Interpreting CNNs via Decision Trees

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

Abstract

This paper presents a method to learn a decision tree to quantitatively explain the logic of each prediction of a pre-trained convolutional neural networks (CNNs). Our method boosts the following two aspects of network interpretability. 1) In the CNN, each filter in a high conv-layer must represent a specific object part, instead of describing mixed patterns without clear meanings. 2) People can explain each specific prediction made by the CNN at the semantic level using a decision tree, i.e. *which filters (or object parts) are used for prediction and how much they contribute in the prediction. To conduct such a quantitative explanation of a CNN, our method learns explicit representations of object parts in high conv-layers of the CNN and mines potential decision modes memorized in fully-connected layers. The decision tree organizes these potential decision modes in a coarse-to-fine manner. Experiments have demonstrated the effectiveness of the proposed method.*

1. Introduction

Convolutional neural networks (CNNs) [15, 13, 10] have achieved superior performance in many visual tasks, such as object classification and detection. However, besides the discrimination power, model interpretability is still a significant challenge for neural networks. Many studies have been proposed to visualize, analyze, or semanticize feature representations hidden inside a CNN, in order to obtain insight understanding of network representations.

In this study, we propose a new task, *i.e.* using a decision tree to quantitatively explain the logic of CNN predictions at the semantic level. Note that our decision tree summarizes "generic" decision modes for different images to produce prior explanations of CNN predictions. This is different from the visualization of image pixels that are related to CNN outputs. This task requires us to answer the following three questions.

- **How many types of patterns are memorized by the CNN?** [2] defined six types of patterns for CNNs, *i.e.* objects, parts, scenes, textures, materials, and colors. We can roughly consider the first two types as "part patterns" and summarize the last four types as "texture patterns." In this research, we limit our attention to effects of part patterns in the task of object classification. It is because that compared to texture patterns, part patterns are usually contained in higher conv-layers and contribute to the classification task more directly.
- **For each input image, which object-part patterns are used for prediction?** Given different images, a CNN may activate different sets of object-part patterns for prediction. Let us take the bird classification as an example. The CNN may use head patterns as rationales to classify a standing bird and take wing patterns to distinguish a flying bird. We regard such different

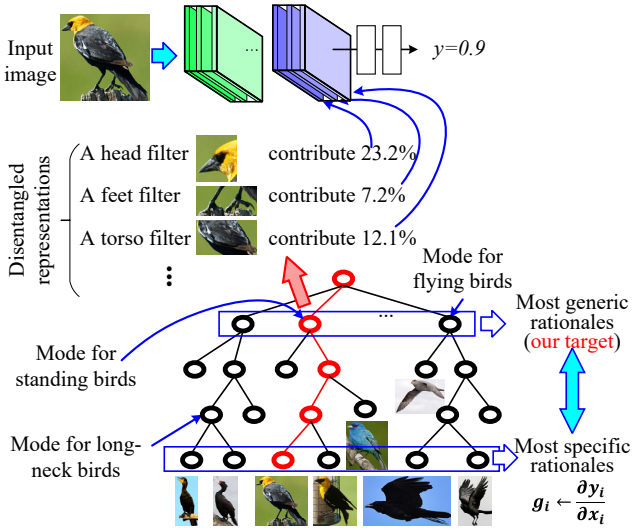

Figure 1. Decision tree that explains a CNN prediction at the semantic level. We learn a CNN for object classification with disentangled representations in the top conv-layer, where each filter represents a specific object part. The decision tree encodes various decision modes hidden inside fully-connected layers of the CNN in a coarse-to-fine manner. Given an input image, we infer a parse tree (red lines) to quantitatively analyze rationales for the CNN prediction, *i.e.* which object parts (or filters) are used for prediction and how much an object part (or filter) contributes to the prediction. We are more interested in high-layer decision modes that summarize low-layer modes and provide compact logic of CNN predictions.

selections of object-part patterns as different decision modes of the CNN. We need to mine such decision modes from a CNN as rationales for each CNN prediction.

- **How to quantitatively measure the contribution of each object-part pattern to a certain prediction?** Our task is to identify the exact contribution of each object-part pattern in the prediction, *e.g.* a head pattern contributes 23%, and a feet pattern contributes 7%.

The above three questions require us 1) to identify the semantic meaning of each neural activation in the feature map of a conv-layer and 2) to quantitatively measure the contribution of different neural activations, which propose significant challenges for state-of-the-art algorithms.

In this paper, we propose to slightly revise the CNN for disentangled representations and learn a decision tree to explain CNN predictions. We are given object images in a certain category and random images as positive and negative samples as inputs to learn both the CNN and the decision tree. We do not label any parts or textures as additional supervision[1]. Firstly, we add the filter loss proposed in [30] to the CNN, which pushes each filter in the top conv-layer towards the representation of an object part. Secondly, we invent a decision tree to quantitatively explain the decision mode for an input image, *i.e.* which object parts (filters) are used for prediction and how much they contribute.

