**	**0.181**	**0.164**

The 3rd and 4th columns show the number of part annotations and the average number of questions for training.

in the Pascal VOC Part dataset. For the ILSVRC 2013 DET Animal-Part dataset and the CUB200-2011 dataset, we learned an AOG for the head part[3] of each category. It is because all categories in the two datasets contain the head part. We did not train human annotators. Shape differences between two part templates were often very vague, so that an annotator could assign a part to either part template.

Table 2 shows how the AOG grew when people annotated more parts during the QA process. Given AOGs learned for the PASCAL VOC Part dataset, we computed the average number of children for each node in different AOG layers. The AOG mainly grew by adding new branches to represent new part templates. The refinement of an existing AOG branch did not significantly change the node number of the AOG.

Fig. 3 analyzes activation states of latent patterns in AOGs that were learned with different numbers of part annotations. Given a testing image I for part parsing, we only focused on the inferred latent pat-

3. It is the "forehead" part for birds in the CUB200-2011 dataset.

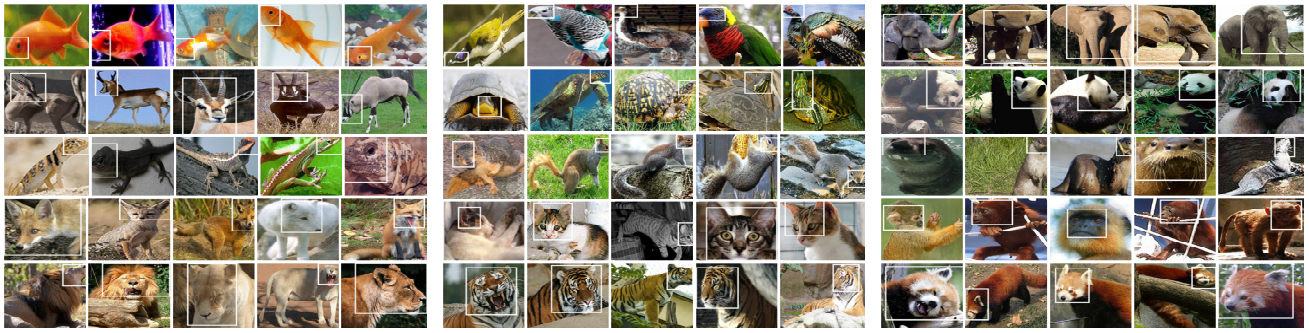


Fig. 4. Part localization results based on AOGs.

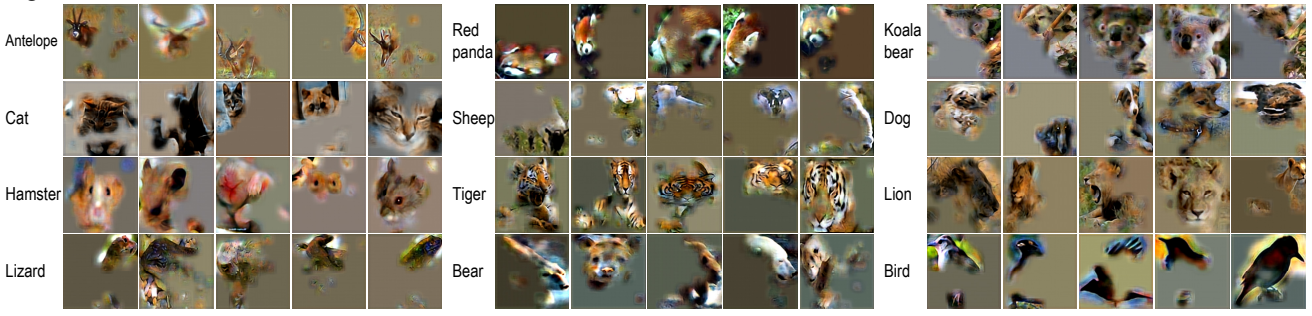


Fig. 5. Visualization of latent patterns in AOGs for the head part. The up-convolutional net [13] synthesizes images corresponding neural activations, which are selected by the AOG during part parsing. We only visualize neural activations selected from conv-layers 5–7. Some latent patterns select neural units corresponding to constituent regions *w.r.t.* the target part, while other latent patterns describe contexts.

