# Mining Object Parts from CNNs via Active Question-Answering

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

## Abstract

*Given a convolutional neural network (CNN) that is pre-trained for object classification, this paper proposes to use active question-answering to semanticize neural patterns in conv-layers of the CNN and mine part concepts. For each part concept, we mine neural patterns in the pre-trained CNN, which are related to the target part, and use these patterns to construct an And-Or graph (AOG) to represent a four-layer semantic hierarchy of the part. As an interpretable model, the AOG associates different CNN units with different explicit object parts. We use an active human-computer communication to incrementally grow such an AOG on the pre-trained CNN as follows. We allow the computer to actively identify objects, whose neural patterns cannot be explained by the current AOG. Then, the computer asks human about the unexplained objects, and uses the answers to automatically discover certain CNN patterns corresponding to the missing knowledge. We incrementally grow the AOG to encode new knowledge discovered during the active-learning process. In experiments, our method exhibits high learning efficiency. Our method uses about $1/6$–$1/3$ of the part annotations for training, but achieves similar or better part-localization performance than fast-RCNN methods.*

## 1. Introduction

Convolutional neural networks (CNNs) [17, 16] have been trained to achieve near human-level performance on object detection. However, CNN methods still face two issues in real-world applications. First, many visual tasks require detailed interpretations of object structures for hierarchical understanding of objects (*e.g.* part localization and parsing). This is beyond the detection of object bounding boxes. Second, weakly-supervised learning is also a difficult problem for CNNs. Unlike data-rich applications (*e.g.* pedestrian/vehicle detection), many tasks require modeling certain object parts on the fly. For example, people may hope to use only a few examples to quickly teach a robot how to grasp a certain type of object parts for an occasional task.

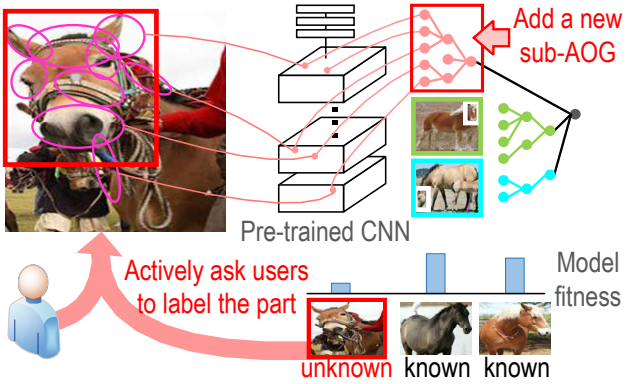


Figure 1. Semanticizing knowledge in a pre-trained CNN via active question-answering (QA). We mine latent patterns from the CNN to explain certain object parts, and organize such patterns into a semantic hierarchy. Our method automatically identifies objects whose parts cannot be explained by part templates in the current AOG, asks about the objects, and uses the answers to mine patterns from these objects. The mined patterns represent new part templates, and are organized as new branches in the AOG.

In this study, we propose a new strategy to model a certain object part using a few part annotations, *i.e. using an active question-answering (QA) process to mine latent patterns that are related to the part from a pre-trained CNN. We use an And-Or graph (AOG) as an interpretable model to associate these patterns with the target part.*

We develop our method based on the following three ideas: **1)** When a CNN is pre-trained using objects of a category with object-box annotations, most appearance knowledge of the target category may have been encoded in conv-layers of the CNN. **2)** Our task is to mine latent patterns from complex neural activations in the conv-layers. Each pattern individually acts as a detector of a certain region of an object. We use the mined regional patterns to construct an AOG to represent the target part. **3)** Because the AOG represents the part's neural patterns with clear semantic hierarchy, we can start an active QA to incrementally grow new AOG branches to encode new part templates, so as to enrich the knowledge in the AOG.

More specifically, during the active QA, the computer discovers objects whose neural activations cannot be explained by the current AOG and asks human users for su-

pervision. We use the answers to grow new AOG branches for new part templates given in answers. Active QA makes the part knowledge efficiently learned with very limited human supervision.

**CNN generalization:** Before we introduce inputs and outputs of our QA-based learning, we clarify our target of CNN generalization, *i.e.* growing semantic AOGs to explain semantic hierarchy hidden within the conv-layers of a pre-trained CNN.

As shown in Fig 2, the AOG has four layers, which encode a clear semantic hierarchy ranging from *semantic part*, *part templates*, *latent patterns*, to *CNN units*. In the AOG, we use AND nodes to represent compositional regions of a part, and use OR nodes to encode a list of alternative template/deformation candidates for a local region. The top *part* node (OR node) uses its children to represent a number of template candidates for the part. Each *part template* (AND node) in the second layer has a number of children as latent patterns to represent its constituent regions (*e.g.* an eye in the face part). Each *latent pattern* in the third layer (OR node) naturally corresponds to a certain range of units within a CNN conv-slice. We select a *CNN unit* within this range to account for geometric deformation of the latent pattern.

Note that we do not further fine-tune the original convolutional weights within the pre-trained CNN. This allows us to continuously grow AOGs for different parts, without the risk of model drifting.

**Inputs and outputs of QA-based learning:** Given a pre-trained CNN and its training samples (*i.e.* object images without any part annotations), we incrementally grow AOG branches for the target part. In each step of QA, we let the CNN use the current AOG to localize the target part among all the unannotated images. Our method actively identifies object images, whose parts cannot be well explained by the AOG. Among all the unexplained objects, our method predicts the potential gain of asking about each unexplained object, and thus determines a best sequence of questions for QA. As in Fig. 3, the user is able to give five types of answers to explicitly guide the AOG growth. Given each specific answer, the computer may refine an existing part template or mine latent patterns to construct a new AOG branch for a new part template.

**Learning from weak supervision:** Unlike previous end-to-end batch learning, there are two mechanisms to ensure the stability of weakly-supervised learning. 1) We transfer patterns in a pre-trained object-level CNN to the target part concept, instead of learning all knowledge from scratch. These patterns are supposed to consistently describe the same part region among different object images. The pattern-mining process purifies the CNN knowledge for better representation of the target part. 2) We use active QA to collect training samples, in order to avoid wasting human labor of annotating object parts that can be well explained by the AOG.