As shown in Fig. 1, each node in the decision tree represents a specific decision mode, and the decision tree organizes all decision modes in a coarse-to-fine manner. Nodes near to the top root node mainly represent common decision modes shared by many samples. Nodes near terminal nodes correspond to fine-grained modes of minority samples. In particular, each terminal mode corresponds to gradients of the CNN output *w.r.t.* different object parts in a certain image.

Compared to terminal fine-grained modes, we are more interested in generic decision modes in high-level nodes. These decision modes usually select significant object-part patterns as rationales for CNN prediction and ignore insignificant ones, which provide compact logic of CNN predictions.

Inference: When the CNN makes a prediction for an input image, the decision tree determines a parse tree (see red lines in Fig. 1) to explain the prediction at different levels.

Contributions: In this paper, we focus on a new task, *i.e.* disentangling CNN representations and learning a decision tree to quantitatively explain the logic of each CNN prediction. We propose a simple yet effective method to learn the decision tree without using any annotations of object parts as additional supervision to learn CNNs. Theoretically, our method is a generic approach to revising CNNs and learning a tight coupling of CNNs and decision trees. Experiments have demonstrated the effectiveness of the proposed method on VGG networks.

[1]Part annotations are not used to learn the CNN and the decision tree. After the learning procedure, we label object parts for top-layer filters to compute part-level contributions in Equation (14).

2. Related work

In this section, we limit our discussion to the literature of opening the black box of CNN representations.

CNN visualization: Visualization of filters in a CNN is the most direct way of exploring the pattern hidden inside a neural unit. Gradient-based visualization [27, 16] estimates the input image that maximizes the activation score of a neural unit. [7] proposed up-convolutional nets to invert feature maps of conv-layers into images. Unlike gradient-based methods, up-convolutional nets cannot mathematically ensure that the visualization result reflects actual neural representations.

[31] proposed a method to accurately compute the image-resolution receptive field of neural activations in a feature map. The estimated receptive field of a neural activation is smaller than the theoretical receptive field based on the filter size. The accurate estimation of the receptive field is crucial to understand a filter's representations.

Network diagnosis: Going beyond visualization, some methods diagnose a pre-trained CNN to obtain insight understanding of CNN representations.

[26] evaluated the transferability of filters in intermediate conv-layers. [1] computed feature distributions of different categories in the CNN feature space. Methods of [8, 19] propagated gradients of feature maps *w.r.t.* the CNN loss back to the image, in order to estimate image regions that directly contribute the network output. [17] proposed a LIME model to extract image regions that are used by a CNN to predict a label.

Network-attack methods [12, 23] diagnosed network representations by computing adversarial samples for a CNN. In particular, influence functions [12] 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. [14] discovered knowledge blind spots (unknown patterns) of a pre-trained CNN in a weakly-supervised manner. [29] developed a method to examine representations of conv-layers and automatically discover potential, biased representations of a CNN due to the dataset bias.

CNN semanticization: Compared to the diagnosis of CNN representations, some studies aim to learn more meaningful CNN representations. Some studies extracted neural units with certain semantics from CNNs for different applications. Given feature maps of conv-layers, [31] extracted scene semantics. Simon *et al.* mined objects from fea-

ture maps of conv-layers [20], and learned object parts [21]. [18] proposed a capsule model, which used a dynamic routing mechanism to parse the entire object into a parsing tree of capsules, and each capsule may encode a specific meaning. [30] proposed to learn CNNs with disentangled intermediate-layer representations. [4, 11] learned interpretable input codes for generative models.

Decision trees for neural networks: [9] proposed to distillate representations of a neural network for image classification into a decision tree, but the decision tree did not explain the network logic in a human-interpretable manner. [25] learned a decision tree via knowledge distillation to represent the output feature space of a RNN. This approach used the tree logic to regularize the RNN for better representations.

In spite of the use of tree structures, there are two main differences between the above two studies and our research. Firstly, we focus on a new task of using a tree to semantically explain the logic of each prediction made by a pre-trained CNN. In contrast, decision trees in above studies are mainly learned for classification and cannot provide semantic-level explanations. Secondly, we summarize decision modes from gradients *w.r.t.* neural activations of object parts as rationales to explain CNN prediction. Compared to above "distillation-based" methods, our "gradient-based" decision tree reflects the CNN logic more directly and strictly.

3. Algorithm

3.1. Preliminaries: learning a CNN with disentangled representations

[30] has learned disentangled CNN representations by adding a loss to each filter in the top conv-layer, which pushes the representation of the filter towards a specific object part[2]. Note that people do not need to label object parts for supervision. The CNN assigns each filter with a certain part automatically during the end-to-end learning.

Let $x_i \in \mathbb{R}^{L\times L\times D}$ denote the feature map of the top conv-layer produced by the CNN on a given image I, where L is referred to as the height/width of the feature map, and D is referred to as the filter number. $x^{(d)} \in \mathbb{R}^{L\times L}$ denotes the feature map of the d-th filter f. As shown in Fig. 2, people design L^2 positive templates $\mathbf{T}^+ = \{T_{1,1}, T_{1,2}, \ldots, T_{L,L}\}$ for positive samples to denote ideal activation shapes when the target object part of the filter f appears at L^2 different location candidates on $x^{(d)}$. A negative template T^- is also used to describe feature maps on negative samples. The loss for the filter f is given as the minus mutual information between all feature maps and all templates.