Fig. 6. Image patches corresponding to different latent patterns.

terns and neural units, *i.e.* latent patterns and their inferred neural units under the selected part template. Let $\mathbf{V}$ and $\mathbf{V}' \subset \mathbf{V}$ denote all units in a specific conv-layer and the inferred units, respectively. a_v denotes the activation score of $v \in \mathbf{V}$ after the ReLU operation. a_v is also normalized by the average activation level of v's corresponding feature maps *w.r.t.* different images. Thus, in Fig. 3(left), we computed the ratio of the inferred activation energy as $\frac{\sum_{v\in\mathbf{V}'} a_v}{\sum_{v\in\mathbf{V}} a_v}$. For each inferred latent pattern u, a_u denotes the activation score of its selected neural unit[4]. Fig. 3(middle) measures the relative magnitude of the inferred activations, which was measured as $\frac{\mathbb{E}_{u\in\mathbf{U}}[a_u]}{\mathbb{E}_{v\in\mathbf{V}}[a_v]}$. Fig. 3(right) shows the ratio of the latent patterns being strongly activated. We used a threshold $\tau = \mathbb{E}_{v\in\mathbf{V}}[a_v]$ to identify strong activations, *i.e.* computing the activation ratio as $\mathbb{E}_{u\in\mathbf{U}}[\mathbf{1}(a_u > \tau)]$. Curves in Fig. 3 were reported as the average performance using images in the CUB200-2011 dataset.

Fig. 5 visualizes latent patterns in the AOG based on the technique of [13]. More specifically, Fig. 6 lists images patches inferred by different latent patterns in the AOG with high inference scores. It shows that each latent pattern corresponds to a specific part shape through different images.

Fig. 4 shows part localization results based on AOGs. Tables 3, 5, and 4 compare the part-localization performance of different baselines on different benchmark datasets using the evaluation metric of the normalized distance. Tables 4 and 5 show both the number of part annotations and the number of questions. In particular, when we learned AOGs using the CUB200-2011 dataset [66], we required five human annotators to conduct the QA process for five times, in order to compute the mean value and the standard deviation of the normalized distance. As shown in

4. Two latent patterns may select the same neural unit

Table 4, various annotators could generate relatively consistent results. Channel maps of the last 14×14 feature map of the VGG-16 and those of the ResNet-101 are compared in Fig. 7. Because of skip connections, feature maps in residual networks were usually densely activated, which were substantially different from sparsely activated features maps in traditional neural networks without skip connections. Thus, feature maps in residual networks were not simply activated by object parts, which explained the bad performance of AOGs based on residual networks.

Table 6 lists part-localization performance, which was evaluated by the PCP metric. In particular, the method of *Ours+fastRCNN* combined our method and the fast-RCNN to refine part-localization results[5]. Our method learned AOGs with about $1/6$–$1/2$ part annotations, but exhibited superior performance to the second best baseline.

Effects of unannotated objects: In order to evaluate the effects of incorporating unannotated objects during the learning of AOGs, we set $\lambda^{\text{unant}} = 0$ in Equation (5) to test the performance without unannotated objects. AOGs were learned using the CUB200-2011 dataset, and the performance was reported in Table 4. The use of annotated objects improved a bit the localization performance and significantly reduced the number of questions in active QA.

Effects of the order of questions: In order to quantify the effect of the order of questions, we learned AOGs by changing the order of questions. AOGs were learned using the CUB200-2011 dataset, and we learned three AOGs by randomly selecting objects for QA. In this way, the three AOGs were learned using different part annotations. The average performance was reported in Table 8, which shows that the randomly changed question order hurt the performance of AOGs.

Effects of part templates: In order to evaluate the effects of part-template nodes, we randomly removed M part-template nodes ($M = 1, 3$, or 5) from a pre-trained AOG to test the performance. The AOG was pre-trained using the CUB200-2011 dataset. Three sets of M part templates were randomly selected from the AOG to construct three new AOGs, and the average performance of the three AOGs was reported in Table 7. The removal of part templates significantly affected the part-localization performance.

Effects of annotations: In order to evaluate the effects of the number of part annotations, we increased the number of annotations to 60. We learned three AOGs to localize three different parts (head, beak and tail) in the CUB200-2011 dataset. As shown in Fig. 8, the results were saturated when we annotated about 40 object parts. Thus, we could considered the performance at the saturation point as the upper bound of the performance of our method.

5. We used part boxes annotated during the QA process to learn a fast-RCNN for part detection. Given the inference result Λ_v of part template v on image I, we define a new inference score for localization refinement $S_v^{\text{new}}(\Lambda_v^{\text{new}}) = S_v + \lambda_1 \Phi(\Lambda_v^{\text{new}}) + \lambda_2 \frac{\|\mathbf{p}_v - \mathbf{p}_v^{\text{new}}\|}{2\sigma^2}$, where $\sigma = 70$ pixels, $\lambda_1 = 5$, and $\lambda_2 = 10$. $\Phi(\Lambda_v^{\text{new}})$ denotes the fast-RCNN's detection score for the patch of Λ_v^{new}.