We use object-level annotations for pre-training, considering the following two facts: 1) Only a few datasets [6, 42] provide part annotations, and most benchmark datasets [13, 26, 20] mainly have annotations of object bounding boxes. 2) More crucially, different applications may focus on different object parts, and it is impractical to annotate a large number of parts for each specific task.

**Contributions:** Contributions of this study can be summarized as follows. 1) We mine and represent latent patterns hidden in a pre-trained CNN using an AOG. The AOG representation enables the QA *w.r.t* the semantic hierarchy of the target part. 2) We propose to use active QA to explicitly learn the semantics of each AOG branch, which ensures a high learning efficiency. 3) In experiments, our method exhibits superior performance to other baselines in terms of weakly-supervised part localization. For example, our methods with 11 part annotations outperformed fast-RCNNs with 60 annotations in Fig. 5.

## 2. Related work

**Passive CNN visualization vs. active CNN semanticization:** In order to explore the hidden semantics in the CNN, many studies visualized and analyzed patterns of CNN units [44, 23, 33, 1, 21].

However, from the perspective of semanticizing CNN units, CNN visualization and our active QA go in two opposite directions. Given a certain unit in a pre-trained CNN, the former mainly visualizes the potential visual pattern of the unit passively. However, the latter focuses on a more fundamental problem in real applications, *i.e. given a query of modeling/refining certain object parts, can we efficiently discover certain patterns that are related to the part concepts, within the pre-trained CNN from its complex neural activations?* Given CNN feature maps, Zhou *et al.* [48, 49] discovered latent "scene" semantics. Simon *et al.* discovered objects [30] from CNN activations in an unsupervised manner, and learned part concepts in a supervised fashion [32]. AOG structure is suitable for representing semantic hierarchy of objects [50, 29], and [46] used an AOG to represent the CNN. In this study, we used semantic-level QA to incrementally mine part semantics from the CNN and grow the AOG. Such a "white-box" representation of the CNN knowledge also guided further active QA.

**Unsupervised/active learning:** Many methods have been developed to learn object models in an unsupervised or weakly supervised manner. Methods of [5, 36, 47, 31] learned with image-level annotations without labeling object bounding boxes. [11, 7] did not require any annotations during the learning process. [8] collected training data online from videos to incrementally learn models. [12, 37] discovered objects and identified actions

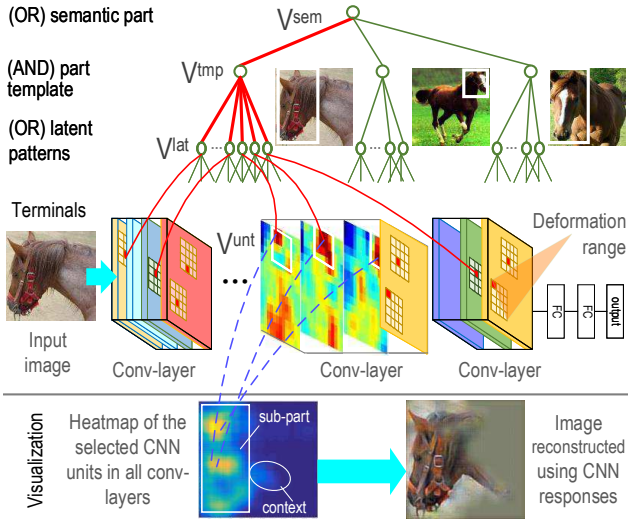


Figure 2. And-Or graph grown on the pre-trained CNN as a semantic branch. The AOG associates certain CNN units with certain image regions. The red lines indicate the parse graph.

from language Instructions and videos. Inspired by active learning [38, 41, 22], the idea of learning from question-answering has been used to learn object models [9, 27, 39]. Branson *et al.* [4] 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 semanticize the CNN and transfer the hidden knowledge to the AOG.

**Modeling "objects" vs. modeling "parts" in un-/weakly-supervised learning:** In the scope of unsupervised learning and/or weakly-supervised learning, modeling parts is usually more challenging than modeling entire objects. Given image-level labels (without object bounding boxes), object discovery [24, 30, 25] and co-segmentation [3] can be achieved by identifying common foreground patterns from complex background. In addition, there are some strong prior knowledges for object discovery, such as closed boundaries and common object structures.

In contrast, to the best of our knowledge, there is no mechanism to distinguish a certain part concept from other parts of the same object. It is because 1) all the parts represent common foreground patterns among objects; 2) some parts (*e.g.* the abdomen) do not have shape boundaries to identify their shape extent. Thus, up to now, people mainly extract implicit middle-level part patches [35], but it is difficult to capture explicit semantic meanings of these parts.

## 3. Preliminaries: And-Or graph on a CNN

In this section, we briefly introduce an AOG, which is designed to explain the latent semantic structure within the CNN. As shown in Fig. 2, an AOG has four layers, *i.e. semantic part* (OR node), *part template* (AND node), *latent pattern* (OR node), and *CNN unit*. In the AOG, an OR node encodes a number of alternative candidates as children. An AND node uses its children to represent its constituent regions. For example, 1) the semantic part (OR node) encodes a number of template candidates for the part as children. 2) Each part template (AND node) encodes the spatial relationship between its children latent patterns (each child corresponds to a constituent region or a contextual image region). 3) Each latent pattern (OR node) takes a number of CNN units in a certain conv-slice as children to represent alternative deformation candidates of the pattern (the pattern may appear in different image positions).

Given an image $I$[1], we use the CNN to compute neural activations on $I$ in its conv-layers, and then use the AOG for *hierarchical part parsing*. *I.e.* we use the AOG to semanticize the neural activations and localize the target part.

We use $V^{\text{sem}}$, $V^{\text{tmp}} \in \Omega^{\text{tmp}}$, $V^{\text{lat}} \in \Omega^{\text{lat}}$, and $V^{\text{unt}} \in \Omega^{\text{unt}}$, respectively, to denote nodes at the four layers. During the parsing procedure, 1) the top node $V^{\text{sem}}$ selects a part template $\hat{V}^{\text{tmp}}$ to explain the whole part; 2) $\hat{V}^{\text{tmp}}$ let its children latent patterns use their own parsing configurations to vote for $\hat{V}^{\text{tmp}}$'s position, thereby parsing an image region for $\hat{V}^{\text{tmp}}$; 3) each latent pattern $V^{\text{lat}} \in Child(\hat{V}^{\text{tmp}})$ selects a CNN-unit child with a certain deformation range $\hat{V}^{\text{unt}} \in Child(V^{\text{lat}})$ as a stand-in of the pattern.