[2] [30] assumes that positive samples belong to a category and share common parts, while negative samples are random images. A filter mainly encodes parts of positive samples.

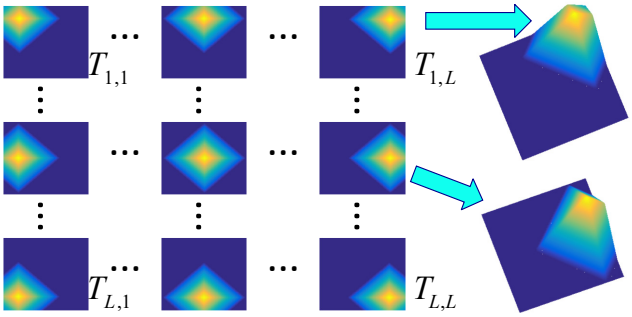

Figure 2. Positive templates for feature maps of a filter f. Each template denotes the ideal activation shape when the object part of the filter f appears at a specific map location.

$$\begin{aligned} Loss_f &= -MI(\mathbf{X};\mathbf{T}) \quad \text{for the } d\text{-th filter } f \\ &= -\sum_{T\in\mathbf{T}} p(T) \sum_{x^{(d)}\in\mathbf{X}} p(x^{(d)}|T) \log \frac{p(x^{(d)}|T)}{p(x^{(d)})} \end{aligned} \quad (1)$$

where $\mathbf{X}$ denotes the set of feature maps of filter f on all training samples, and $\mathbf{T}$ denotes all the L^2+1 templates. The prior probability $p(T)$ is defined as a constant. The fitness between a feature map $x^{(d)}$ and a template T is measured as the conditional likelihood $p(x^{(d)}|T)$.

$$\forall T \in \mathbf{T}, \qquad p(x^{(d)}|T) = \frac{1}{Z_T} \exp\left[tr(x^{(d)} \cdot T)\right] \quad (2)$$

where $Z_T = \sum_{x^{(d)}\in\mathbf{X}} \exp[tr(x^{(d)} \cdot T)]$. $tr(\cdot)$ indicates the trace of a matrix, *i.e.* $tr(x^{(d)} \cdot T) = \sum_{h,w} x^{(h,w,d)} T_{w,h}$, where $x^{(h,w,d)}$ and $T_{h,w}$ denotes the element (h,w) of the matrices. Please see [30] for technical details.

3.2. Learning a decision tree

We focus on the CNN for single-category classification. Let $\mathbf{\Omega} = \mathbf{\Omega}^+ \cup \mathbf{\Omega}^-$ denote indexes of training samples, which consists of positive samples $\mathbf{\Omega}^+$ and negative samples $\mathbf{\Omega}^-$. Given each training image $I_i, i \in \mathbf{\Omega}$, $y_i^* \in \{-1,+1\}$ and y_i denote the ground-truth and the estimated label of the input image, respectively. We train the CNN based on the log logistic loss.

Part concepts in filters: The loss in Equation (1) ensures that each filter represents a specific object part. Let us focus on the d-th channel of the feature map x_i, $x_i^{(d)} \in \mathbb{R}^{L\times L}$, which is produced by a specific filter f. The channel represents a disentangled object part. We can re-write the filter loss in Equation (1) as

$$\begin{aligned} Loss_f = &- H(\mathbf{T}) + H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X}) \\ &+ \textstyle\sum_{x^{(d)}} p(\mathbf{T}^+, x^{(d)}) H(\mathbf{T}^+|X = x^{(d)}) \end{aligned} \quad (3)$$

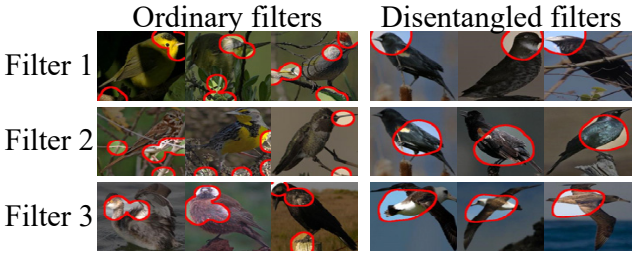

Figure 3. Comparisons between ordinary CNN feature maps and disentangled feature maps that are used in this study. We visualize image regions corresponding to each feature map.

where the first term $H(\mathbf{T}) = -\sum_{T\in\mathbf{T}} p(T)\log p(T)$ is a constant. The second term $H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X})$ encourages each filter can be exclusively triggered by positive samples. The third term $H(\mathbf{T}^+|X = x^{(d)})$ encourages a low entropy of the spatial distribution of neural activations, *i.e.* each filter can only be activated by a single region of an object. It is assumed that a pattern that repetitively appears at different regions of an object usually represents a repetitive texture, instead of an object part. As shown in Fig. 3, this loss ensures the d-th filter f represents a part of the target object.