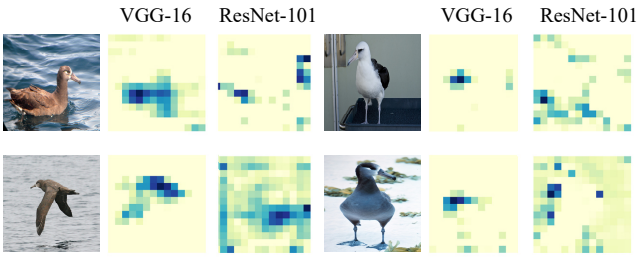


Fig. 7. Comparison of channel maps of the last 14×14 feature maps (after an ReLU operation). Feature maps of the VGG-16 are more likely to represent object parts than the ResNet-101.

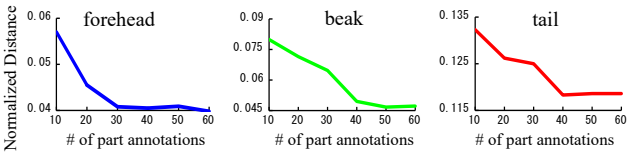


Fig. 8. Part localization performance on the CUB200-2011 dataset.

Effects of errors involved in the human responses: In order to investigate the effects of the errors in the human responses, we involved two types of errors during the learning of AOGs: errors in the object bounding boxes, and errors in the answers to the questions. More specifically, we annotated object parts inaccurately and answered 25% of questions incorrectly. We learned three AOGs using the CUB200-2011 dataset, and reported the average performance in Table 9. The errors involved in human responses hurt the performance of AOGs, but the result was still better than most of baselines in Table 4.

4.6 Justification of the methodology

We have three reasons to explain the good performance of our method. First, **generic information**: the latent patterns in the AOG were pre-fine-tuned using massive object images in a category, instead of being learned from a few part annotations. Thus, these patterns reflected generic part appearances and did not over-fit to a few part annotations.

Second, **fewer model drifts:** Instead of learning new CNN parameters, our method just used limited part annotations to mine the related patterns to represent the part concept. In addition, during active QA, Equation (10) usually selected objects with common poses for QA, *i.e.* choosing objects sharing common latent patterns with many other objects. Thus, the learned AOG suffered less from the model-drift problem.

TABLE 7
Effects of removing different numbers of part templates from a pre-trained AOG.

	Original AOG	remove 1 parts	remove 3 parts	remove 5 parts
# of parts templates	10	9	7	5
Avg. normalized distance	**0.0493**	0.0580	0.0608	0.1010

TABLE 8
Effects of the order of questions.

	AOGs learned by our method	AOGs learned by randomly changing questions
Avg. normalized distance when annotating 10 parts	0.0570 ± 0.00449	0.0726 ± 0.01318
Avg. normalized distance when annotating 20 parts	0.0455 ± 0.00439	0.0732 ± 0.01340

TABLE 9
Effects of the errors involved in human responses.

	AOGs learned by our method	AOGs learned by involving some errors
Avg. normalized distance when annotating 10 parts	0.0570 ± 0.00449	0.0806 ± 0.01460
Avg. normalized distance when annotating 20 parts	0.0455 ± 0.00439	0.0720 ± 0.00542

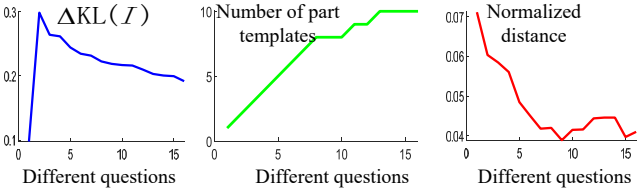


Fig. 9. $\Delta KL(I)$, the number of part templates, and the normalized distance of the AOG learned after different numbers of questions.

TABLE 6
Part localization evaluated using the PCP metric.