We define a parse graph $pg_I$ to denote the parsing configurations. As the red lines in Fig. 2, $pg_I$ is a tree of image regions that are assigned to AOG nodes, $pg = \{\Lambda_{I,V^{\text{sem}}}\} \cup \{\Lambda_{I,\hat{V}^{\text{tmp}}}\} \cup_{V^{\text{lat}} \in Child(\hat{V}^{\text{tmp}})} \{\Lambda_{I,V^{\text{lat}}}\}$, where for each node $V$, $\Lambda_{I,V}$ denotes the image region that is parsed for $V$. We use $\Lambda_V$ to simplify the notation of $\Lambda_{I,V}$, without ambiguity.

We design an inference score $S_I(V|\Lambda_V)$ for each node $V$ to measure the compatibility between a given region $\Lambda_V$ and $V$ (as well as the AOG branch under $V$). Thus, hierarchical part parsing on a given image $I$ can be achieved in a bottom-up manner. We compute inference scores for CNN units, then propagate the scores to latent patterns and part templates, and finally obtain the score of the top node as the overall inference score $\mathbf{L}(I, \boldsymbol{\theta})$. We determine the parse graph $\hat{pg}_I$ that maximizes the overall score:

$$\mathbf{L}(I, \boldsymbol{\theta}) = S_I(V^{\text{sem}}|\Lambda_{V^{\text{sem}}}), \quad \hat{pg}_I = \text{argmax}_{pg_I} \mathbf{L}(I, \boldsymbol{\theta})|_{pg_I} \quad (1)$$

where $\boldsymbol{\theta}$ denotes the AOG parameters.

**Terminal nodes (CNN units):** Each terminal node under a latent pattern takes a certain square within a certain conv-slice, which represents deformation candidates of the latent pattern. Each $V^{\text{unt}}$ corresponds to a fixed image region $\Lambda_{V^{\text{unt}}}$. *I.e.* we propagate $V^{\text{unt}}$'s receptive field to the image plane, and use the final field as $\Lambda_{V^{\text{unt}}}$. The score of $V^{\text{unt}}$, $S_I(V^{\text{unt}})$[2], is designed to describe the neural response

[1] Considering CNN's superior performance in object detection, as in [6], we regard object detection and part localization as two separate processes for evaluation. Thus, we crop $I$ to only contain the object and resize $I$ for CNN inputs to simplify the scenario of learning for part localization.

[2] Please see [46] for detailed settings.

value of $V^{\text{unt}}$ and its local deformation level.

**OR nodes:** Given children's parsing configurations of an OR node (either $V^{\text{sem}}$ or $V^{\text{lat}}$), $V^O$ selects the child $\hat{V}$ with the highest score, and propagates $\hat{V}$'s parsing result to $V^O$:

$$S_I(V^O|\hat{\Lambda}_{V^O}) = \max_{V \in Child(V^O)} S_I(V|\hat{\Lambda}_V), \quad \hat{\Lambda}_{V^O} \leftarrow \hat{\Lambda}_{\hat{V}} \tag{2}$$

**AND nodes:** Given parsing results of a part template $V^{\text{tmp}}$'s children latent patterns, we parse an image region for $V^{\text{tmp}}$, which maximizes its score.

$$S_I(V^{\text{tmp}}|\Lambda_{V^{\text{tmp}}}) = \sum_{V^{\text{lat}} \in Child(V^{\text{tmp}})} \left[ S_I(V^{\text{lat}}|\hat{\Lambda}_{V^{\text{lat}}}) + S^{\text{inf}}(\Lambda_{V^{\text{tmp}}}|\hat{\Lambda}_{V^{\text{lat}}}) \right]$$
$$\hat{\Lambda}_{V^{\text{tmp}}} = \text{argmax}_{\Lambda_{V^{\text{tmp}}}} S_I(V^{\text{tmp}}|\Lambda_{V^{\text{tmp}}}) \tag{3}$$

where $S^{\text{inf}}(\Lambda_{V^{\text{tmp}}}|\hat{\Lambda}_{V^{\text{lat}}})$[2] measures the spatial compatibility between parsing configurations of $\Lambda_{V^{\text{tmp}}}$ and $\hat{\Lambda}_{V^{\text{lat}}}$ on $I$.

**AOG construction:** The method for constructing an AOG based on part annotations was proposed in [46]. We briefly summarize this method as follows. Let $\mathbf{I}$ denote a set of cropped object images of a category. Among all objects in $\mathbf{I}$, only a small number of objects, $\mathbf{I}^{\text{ant}} = \{I_i | i = 1, 2, \ldots, m\} \subset \mathbf{I}$, have annotations of the target part. For each annotated object $I \in \mathbf{I}^{\text{ant}}$, we label two terms $(\Lambda^*_{I,V^{\text{sem}}}, V_I^{\text{tmp}*})$. $\Lambda^*_{I,V_I^{\text{sem}}}$ denotes the ground-truth bounding box of the part, and $V_I^{\text{tmp}*}$ specifies the true choice of the part template for the part in $I$. For the first two layers of the AOG, the AOG is set to only contain the part templates that appear in part annotations.