Internal logic for CNN prediction: As discussed in [21], the decision mode encoded in fully-connected layers for I_i can be roughly described by piecewise-linear representations:

$$y_i = \frac{\partial y_i}{\partial x_i} \otimes x_i + b_i \tag{4}$$

where $\otimes$ denotes the convolution. The gradient *w.r.t.* the feature map $\frac{\partial y_i}{\partial x_i}$ can be computed via gradient back-propagation. To simplify the computation, we use vectors $\mathbf{x}_i, \mathbf{g}_i \in \mathbb{R}^D$ to approximate the 3-order tensors $x_i, \frac{\partial y_i}{\partial x_i} \in \mathbb{R}^{L\times L\times D}$ as $\mathbf{x}_i^{(d)} = \frac{1}{s_d}\sum_{h,w} x_i^{(h,w,d)}$ and $\mathbf{g}_i^{(d)} = \frac{s_d}{L^2}\sum_{h,w}\frac{\partial y_i}{\partial x_i^{(h,w,d)}}$, where $\mathbf{x}_i^{(d)}$ denotes the d-th element of $\mathbf{x}_i$ and $x_i^{(h,w,d)}$ denotes the element at the location (h,w,d) in x_i. $s_d = \mathbf{E}_{i,h,w}\, x_i^{(h,w,d)}$ measures activation magnitudes of the d-th filter. The s_d-based normalization is conducted for more convincing results.

Considering part semantics of filters, activation values in different dimensions of $\mathbf{x}_i$ reflect the signal strength of different object parts, and the gradient $\mathbf{g}_i$ corresponds to the selection of object parts for the CNN prediction. Thus, we use $\mathbf{x}_i$ and $\mathbf{g}_i$ to describe the prediction rationale, *i.e.* using which object parts for prediction.

$$y_i \approx \mathbf{g}_i^T\mathbf{x}_i + b_i \tag{5}$$

Without loss of generality, we further normalize the gradient term $\mathbf{g}_i$ to a unit vector for more reasonable results[3].

[3] $y_i \leftarrow y_i/\|\mathbf{g}_i\|$, $\mathbf{g}_i \leftarrow \mathbf{g}_i/\|\mathbf{g}_i\|$, and $b_i \leftarrow b_i/\|\mathbf{g}_i\|$.

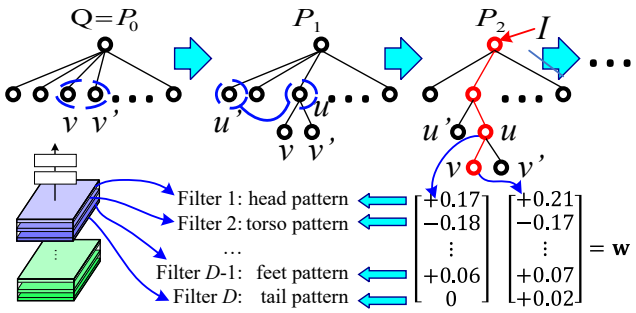

Figure 4. Process of learning a decision tree. Red lines in P_3 indicate a parse tree to explain rationales of a given image I.

Tree: As shown in Fig. 4, we extract decision modes encoded in fully-connected layers of the CNN and build a decision tree to organize the hierarchy of decision modes. From the top node to terminal nodes, the decision tree encodes decision modes in a coarse-to-fine manner. Each node v in the decision tree represents a common decision mode shared by a group of positive[2] training samples $\Omega_v \subset \mathbf{\Omega}^+$. The node v may has a set of children nodes $v' \in Child(v)$ to further divide the decision mode of v into fine-grained modes. Decision modes in terminal nodes is close to gradients $\mathbf{g}_i$ of specific samples. For each node v, we formulate its rationales of its decision mode as

$$h_v(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b, \qquad \mathbf{w} = \boldsymbol{\alpha}\circ\overline{\mathbf{g}} \tag{6}$$

where $\mathbf{w}$ is referred to as rationales of the decision mode. $\overline{\mathbf{g}}$ denotes a unit gradient $\|\overline{\mathbf{g}}\|_2 = 1$. $\boldsymbol{\alpha} \in \{0,1\}^D$ is given as a binary vector to select a few filters to construct the decision mode. $\circ$ denote element-wise multiplications. $\overline{\mathbf{g}}$ reflects common gradients of all samples within v's sample set Ω_v, and we compute parameters $\boldsymbol{\alpha}$ and b via sparse representations.

$$\max_{\overline{\mathbf{g}}} \textstyle\sum_{i\in\Omega_v} cosine(\mathbf{g}_i, \overline{\mathbf{g}}), \quad s.t.\ \ \overline{\mathbf{g}}^T\overline{\mathbf{g}} = 1 \tag{7}$$

$$\min_{\boldsymbol{\alpha},b} \tfrac{1}{\|\Omega_v\|}\textstyle\sum_{i\in\Omega_v}(\mathbf{w}^T\mathbf{x}_i + b - y_i)^2 + \lambda\|\boldsymbol{\alpha}\|_1 \tag{8}$$