	# of part annot.	VOC Part	ILSVRC Animal
SS-DPM-Part [2]	60	7.2	19.2
PL-DPM-Part [33]	60	6.7	12.8
Part-Graph [8]	60	11.0	25.6
fc7+linearSVM	60	13.5	24.5
fc7+RBF-SVM	60	9.5	18.8
VAE+linearSVM [27]	30	6.7	–
CoopNet+linearSVM [67]	30	5.6	–
Fast-RCNN (1 ft) [21]	30	34.5	62.3
Fast-RCNN (2 fts) [21]	30	45.7	68.6
Ours+fastRCNN	10	33.0	53.0
Ours+fastRCNN	20	**47.2**	64.9
Ours+fastRCNN	30	**50.5**	**71.1**

Third, **high QA efficiency:** Our QA process balanced both the commonness and the accuracy of a part template in Equation (10). In the early steps of QA, our approach was prone to asking about new part templates, because objects with un-modeled part appearance usually had low inference scores. In later QA steps, common part appearances had been modeled, and our method gradually changed to ask about objects belonging to existing part templates to refine the AOG. Our method did not waste much labor of labeling objects that had been well modeled or had strange appearance. Fig. 9 visualizes how the normalized distance, the AOG size, and the normalized distance changed with the number of questions. We found that except for the first question, $\Delta KL(I)$ usually decreased during the QA process. Therefore, we may set a threshold for $\Delta KL(I)$ as the stopping criterion of the active QA. For example, according to Fig. 9, if we set the stop criterion as $\Delta KL(I) < 0.25$, $\Delta KL(I) < 0.22$, and $\Delta KL(I) < 0.20$, then the normalized distance of the learned AOG would be 0.0561, 0.0420, 0.0446, respectively.

5 Summary and Discussion

In this paper, we have proposed a method to bridge and solve the following three crucial issues in computer vision simultaneously.

- Removing noisy representations in conv-layers of a CNN and using an AOG model to reveal the semantic hierarchy of objects hidden in the CNN.
- Enabling people to communicate with neural representations in intermediate conv-layers of a CNN directly for model learning, based on the semantic representation of the AOG.
- Weakly-supervised transferring of object-part representations from a pre-trained CNN to model object parts at the semantic level, which boosts the learning efficiency.

Our method incrementally mines object-part patterns from conv-layers of a pre-trained CNN and uses an AOG to encode the mined semantic hierarchy. The AOG semanticizes neural units in intermediate feature maps of a CNN by associating these units with semantic parts. We have proposed an active QA strategy to learn such an AOG model with limited human annotations. We have tested the proposed method for a total of 37 categories in three benchmark datasets. Our method has outperformed other baselines in the application of part localization. For example, our method with 11 part annotations performed better than fast-RCNN with 60 part annotations on the ILSVRC dataset in Fig. 8.

ACKNOWLEDGMENTS

This work is partially supported by National Natural Science Foundation of China (U19B2043 and 61906120), DARPA XAI Award N66001-17-2-4029, NSF IIS 1423305, and ARO project W911NF1810296.

REFERENCES

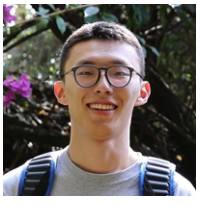
[1] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. *In ICCV*, 2015.
[2] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. *In ECCV*, 2012.
[3] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Interactively co-segmenting topically related images with intelligent scribble guidance. *In IJCV*, 2011.
[4] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*, 2017.
[5] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive training of deformable part models. *In ICCV*, 2011.
[6] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.
[7] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. *In ICCV*, 2015.
[8] 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.
[9] M. Cho, S. Kwak, C. Schmid, and J. Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. *In CVPR*, 2015.
[10] Y. Cong, J. Liu, J. Yuan, and J. Luo. Self-supervised online metric learning with low rank constraint for scene categorization. *In IEEE Transactions on Image Processing*, 22(8):3179–3191, 2013.
[11] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215, 2008.
[12] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. *In CHI*, 2014.
[13] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016.
[14] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *In NIPS*, 2014.
[15] L. Duan, D. Xu, I. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *In CVPR*, 2010.
[16] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*.
[17] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In ICCV*, 2017.
[18] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowledge and information systems*, 35(2):249–283, 2013.
[19] P. W. Gallagher, S. Tang, and Z. Tu. What happened to my dog in that network: unraveling top-down generators in convolutional neural netowrks. *In arXiv:1511.07125v1*, 2015.
[20] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation in backpropagation. *In ICML*, 2015.
[21] R. Girshick. Fast r-cnn. *In ICCV*, 2015.
[22] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *In International Journal of Computer Vision (IJCV)*, 126(5):476–494, May 2018.
[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2016.
[24] A. Holub, P. Perona, and M. C. Burl. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.
[25] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. P. Xing. Harnessing deep neural networks with logic rules. *In ACL*, 2016.
[26] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *In CVPR*, 2017.
[27] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *In ICLR*, 2014.
[28] P. Koh and P. Liang. Understanding black-box predictions via influence functions. *In ICML*, 2017.
[29] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012.
[30] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. *In AAAI*, 2017.
[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 1998.
[32] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR94*, pages 3–12. Springer, 1994.
[33] B. Li, W. Hu, T. Wu, and S.-C. Zhu. Modeling occlusion by discriminative and-or structures. *In ICCV*, 2013.
[34] D. Lin, X. Shen, C. Lu, and J. Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. *In CVPR*, 2015.
[35] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar. Microsoft coco: Common objects in context. *In arXiv:1405.0312v3 [cs.CV]*, 21 Feb 2015.
[36] C. Long and G. Hua. Multi-class multi-annotator active learning with robust gaussian process for visual recognition. *In ICCV*, 2015.
[37] Y. Lu. Unsupervised learning on neural network outputs with application in zero-shot learning. *In IJCAI*, 2016.
[38] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *In CVPR*, 2015.
[39] N. A. H. Mamitsuka et al. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML98)*, volume 1. Morgan Kaufmann Pub, 1998.
[40] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79, 2004.
[41] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.
[42] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? weakly-supervised learning with convolutional neural networks. *In CVPR*, 2015.
[43] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. *In ICCV*, 2015.
[44] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. *In KDD*, 2016.
[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *In IJCV*, 115(3):211–252, 2015.
[46] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. *In CVPR*, 2015.
[47] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In ICCV*, 2017.
[48] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.
[49] K. J. Shih, A. Mallya, S. Singh, and D. Hoiem. Part localization using multi-proposal consensus for fine-grained categorization. *In BMVC*, 2015.
[50] Z. Si and S.-C. Zhu. Learning and-or templates for object recognition and detection. *In PAMI*, 2013.
[51] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015.
[52] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. *In ACCV*, 2014.
[53] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *In arXiv:1312.6034*, 2013.
[54] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.
[55] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal

supervision. *In ICML*, 2014.
[56] Y. C. Song, I. Naim, A. A. Mamun, K. Kulkarni, P. Singla, J. Luo, D. Gildea, and H. Kautz. Unsupervised alignment of actions in video with text descriptions. *In IJCAI*, 2016.
[57] J. Su, D. V. Vargas, and S. Kouichi. One pixel attack for fooling deep neural networks. *In arXiv:1710.08864*, 2017.
[58] Q. Sun, A. Laddha, and D. Batra. Active learning for structured probabilistic models with histogram approximation. *In CVPR*, 2015.
[59] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *In ICLR*, 2014.
[60] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001.
[61] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.
[62] K. Tu, M. Meng, M. W. Lee, T. E. Choe, and S.-C. Zhu. Joint video and text parsing for understanding events and answering queries. *In IEEE MultiMedia*, 2014.
[63] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *In IJCV*, 104(2):154–171, 2013.
[64] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *In CVPR*, 2011.
[65] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *2011 International Conference on Computer Vision*, pages 2524–2531. IEEE, 2011.
[66] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, In California Institute of Technology, 2011.
[67] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *In arXiv 1609.09408*, 2016.
[68] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *In NIPS*, 2014.
[69] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014.
[70] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. *In ECCV*, 2014.
[71] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable graphs on convnets via multi-shot learning. *In AAAI*, 2017.
[72] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Mining object parts from cnns via active question-answering. *In CVPR*, 2017.
[73] Q. Zhang, X. Song, X. Shao, H. Zhao, and R. Shibasaki. Attributed graph mining and matching: An attempt to define and extract soft attributed patterns. *In CVPR*, 2014.
[74] Q. Zhang, X. Song, X. Shao, H. Zhao, and R. Shibasaki. Object discovery: soft attributed graph mining. *In PAMI*, 38(3), 2016.
[75] Q. Zhang, W. Wang, and S.-C. Zhu. Examining cnn representations with respect to dataset bias. *In AAAI*, 2018.
[76] Q. Zhang, Y.-N. Wu, and S.-C. Zhu. Mining and-or graphs for graph matching and object discovery. *In ICCV*, 2015.
[77] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015.
[78] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016.
[79] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8(6):536–544, 2003.
[80] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.
[81] 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.
[82] S. Zhu and D. Mumford. A stochastic grammar of images. *In Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006.

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

Jie Ren is a student at the Shanghai Jiao Tong University. Her research mainly focuses on computer vision and machine learning.

Ge Huang is an undergraduate student at the Shanghai Jiao Tong University. Her research interests include computer vision and machine learning.



Ruiming Cao received the B.S. degree in computer science from the University of California, Los Angeles, in 2017. Now, he is a master student at the University of California, Los Angeles. His research mainly focuses on computer vision.

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

Song-Chun Zhu received a Ph.D. degree from Harvard University in 1996, and is a professor with the Departments of Statistics and Computer Science at UCLA. He has published over 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 Intl 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. He is a Fellow of IEEE since 2011.