Thus, AOG construction is to mine a total of $n$ different latent patterns for each part template $V^{\text{tmp}}$, where $n$ is a hyper-parameter. For each latent pattern $V^{\text{lat}}$, parameters $\boldsymbol{\theta}_{V^{\text{lat}}} \subset \boldsymbol{\theta}$ mainly determine 1) $V^{\text{lat}}$'s deformation range and 2) the prior displacement from $V^{\text{tmp}}$ to $V^{\text{lat}}$. The estimation of $\boldsymbol{\theta}_{V^{\text{lat}}}$ can be roughly written as[2]

$$\max_{\boldsymbol{\theta}} \left\{ \underset{I \in \mathbf{I}_{V^{\text{tmp}}}}{\text{mean}} S_I(V_I^{\text{tmp}*}|\Lambda_{V_I^{\text{tmp}*}} = \Lambda^*_{I,V^{\text{sem}}}) + \underset{I' \in \mathbf{I}}{\text{mean}} S_I^{\text{local}}(V_I^{\text{tmp}*}) \right\} \tag{4}$$

where $\mathbf{I}_{V^{\text{tmp}}} = \{I \in \mathbf{I}^{\text{ant}} | V_I^{\text{tmp}*} = V^{\text{tmp}}\}$. Compared to $S_I(V_I^{\text{tmp}*}|\Lambda_{V_I^{\text{tmp}*}})$, $S_I^{\text{local}}(V_I^{\text{tmp}*}) = \sum_{V^{\text{lat}} \in Child(V_I^{\text{tmp}*})} S_I(V^{\text{lat}}|\hat{\Lambda}_{V^{\text{lat}}})$ is an inference score that ignores the pairwise spatial compatibility.

# 4. Learning from active question-answering

## 4.1. Overview of knowledge mining

Compared to conventional batch learning, our method uses a more efficient learning strategy, which allows the computer to actively detect blind spots in its knowledge system and ask questions. In general, knowledge blind spots in the AOG include 1) neural-activation patterns in the CNN that have not been modeled and 2) the inaccuracy of the existing latent patterns. We assume that the unexplained neural patterns potentially reflect new part templates, while the inaccurate latent patterns correspond to the sub-optimally modeled part templates.

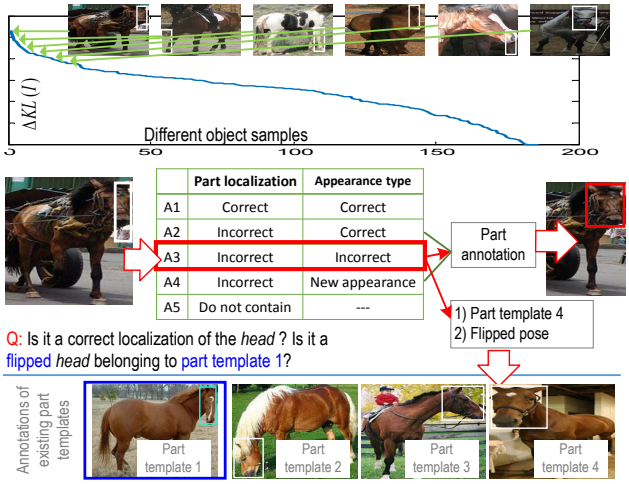


Figure 3. Illustration of the QA process. (top) We sort and select objects. (bottom) We show questions asked for each target object.

Because an AOG is an interpretable representation that explicitly encodes object parts, we can represent blind spots of the knowledge using linguistic description. We use a total of five types of answers to explicitly project these blind spots onto specific semantic details of objects. In this way, the computer selects and asks a series of questions. Based on the answers, the AOG incrementally grows new semantic branches to explain new part templates and refine AOG branches of existing part templates.

The computer repeats the following process in each QA step. Let $\mathbf{I}$ denote a set of object images. As shown in Fig. 3, the computer first uses the current AOG to localize object parts on all unannotated objects in $\mathbf{I}$. Based on localization results, the computer selects and asks about an object $I$, from which the computer believes it can obtain the most information gain. A question $q = (I, \hat{V}^{\text{tmp}}, \hat{\Lambda}_{V^{\text{sem}}})$ requires people to determine whether the computer determines the correct part template $\hat{V}^{\text{tmp}}$ and accurately localizes the part in $\hat{\Lambda}_{V^{\text{sem}}}$, and expects one of the following answers.

**Answer 1:** the part detection is correct. **Answer 2:** the computer chooses the true template for the part in the parse graph, but it does not accurately localizes the target 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 object. In addition, in case of receiving Answers 2–4, the computer will ask people to annotate the target part. In case of getting Answer 3, the computer will require people to specify the part template, as well as whether the object is flipped. Then, our method uses the new annotation to refine (for Answers 2–3) or create (for Answer 4) the AOG branch for the annotated part template based on Eq. (4).

## 4.2. Question ranking

The core of the QA process is to select a sequence of objects that reduce the AOG uncertainty the most. There-

fore, in this section, we design a loss function to measure the incompatibility between the AOG knowledge and the actual part appearance in the object samples. We predict the potential gain (decrease of the loss) of asking about each object. Objects with large gains usually correspond to unexplained or not well explained CNN neural activations. Note that annotating the part in an object may also help explain 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 leads to the most gain in each step.

For each object $I \in \mathbf{I}$, 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. $y \in \{+1, -1\}$ is a label indicating whether $I$ contains the target part. The current AOG estimates the probability of object $I$ containing the target part as $\mathbf{Q}(y=+1|I) = \frac{1}{Z}\exp[\beta \mathbf{L}(I, \boldsymbol{\theta})]$, where $Z$ and $\beta$ are scaling parameters (see Section 5.1 for details); $\mathbf{Q}(y = -1|I) = 1 - \mathbf{Q}(y = +1|I)$. Let $\mathbf{I}^{\text{ask}} \subset \mathbf{I}$ denotes the objects that have been asked during previous QA. For each asked object $I \in \mathbf{I}^{\text{ask}}$, we set its prior distribution $\mathbf{P}(y=+1|I)=1$ if $I$ contains the target part according to previous answers; $\mathbf{P}(y = +1|I) = 0$ otherwise. For each un-asked object $I \in \mathbf{I} \setminus \mathbf{I}^{\text{ask}}$, we set its prior distribution based on statistics of previous answers, $\mathbf{P}(y=+1|I) = \text{mean}_{I' \in \mathbf{I}^{\text{ask}}} \mathbf{P}(y=+1|I')$. Therefore, we formulate the loss function as the KL divergence between the prior distribution $\mathbf{P}$ and the estimated distribution $\mathbf{Q}$, and seek to minimize the KL divergence via QA.