Learning: The basic idea of learning a decision tree is to summarize common decision modes from specific decision modes of different samples to represent general rationales for CNN predictions. At the beginning, we initialize the gradient $\mathbf{g}_i$ of each positive[2] sample I_i as a terminal node by setting $\overline{\mathbf{g}} = \mathbf{g}_i$ and $\boldsymbol{\alpha} = \mathbf{1}$. Thus, we build an initial tree Q as shown in Fig. 4, in which the top node takes gradients of all positive[2] samples as children. Then, in each step, we select and merge two nodes $v, v' \in V$ in the second layer (*i.e.* children of the top node) to obtain a new node u, where V denotes the set of nodes in the second layer. v and v' become u's children, and u replaces v and v' as a new child of the top node. In this way, we gradually modify the initial tree $P_0 = Q$ towards the final decision tree after T merging

operations as

$$Q = P_0 \to P_1 \to P_2 \to \cdots \to P_T = \hat{P} \tag{9}$$

We formulate the overall objective of learning as follows.

$$\max_P E, \qquad E = \frac{\prod_{i\in\mathbf{\Omega}^+} P(\mathbf{x}_i)}{\prod_{i\in\mathbf{\Omega}^+} Q(\mathbf{x}_i)} e^{-\beta\|V\|} \tag{10}$$

where $P(\mathbf{x}_i)$ denotes the likelihood of $\mathbf{x}_i$ being positive that is estimated by the tree P. $\prod_i P(\mathbf{x}_i)$ indicates the discriminative power of P. β is a scaling parameter. This objective penalizes the decrease of the discriminative power and encourages the system to use a few decision modes as compact representations for the CNN.

We compute the likelihood of $\mathbf{x}_i$ being positive as

$$P(\mathbf{x}_i) = e^{\gamma\hat{h}(\mathbf{x}_i)} / \sum\nolimits_{j\in\mathbf{\Omega}} e^{\gamma\hat{h}(\mathbf{x}_j)} \tag{11}$$

where $\hat{h}(\mathbf{x}_i)$ denotes the prediction based on $\mathbf{x}_i$, *i.e.* the value $\hat{h}(\mathbf{x}_i) = h_{\hat{v}}(\mathbf{x}_i)$ estimated by the best child $\hat{v} \in V$ in the second layer. γ is a constant scaling parameter.

In the t-th step, we merge two nodes $v, v' \in V$ in the second layer of P_{t-1} to get a new node u and obtain a tree P_t. The increase of E is given as

$$\begin{aligned}\Delta\log E =& \beta + \sum\nolimits_{i\in\mathbf{\Omega}^+}\Big[\gamma\hat{h}_t(\mathbf{x}_i) - \gamma\hat{h}_{t-1}(\mathbf{x}_i) \\ & - \log\sum\nolimits_{j\in\mathbf{\Omega}} e^{\gamma\hat{h}_t(\mathbf{x}_j)} + \log\sum\nolimits_{j\in\mathbf{\Omega}} e^{\gamma\hat{h}_{t-1}(\mathbf{x}_j)}\Big]\end{aligned} \tag{12}$$

Based on the above equation, we learn the decision tree via a greedy strategy. In each step, we select and merge the nodes $v, v' \in V$ that maximize the value of $\frac{\Delta\log E}{\|\Omega_v\|+\|\Omega_{v'}\|}$. We normalize $\Delta\log E$ using the sample number of the selected nodes to get more reasonable clustering performance. Because each node merger operation only affects $\hat{h}(\mathbf{x}_i)$ values of a few examples in $\Omega_v \cup \Omega_{v'}$, we can quickly estimate $\Delta\log E$ for each pair of nodes (v, v').

3.3. Interpreting CNNs

Given a testing image I_i, we use the CNN to make a prediction y_i. We can use the decision tree to compute quantitative explanations for rationales of the prediction. During the inference procedure, we can infer a parse tree, which starts from the top node, in a top-down manner. Red lines in Fig. 4 shows a parse tree. When we select node u to represent the decision mode of I_i, we can further select its child $\hat{v}$ that maximizes the compatibility between I_i's true gradients $\mathbf{g}_i$ and the node mode as a more fine-grained mode:

$$\hat{v} = \text{argmax}_{v\in Child(u)} cosine(\mathbf{g}_i, \mathbf{w}_v) \tag{13}$$

where $\mathbf{w}_v$ denotes the parameter of node v (unlike in Equation (6), we add the subscript v to differentiate the parameter from those of other nodes).

A node v in the decision tree provides an explanation for the prediction of image I_i at a certain fine-grained level. We compute the vector $\boldsymbol{\rho}_i$ and $\boldsymbol{\varrho}_i$ to evaluate the contribution of different filters in the prediction.

$$\boldsymbol{\rho}_i = \mathbf{w}_v \circ \mathbf{x}_i, \qquad \boldsymbol{\varrho}_i = \mathbf{A}\boldsymbol{\rho}_i \tag{14}$$

where the d-th element of $\boldsymbol{\rho}_i$, $\rho_i^{(d)}$, denotes the contribution to the CNN prediction that is made by the d-th filter. If $\rho_i^{(d)} > 0$, then the d-th object part makes a positive contribution. If $\rho_i^{(d)} < 0$, then the d-th filter makes a negative contribution. We use a matrix $\mathbf{A} \in \{0,1\}^{M\times D}$ to assign each filter in the top conv-layer with a specific object part, where M is the part number. Similarly, the m-th element of $\boldsymbol{\varrho}_i$, $\varrho_i^{(m)}$ measures the contribution of the m-th object part.

4. Experiments

4.1. Implementation details

We learned a disentangled CNN based on the structure of the VGG-16 network [22]. We followed the technique of [30] to modify a VGG-16 network to a disentangled CNN, which changed the top conv-layer of the VGG-16 to a disentangled conv-layer and further added a disentangled conv-layer on the top conv-layer. We used feature maps of the new top conv-layer as the input of our decision tree. We loaded parameters of all thirteen old conv-layers directly from a VGG-16 network that was pre-trained using images in the ImageNet ILSVRC 2012 dataset [6] with a loss for 1000-category classification. We initialized parameters of the new top conv-layer and all fully-connected layers. We then fine-tuned the CNN for single-category classification using three benchmark datasets (which will be introduced later). We simply set parameters as $\beta = 1$, $\gamma = 1/\mathbf{E}_{i\in\mathbf{\Omega}^+}[y_i]$, and $\lambda = 10^{-6}\sqrt{\|\Omega_v\|}$ in all experiments to enable fair comparisons.

4.2. Datasets

Because the quantitative explanation of CNN predictions requires us to assign each filter in the top conv-layer with a specific object part, we used three benchmark datasets with ground-truth art annotations to evaluate our method. The selected datasets include the PASCAL-Part Dataset [5], the CUB200-2011 dataset [24], and the ILSVRC 2013 DET Animal-Part dataset [28]. Just like in most part-localization studies [5, 28], we used animal categories, which prevalently contain non-rigid shape deformation, for evaluation. *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 [3, 21, 28], we ignored species labels and regarded all these images as a single bird category. The ILSVRC 2013 DET Animal-Part dataset [28] consists of 30

animal categories among all the 200 categories for object detection in the ILSVRC 2013 DET dataset [6].

4.3. Analysis of object parts for prediction

In this subsection, we analyzed the contribution of different object parts in the CNN prediction, when we assigned each filter with a specific object part. The vector $\boldsymbol{\varrho}_i$ in Equation (14) specifies the contribution of different object parts in the prediction of y_i. For the m-th object part, we computed $contri_m = |\varrho_i^{(m)}| / \sum_{m'=1}^{M} |\varrho_i^{(m')}|$ as the ratio of the contribution of the part.

More specifically, for CNNs based on the ILSVRC 2013 DET Animal-Part dataset, we manually labeled the object part for each filter in the top conv-layer. For CNNs based on the Pascal VOC Part dataset [5], [30] merged tens of small parts into several major landmark parts for the six animal categories. Given a CNN for a certain category, we used [31] to estimate regions in different images that corresponded to each filter's neural activations, namely the *image receptive field* of the filter (please see Figs. 6 and 3). For each filter, we selected a part from all major landmark parts, which was closest to the filter's image receptive field through all positive samples. For the CNN based on the CUB200-2011 dataset, we used ground-truth positions of the breast, forehead, nape, tail of birds as major landmark parts. Similarly, we assigned each filter in the top conv-layer with the nearest landmark part.

4.4. Evaluation metrics

We used three metrics to evaluate the accuracy of the decision tree. The first metric is the *classification accuracy*. Because $\hat{h}(\mathbf{x}_i)$ denotes the prediction of y_i based on the best child in the second layer, we regarded $\hat{h}(\cdot)$ as the output of the tree and we evaluated the discrimination power of $\hat{h}(\cdot)$. We used values of $\hat{h}(\mathbf{x}_i)$ for classification and compared its classification accuracy with the accuracy of the CNN.

Because the disentangled CNN representation selectively learns object-part patterns in positive samples and treats negative samples as random images, we use the other two metrics to identify whether tree nodes correctly reflect CNN representations of positive samples. The second metric, namely the *prediction error*, measures the error of the estimated value $\hat{h}(\mathbf{x}_i)$ *w.r.t* the true value y_i. We computed the prediction error as $\mathbf{E}_{i\in\mathbf{\Omega}^+}[|\hat{h}(\mathbf{x}_i) - y_i|]/(\max_{i\in\mathbf{\Omega}} y_i - \min_{i\in\mathbf{\Omega}} y_i)$, where we normalized the error using the value range of y_i.