$$\begin{aligned} Loss = \mathbf{KL}(\mathbf{P}\|\mathbf{Q}) &= \sum\nolimits_{I\in\mathbf{I}} \sum\nolimits_{y} \mathbf{P}(y, I) \log \frac{\mathbf{P}(y, I)}{\mathbf{Q}(y, I)} \\ &= \lambda \sum\nolimits_{I\in\mathbf{I}} \sum\nolimits_{y} \mathbf{P}(y|I) \log \frac{\mathbf{P}(y|I)}{\mathbf{Q}(y|I)} \end{aligned} \tag{5}$$

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

In fact, both the prior distribution $\mathbf{P}$ and the estimated distribution $\mathbf{Q}$ keep changing during the QA process. Let us assume that the computer selects object $\tilde{I} \in \mathbf{I} \setminus \mathbf{I}^{\text{ask}}$ and that people annotate its part. The annotation would encode the part knowledge of $\tilde{I}$ into the AOG and greatly change the estimated distribution for objects that are similar to $\tilde{I}$. For each object $I' \in \mathbf{I}$, we predict its estimated distribution after the new part annotation as

$$\begin{aligned} \tilde{\mathbf{Q}}(y=+1|I') &= \frac{1}{Z}\exp[\beta \mathbf{L}(I', \boldsymbol{\theta}^{\text{new}})|_{\tilde{I}}] \\ \mathbf{L}(I', \boldsymbol{\theta}^{\text{new}})|_{\tilde{I}} &= \mathbf{L}(I', \boldsymbol{\theta}) + \Delta\mathbf{L}(\tilde{I}, \boldsymbol{\theta}) e^{-\alpha \cdot dist(I', \tilde{I})} \end{aligned} \tag{6}$$

where $\mathbf{L}(I', \boldsymbol{\theta}^{\text{new}})|_{\tilde{I}}$ indicates the predicted inference score of $I'$ when we 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 $\tilde{I}$. We estimate the score increase of $\tilde{I}$ as $\Delta\mathbf{L}(\tilde{I}, \boldsymbol{\theta}) = \text{mean}_{I\in\mathbf{I}^{\text{ant}}} \mathbf{L}(I, \boldsymbol{\theta}) - \mathbf{L}(\tilde{I}, \boldsymbol{\theta})$. $\alpha$ 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 CNN 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[3]. Thus, $\exp[\alpha \cdot dist(I', \tilde{I})]$ measures the similarity between $I'$ and $\tilde{I}$. In addition, if $I'$ and $\tilde{I}$ are assigned with different part templates by the current AOG, we may ignore the similarity between $I'$ and $\tilde{I}$ (by setting an infinite distance between them) to achieve better performance. Based on the prediction in Eq. (6), we can predict the changes of the KL divergence after the new annotation on $\tilde{I}$ as

$$\Delta\mathbf{KL}(\tilde{I}) = \lambda \sum\nolimits_{I\in\mathbf{I}} \sum\nolimits_{y} \mathbf{P}(y|I) \log \frac{\tilde{\mathbf{Q}}(y|I)}{\mathbf{Q}(y|I)} \tag{7}$$

Thus, in each step, the computer selects and asks about the object that maximize the decrease of the KL divergence.

$$\hat{I} = \text{argmax}_{I\in\mathbf{I}\setminus\mathbf{I}^{\text{ask}}} \Delta\mathbf{KL}(I) \tag{8}$$

**QA implementations:** In the beginning, for each object $I$, we initialize its prior distribution as $\mathbf{P}(y=+1|I)=1$ and its estimated distribution as $\mathbf{Q}(y = +1|I) = 0$. Then, the computer selects and asks about an object $\hat{I}$ based on Eq. (8). We use the answer to update $\mathbf{P}$. If new object parts are labeled during the QA process, we apply Eq. (4) to update the AOG. More specifically, if people label a new part template, the AOG 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, the new AOG can provide the new distribution $\mathbf{Q}$. In later steps, the computer repeats the above QA procedure of Eq. (8) and Eq. (4) to ask more questions.

## 5. Experiments

### 5.1. Implementation details

We used the 16-layer VGG network (VGG-16) [34], which was pre-trained using 1.3M images in the ImageNet ILSVRC 2012 dataset [26] with a loss for 1000-category classification. Then, in order to learn part concepts for each category, we further fine-tune the VGG-16 using object images in this category based on the loss for classifying target objects and background. The VGG-16 contains a total of 13 conv-layers and 3 fully connected layers. We selected the last 9 conv-layers as valid conv-layers. We extracted CNN units from these layers to build the AOG.

In our method, three parameters were involved in active QA, *i.e.* $\alpha$, $\beta$, and $Z$. Considering that most object images contained the target part in real applications, we ignored the small probability of $\mathbf{P}(y=-1|I)$ in Eq. (7) to simplify the computation. As a result, the parameter $Z$ was eliminated in the computation of Eq. (7), and the parameter $\beta$ acted as a constant weight for $\Delta\mathbf{KL}(\tilde{I})$, which did not affect object selection in Eq. (8). Therefore, in our experiments, we set $\alpha=4.0$, which achieved the best performance.

[3] $\mathbf{M}_{ii} \propto \exp[\text{mean}_{I\in\mathbf{I}} S_I(V_i^{\text{unt}})]$, where $V_i^{\text{unt}}$ is the CNN unit corresponding to the $i$-th element of $\mathbf{f}_{I'}$.

| 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 |

Table 1. Average number of children of AOG nodes

### 5.2. Datasets

We used three benchmark datasets to test our method, *i.e.* the PASCAL VOC Part Dataset [6], the CUB200-2011 dataset [42], and the ILSVRC 2013 DET Animal-Part dataset [46]. Just like in most part-localization studies [6, 46], we selected animal categories, which prevalently contain non-rigid shape deformation, to test part-localization performance. *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. Like in [4, 32, 46], we ignored species labels and regarded all these images as a single bird category. The ILSVRC 2013 DET Animal-Part dataset [46] was proposed for part localization. It consists of 30 animal categories among all the 200 categories for object detection in the ILSVRC 2013 DET dataset [26].

### 5.3. Baselines

We compared the proposed method with the following thirteen baselines. We designed the first two baselines based on the Fast-RCNN [14]. Note that we fine-tuned the fast-RCNN with a loss for detecting a single class/part from background, rather than for multi-class/part detection, for a fair comparison. In the first baseline, namely *Fast-RCNN (1 ft)*, we directly fine-tuned the VGG-16 network using part annotations to detect parts on well cropped objects. Then, to enable a fair comparison, we conducted the second baseline based on two-stage fine-tuning, namely *Fast-RCNN (2 fts)*. *The Fast-RCNN (2 fts) first fine-tuned the VGG-16 network using a large number of object-box annotations* (more than part annotations) *in the target category, and then fine-tuned the VGG-16 using a few part annotations.*

The third baseline was proposed by [32], namely *CNN-PDD*. *CNN-PDD* selected a conv-slice in a CNN (pre-trained using ImageNet ILSVRC 2012 dataset) to represent and localize the part on well cropped objects. Then, we slightly extended [32] as the fourth baseline *CNN-PDD-ft*. *CNN-PDD-ft* fine-tuned the VGG-16 using object-box annotations, and then applied [32] to the VGG-16 for learning.

The fifth and sixth baselines were the strongly supervised DPM (*SS-DPM-Part*) [2] and the technique in [18] (*PL-DPM-Part*), respectively. They trained DPMs using part annotations for part localization. We used the graphical model proposed in [6] as the seventh baseline, namely *Part-Graph*. The eighth baseline was the interactive learning of DPMs for part localization [4] (*Interactive-DPM*).

Without many training samples, "simple" methods are usually insensitive to the over-fitting problem. Thus, we designed the last four baselines as follows. We used the VGG-16 network that was fine-tuned using object-box annotations, and collected image patches from a cropped object based on the selective search [40]. We used the VGG-16 to extract *fc7* features from each image patch. The two baselines (*i.e. fc7+linearSVM* and *fc7+RBF-SVM*) used a linear SVM and a RBF-SVM, respectively, to detect the target part. The other baselines *VAE+linearSVM* and *CoopNet+linearSVM* used features of the VAE network [15] and the CoopNet [43], respectively, instead of *fc7* features, for part detection.

Finally, the last baseline is the learning of AOGs [46] without QA (*AOG w/o QA*). We annotated parts and part templates on randomly selected objects.

In fact, both object annotations and part annotations are used to learn models in all the thirteen baselines (including those without fine-tuning).

### 5.4. Evaluation metric

It has been discussed in [6, 46] that a fair evaluation of part localization requires removing the factors of object detection. Therefore, we used ground-truth object bounding boxes to crop objects from the original images to produce testing images. Given an object image, object/part detection methods (*e.g. Fast-RCNN (1 ft)*, *Part-Graph*, and *SS-DPM-Part*) usually estimate several bounding boxes for the part with different confidence values. As in [32, 6, 24, 46], the task of part localization takes the most confident bounding box per image as the result. Given part-localization results on objects of a category, we applied the *normalized distance* [32] and the *percentage of correctly localized parts* (PCP) [45, 28, 19] to evaluate part localization. For the normalized distance, we computed 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 PCP, we used the typical metric of "$IoU \geq 0.5$" [14] to identify correct part localizations.

### 5.5. Experimental results

We tested our method on the ILSVRC 2013 DET Animal-Part dataset, the Pascal VOC Part dataset, and the CUB200-2011 dataset. We learned AOGs for parts of the head, the neck, and the nose/muzzle/beak of the six animal categories 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[4] of each category. Because the head is shared by all categories in the two

[4]It is the "forehead" part for birds in the CUB200-2011 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 [18] | 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 [6] | 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 [32] | 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 [32] | 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) [14] | 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) [14] | 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** |

| | Part Annot. | Obj.-box finetune | 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 [18] | 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 [6] | 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 [32] | 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 [32] | 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) [14] | 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) [14] | 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** |

Table 2. Normalized distance of part localization on the ILSVRC 2013 DET Animal-Part dataset. The second column shows the number of part annotations for training. The third 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*).

| | Obj.-box finetune | Part Annot. | #Q | Normalizaed distance |
|---|---|---|---|---|
| SS-DPM-Part [2] | No | 60 | – | 0.2504 |
| PL-DPM-Part [18] | No | 60 | – | 0.3215 |
| Part-Graph [6] | No | 60 | – | 0.3697 |
| fc7+linearSVM | Yes | 60 | – | 0.2786 |
| fc7+RBF-SVM | Yes | 60 | – | 0.3360 |
| Interactive-DPM [4] | No | 60 | – | 0.2011 |
| CNN-PDD [32] | No | 60 | – | 0.2446 |
| CNN-PDD-ft [32] | Yes | 60 | – | 0.2694 |
| Fast-RCNN (1 ft) [14] | No | 60 | – | 0.3105 |
| Fast-RCNN (2 fts) [14] | Yes | 60 | – | 0.1989 |
| AOG w/o QA [46] | Yes | **20** | – | 0.1084 |
| Ours | Yes | **10** | 28 | **0.0626** |
| Ours | Yes | **20** | 112 | **0.0434** |

Table 3. Part localization performance on the CUB200-2011 dataset. See Table 2 for the introduction of the 2nd and 3rd columns. The 4rd column shows the number of questions for training. The fourth 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.

datasets, we selected the head as the target part to enable a fair comparison. We did not train the human annotators. During the active QA process, boundaries between two part templates were often very vague, so an annotator could assign a part with either part templates.