The third metric, namely the *fitness of filter contribution*, compares the ground-truth contribution of different filters in the top conv-layer with the estimated contribution of these filters during the prediction process. When the decision tree uses node $\hat{v}$ to explain the prediction for I_i, the contribution vector $\boldsymbol{\rho}_i$ in Equation (14) denotes the contribution distribution of different filters. $\mathbf{t}_i = \mathbf{g}_i \circ \mathbf{x}_i$ corresponds to the ground-truth contribution distribution. We reported the interaction-of-the-union value between $\boldsymbol{\rho}_i$ and $\mathbf{t}_i$ to measure the fitness of the ground-truth and the estimated filter contributions. *I.e.* we computed the fitness as $\mathbf{E}_{i\in\mathbf{\Omega}^+}[\frac{\min(\hat{\rho}_i^{(d)}, |t_i^{(d)}|)}{\max(\hat{\rho}_i^{(d)}, |t_i^{(d)}|)}]$, where $t_i^{(d)}$ denotes the d-th element of $\mathbf{t}_i$ and $\hat{\rho}_i^{(d)} = \max\{\rho_i^{(d)} sign(t_i^{(d)}), 0\}$. We used non-negative values of $\hat{\rho}_i^{(d)}$ and $|t_i^{(d)}|$, because vectors $\boldsymbol{\rho}_i$ and $\mathbf{t}_i$ may have negative elements.

Dataset	2nd	5th	10th	50th	100th
ILSVRC Animal-Part	4.8	31.6	69.1	236.5	402.1
VOC Part	3.8	25.7	59.0	219.5	361.5
CUB200-2011	5.0	32.0	64.0	230.0	430.0

Table 1. Average number of nodes in the 2nd/5th/10th/50th/100th layer of the decision tree.

Dataset	CNN	2nd	5th	10th	50th	100th	bottom
ILSVRC Animal-Part	96.7	94.4	89.0	88.7	88.6	88.7	87.8
VOC Part	95.4	94.2	91.0	90.1	89.8	89.4	88.2
CUB200-2011	96.5	91.5	92.2	88.3	88.6	88.9	85.3

Table 2. Average classification accuracy based on nodes in the 2nd/5th/10th/50th/100th/bottom layer of the decision tree.

Dataset	2nd	5th	10th	50th	100th	bottom
ILSVRC Animal-Part	0.052	0.064	0.063	0.049	0.034	0.00
VOC Part	0.052	0.066	0.070	0.051	0.035	0.00
CUB200-2011	0.075	0.099	0.101	0.087	0.083	0.00

Table 3. Average prediction error based on nodes in the 2nd/5th/10th/50th/100th/bottom layer of the decision tree.

Dataset	2nd	5th	10th	50th	100th	bottom
ILSVRC Animal-Part	0.23	0.30	0.36	0.52	0.65	1.00
VOC Part	0.22	0.30	0.36	0.53	0.67	1.00
CUB200-2011	0.21	0.26	0.28	0.33	0.37	1.00

Table 4. Average fitness of filter contribution based on nodes in the 2nd/5th/10th/50th/100th/bottom layer of the decision tree.

Evaluation for nodes in different layers: The above three metrics evaluate decision modes (nodes) in the second layer of the decision tree. Because nodes in lower layers encode more fine-grained decision modes, we extended such three metrics to evaluate nodes in low layers. When we evaluated nodes in the k-th layer, we temporarily constructed a new tree by removing all nodes above the k-th layer and directly connecting the top node to nodes in the k-th layer. Thus, we can apply the evaluation to the new tree.

4.5. Experimental results and analysis

Table 1 shows the structure of the decision tree by listing numbers of nodes in different layers of the decision tree. Fig. 5 visualizes decision modes in the decision tree. Fig. 6 shows distributions of object-part contributions to the CNN prediction, which were estimated using nodes in the second layer of decision trees. Tables 2, 3, and 4 show the average classification accuracy, the average prediction error,

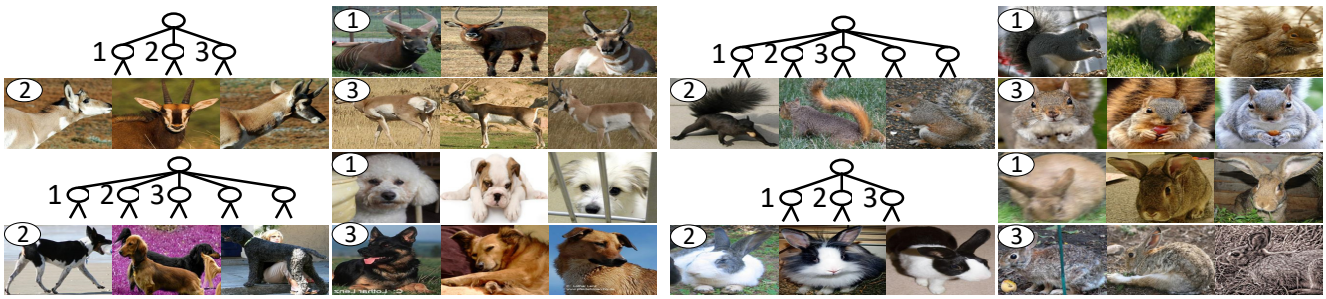

Figure 5. Visualization of decision modes corresponding to nodes in the second layer of the decision tree. We show typical samples of each decision mode for visualization.