In Table 1, we illustrated how the AOG grew when people annotated more parts during the question-answering process. We computed the average number of children for each node in different AOG layers based on the AOGs learned from the PASCAL VOC Part Dataset. It shows that the AOG mainly grew itself by adding new AOG branches for new part templates. The refinement of an AOG branch for an existing part template did not significantly change the size of this AOG branch.

| | Method | Annot. | #Q | bird | cat | cow | dog | horse | sheep | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|
| Head | Fast-RCNN (1 ft) [14] | **10** | – | 0.326 | 0.238 | 0.283 | 0.286 | 0.319 | 0.354 | 0.301 |
| | Fast-RCNN (2 fts) [14] | **10** | – | 0.233 | 0.196 | 0.216 | 0.206 | 0.253 | 0.286 | 0.232 |
| | Fast-RCNN (1 ft) [14] | 20 | – | 0.352 | **0.131** | 0.275 | 0.189 | 0.293 | 0.252 | 0.249 |
| | Fast-RCNN (2 fts) [14] | 20 | – | 0.176 | 0.132 | 0.191 | 0.171 | 0.231 | **0.189** | 0.182 |
| | Fast-RCNN (1 ft) [14] | 30 | – | 0.285 | 0.146 | 0.228 | 0.141 | 0.250 | 0.220 | 0.212 |
| | Fast-RCNN (2 fts) [14] | 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) [14] | **10** | – | 0.251 | 0.333 | 0.310 | 0.248 | 0.267 | 0.242 | 0.275 |
| | Fast-RCNN (2 fts) [14] | **10** | – | 0.317 | 0.335 | 0.307 | 0.362 | 0.271 | 0.259 | 0.309 |
| | Fast-RCNN (1 ft) [14] | 20 | – | 0.255 | 0.359 | 0.241 | 0.281 | 0.268 | 0.235 | 0.273 |
| | Fast-RCNN (2 fts) [14] | 20 | – | 0.260 | 0.289 | 0.304 | 0.297 | 0.255 | 0.237 | 0.274 |
| | Fast-RCNN (1 ft) [14] | 30 | – | 0.288 | 0.324 | 0.247 | 0.262 | 0.210 | 0.220 | 0.258 |
| | Fast-RCNN (2 fts) [14] | 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/Beek | Fast-RCNN (1 ft) [14] | **10** | – | 0.446 | 0.389 | 0.301 | 0.326 | 0.385 | 0.328 | 0.363 |
| | Fast-RCNN (2 fts) [14] | **10** | – | 0.447 | 0.433 | 0.313 | 0.391 | 0.338 | 0.350 | 0.379 |
| | Fast-RCNN (1 ft) [14] | 20 | – | 0.425 | 0.372 | 0.260 | 0.303 | 0.334 | 0.279 | 0.329 |
| | Fast-RCNN (2 fts) [14] | 20 | – | 0.419 | 0.351 | 0.289 | 0.249 | 0.296 | 0.293 | 0.316 |
| | Fast-RCNN (1 ft) [14] | 30 | – | 0.462 | 0.336 | 0.242 | 0.260 | 0.247 | 0.257 | 0.301 |
| | Fast-RCNN (2 fts) [14] | 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** |

Table 4. Part localization on the Pascal VOC Part dataset. The third and fourth columns show the number of part annotations and the average number of questions for training.

Fig. 4 shows the part localization results based on AOGs and visualizes the content of latent patterns in the AOG based on the technique of [10]. Tables 2, 4, and 3 compares part-localization performance of different baselines on the ILSVRC 2013 DET Animal-Part dataset, the Pascal VOC Part dataset, and the CUB200-2011 dataset, respectively. Tables 4, and 3 show both the number of part annotations and the number of questions. Fig. 5 shows the performance of localizing the head part on the PASCAL VOC Part Dataset, when people annotated different number of parts for training. Table 5 shows the results evaluated by the PCP. In particular, the method of *Ours+fastRCNN* combined our method and the fast-RCNN to refine part-localization re-

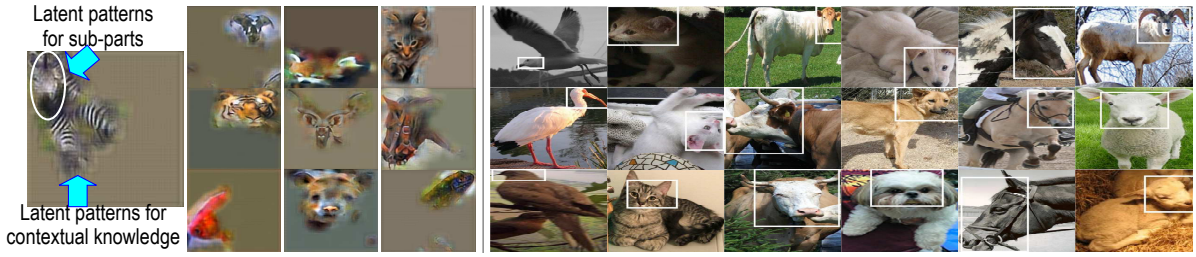


Figure 4. Visualization of latent patterns in AOGs for the head part (left) and part localization results based on AOGs (right).

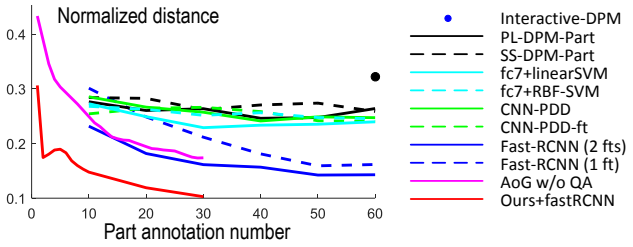


Figure 5. Part localization performance on the Pascal VOC Part dataset.

| | # of part annotations | Performance |
|---|---|---|
| SS-DPM-Part [2] | 60 | 7.2 |
| PL-DPM-Part [18] | 60 | 6.7 |
| Part-Graph [6] | 60 | 11.0 |
| fc7+linearSVM | 60 | 13.5 |
| fc7+RBF-SVM | 60 | 9.5 |
| VAE+linearSVM [15] | 30 | 6.7 |
| CoopNet+linearSVM [43] | 30 | 5.6 |
| Fast-RCNN (1 ft) [14] | 30 | 34.5 |
| Fast-RCNN (2 fts) [14] | 30 | 45.7 |
| Ours+fastRCNN | **10** | 33.0 |
| Ours+fastRCNN | **20** | **47.2** |
| Ours+fastRCNN | **30** | **50.5** |

Table 5. Part localization performance evaluated using the PCP on the Pascal VOC Part dataset.

sults[5]. Our method worked with about $1/6$–$1/2$ part annotations, but exhibited superior performance.

## 6. Justification of the methodology

There are three reasons for the superior performance of our method. First, **richer information**: the latent patterns in the AOG were pre-fine-tuned using a large number of object images in the category, instead of being learned from a few part annotations. Thus, the knowledge contained in these patterns was far beyond that in the objects with part annotations.