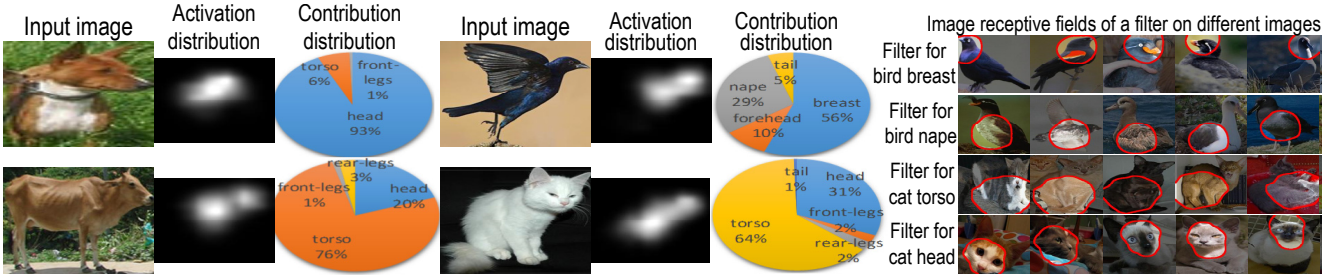

Figure 6. Object-part contributions for CNN prediction. Pie charts show contribution proportions of different parts, which are estimated using the second-layer nodes. Heat maps indicate spatial distributions of neural activations in the top conv-layer (note that the heat maps do not represent distributions of "contributions," because neural activations are not weighted by $\mathbf{g}_i$). Right figures show image receptive fields of different filters. Based on these receptive filters, we assign the filters with different object parts to compute the distribution of object-part contributions.

and the average fitness of filter contribution based on nodes in different layers. Generally speaking, when we used more fine-grained decision modes to explain the prediction logic, the explanation would better fit the actual logic in the CNN, and the prediction error of y_i using decision modes would decrease. However, fine-grained decision modes did not exhibit higher accuracy in classification, because the objective of our method is to summarize decision modes of a pre-trained CNN instead of improving the classification accuracy.

5. Conclusion and discussions

In this study, we focus on a new task of using a decision tree to explain the logic in a CNN at the semantic level. We have developed a method to revise a CNN and built a tight coupling of the CNN and a decision tree. The proposed decision tree encodes decision modes of the CNN as quantitative rationales for each CNN prediction. Our method does not need any annotations of object parts or textures in training samples to guide the learning the CNN. We have tested our method in different benchmark datasets, and experiments have proved the effectiveness of our approach.

Note that theoretically, the decision tree just provides an approximate explanation for CNN predictions, instead of an accurate reconstruction of CNN representation details. There are two reasons. Firstly, without accurate object-part annotations to supervised the learning of CNNs, [30] can only roughly make each filter to represent an object part. The filter may produce incorrect activations in a few challenging samples. Secondly, the decision mode in each node ignores insignificant object-part patterns (filters) to ensure a sparse representation of the decision mode.

References

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

[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*, 2017. 1

[3] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive training of deformable part models. *In ICCV*, 2011. 5

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *In NIPS*, 2016. 3

[5] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *In CVPR*, 2014. 5, 6

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *In CVPR*, 2009. 5, 6

[7] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016. 2

[8] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In arXiv:1704.03296v1*, 2017. 2

[9] N. Frosst and G. Hinton. Distilling a neural network into a soft decision tree. *In arXiv:1711.09784*, 2017. 3

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2016. 1

[11] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: learning basic visual concepts with a constrained variational framework. *In ICLR*, 2017. 3

[12] P. Koh and P. Liang. Understanding black-box predictions via influence functions. *In ICML*, 2017. 2

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

[14] 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. 2

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

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

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. *In KDD*, 2016. 2

[18] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *In NIPS*, 2017. 3

[19] 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 arXiv:1610.02391v3*, 2017. 2

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

[21] M. Simon, E. Rodner, and J. Denzler. Part detector discovery in deep convolutional neural networks. *In ACCV*, 2014. 3, 4, 5

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

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *In arXiv:1312.6199v4*, 2014. 2

[24] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, In California Institute of Technology, 2011. 5

[25] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. *In NIPS TIML Workshop*, 2017. 3

[26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *In NIPS*, 2014. 2

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

[28] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable part graphs on convnets via multi-shot learning. *In AAAI*, 2016. 5

[29] Q. Zhang, W. Wang, and S.-C. Zhu. Examining cnn representations with respect to dataset bias. *In AAAI*, 2018. 2

[30] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. *In arXiv:1710.00935*, 2017. 2, 3, 5, 6, 7

[31] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015. 2, 6