Second, **less model drift:** Instead of learning/fine-tuning new CNN parameters, our method just used limited part annotations to mine "reliable" patterns and organize their spatial relationships to represent the part concept. In addition, during active QA, the computer usually selected and asked about objects with common object poses based on Eq. (6), *i.e.* objects sharing some common latent patterns with many other objects. Thus, the learned AOG suffered less from the over-fitting/model-drift problem.

Third, **high QA efficiency:** Our QA process balanced both the commonness of a part template and the modeling quality of this part template in Eq. (6). In early steps of QA, the computer was prone to asking new part templates, because objects with un-modeled part appearance usually had low inference scores. In later QA steps, common part appearance had been asked and modeled, and the computer gradually changed to ask about objects of existing part templates to refine certain AOG branches. In this way, our method did not waste much computation in labeling objects that had been well explained or objects with infrequent appearance.

## 7. Summary and discussion

In this paper, we aim to pursue answers to the following three questions: 1) whether we can represent a pre-trained CNN using an interpretable AOG model, which reveals semantic hierarchy of objects hidden in the CNN, 2) whether the representation of the CNN knowledge can be clear enough to let people directly communicate with middle-level AOG nodes, and 3) whether we can let the computer directly learn from weak supervision of active QA, instead of strongly supervised end-to-end learning.

We tested the proposed method for a total of 37 categories in three benchmark datasets, and our method exhibited superior performance to other baselines in terms of weakly-supervised part localization. *E.g.* our method with 11 part annotations performed better than fast-RCNN with 60 part annotations on the ILSVRC dataset in Fig. 5.

## Acknowledgement

This work is supported by MURI project N00014-16-1-2007 and DARPA SIMPLEX project N66001-15-C-4035.

[5]We used part boxes annotated during the QA process to learn a fast-RCNN for part detection. Given the inference result $\hat{\Lambda}_{V^{\text{tmp}}}$ of part template $V^{\text{tmp}}$ on image $I$, we define a new inference score for localization refinement $S_I^{\text{new}}(V^{\text{tmp}}|\Lambda_{V^{\text{tmp}}}^{\text{new}}) = S_I^{\text{old}}(V^{\text{tmp}}|\hat{\Lambda}_{V^{\text{tmp}}}) + \lambda_1 \Phi(\Lambda_{V^{\text{tmp}}}^{\text{new}}) + \lambda_2 \frac{\|\mathbf{p}(\hat{\Lambda}_{V^{\text{tmp}}}) - \mathbf{p}(\Lambda_{V^{\text{tmp}}}^{\text{new}})\|}{2\sigma^2}$, where $\sigma = 70$ pixels, $\lambda_1 = 5$, and $\lambda_2 = 10$. $\Phi(\Lambda_{V^{\text{tmp}}}^{\text{new}})$ denotes the fast-RCNN's detection score for the patch of $\Lambda_{V^{\text{tmp}}}^{\text{new}}$. $\mathbf{p}(\Lambda)$ denotes the position of $\Lambda$.

## References


[1] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. *In ICCV*, 2015. 2

[2] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. *In ECCV*, 2012. 6, 7, 8

[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. 3

[4] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive training of deformable part models. *In ICCV*, 2011. 3, 6, 7

[5] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. *In ICCV*, 2015. 2

[6] 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. 2, 3, 6, 7, 8

[7] 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. 2

[8] 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. 2

[9] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. *In CHI*, 2014. 3

[10] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016. 7

[11] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *In NIPS*, 2014. 2

[12] L. Duan, D. Xu, I. Tsang, and J. Luo. Visual event recognition in videos by learning from web data. *In CVPR*, 2010. 2

[13] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. 2

[14] R. Girshick. Fast r-cnn. *In ICCV*, 2015. 6, 7, 8

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *In ICLR*, 2014. 6, 8

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012. 1

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 1998. 1

[18] B. Li, W. Hu, T. Wu, and S.-C. Zhu. Modeling occlusion by discriminative and-or structures. *In ICCV*, 2013. 6, 7, 8

[19] D. Lin, X. Shen, C. Lu, and J. Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. *In CVPR*, 2015. 6

[20] 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. 2

[21] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. *In CVPR*, 2015. 2

[22] C. Long and G. Hua. Multi-class multi-annotator active learning with robust gaussian process for visual recognition. *In ICCV*, 2015. 3

[23] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *In CVPR*, 2015. 2

[24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? weakly-supervised learning with convolutional neural networks. *In CVPR*, 2015. 3, 6

[25] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. *In ICCV*, 2015. 3

[26] 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. 2, 5, 6

[27] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. *In CVPR*, 2015. 3

[28] K. J. Shih, A. Mallya, S. Singh, and D. Hoiem. Part localization using multi-proposal consensus for fine-grained categorization. *In BMVC*, 2015. 6

[29] Z. Si and S.-C. Zhu. Learning and-or templates for object recognition and detection. *In PAMI*, 2013. 2

[30] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015. 2, 3

[31] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015. 2

[32] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. *In ACCV*, 2014. 2, 6, 7

[33] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *In arXiv:1312.6034v2*, 2013. 2

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015. 5

[35] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. *In ECCV*, 2012. 3

[36] 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. 2

[37] 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. 2

[38] Q. Sun, A. Laddha, and D. Batra. Active learning for structured probabilistic models with histogram approximation. *In CVPR*, 2015. 3

[39] 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. 3

[40] 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. 6

[41] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *In CVPR*, 2011. 3

[42] 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. 2, 6

[43] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *In arXiv 1609.09408*, 2016. 6, 8

[44] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014. 2

[45] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. *In ECCV*, 2014. 6

[46] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable graphs on convnets via multi-shot learning. *In AAAI*, 2016. 2, 3, 4, 6, 7

[47] Q. Zhang, Y.-N. Wu, and S.-C. Zhu. Mining and-or graphs for graph matching and object discovery. *In ICCV*, 2015. 2

[48] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015. 2

[49] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016. 2

[50] S. Zhu and D. Mumford. A stochastic grammar of images. *In Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006. 2