QUANTITATIVELY DISENTANGLING AND UNDERSTANDING PART INFORMATION IN CNNS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents an unsupervised method to learn a neural network, namely an *explainer*, to disentangle and quantify the knowledge of object parts that is used for inference by a pre-trained convolutional neural network (CNN). The explainer performs like an auto-encoder, which quantitatively disentangles part features from intermediate layers and uses the part features to reconstruct CNN features without much loss of information. The disentanglement and quantification of part information help people understand intermediate-layer features used by the CNN. More crucially, we learn the explainer via knowledge distillation without using any annotations of object parts or textures for supervision. In experiments, our method was widely used to diagnose features of different benchmark CNNs, and explainers significantly boosted the feature interpretability.

1 INTRODUCTION

Convolutional neural networks (CNNs) (LeCun et al., 1998; Krizhevsky et al., 2012; He et al., 2016) have shown promise in many visual tasks, but the model interpretability has been an Achilles' heel of deep neural networks for decades. Especially in recent years, helping CNNs to earn trust for safety issues in critical applications has become a new demand for CNNs beyond the accuracy.

Therefore, in order to explain the knowledge contained in a CNN, we aim to quantify signals in intermediate-layer features, which represent object objects, away from signals for other concepts (*e.g.* textures). Note that as shown in Table 1, our objective is **not** to qualitatively visualize object-part features within a CNN. Instead, this work quantitatively explains how much information of object parts and how much information of other concepts are used by the CNN for prediction. The quantitative disentanglement of object-part information and other concepts from an intermediate layer can provide new insights of information processing inside the CNN, and is of considerable theoretical values in explainable AI.

Crucially, we are not given any manual annotations of object parts in this study. Bau et al. (2017) used annotations of object parts[2], textures and other visual concepts to quantify the semantic meaning of each filter. However, the requirement for the rigorous analysis of part concepts conflicts with the fact that there does exist comprehensive annotations of all object parts and textures. It is because object parts encoded by the CNN usually do not have explicit names (*e.g.* the combination of the neck and the left shoulder), thereby difficult to be annotated. In comparison, we propose to objectively quantify the ratio of object-part concepts to all other concepts (see Table 3), which is not supposed not to be affected by the potential subjective bias in human annotations.

Therefore, potential issues in above tasks can be summarized as follows.

• Automatic disentanglement of explainable[1] (object-part) information and unexplainable information that are activated in an intermediate layer is necessary for a rigorous and trustworthy diagnosis of CNNs. Each filter of a conv-layer usually encodes a mixture of various semantics and noises (see Fig. 1).

[2]In (Zhang et al., 2018c), both "objects" and "parts" are categorized as parts.

[1]In this study, we do not distinguish the subtle difference between terms of *interpretable* and *explainable*. We use terms of *interpretable conv-layers* and *interpretable filters* as defined in (Zhang et al., 2018c).

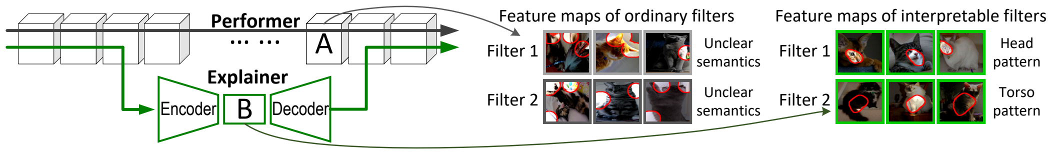


Figure 1: Explainer network. We use an explainer network to disentangle object-part features that used by a pre-trained performer network. The explainer network disentangles object-part features (B) from the performer to mimic signal processing in the performer. The explainer network can also invert the disentangled object-part features to reconstruct features of the performer without much loss of information. We compare traditional features (A) in the performer and the disentangled features (B) in the explainer on the right. The gray and green lines indicate the information-pass route during the inference process and that during the diagnosis process, respectively.

	Traditional CNNs	Interpretable CNNs (Zhang et al., 2018c)	Capsule nets (Sabour et al., 2017)	Visualization & diagnosis	Explainer
Can explain pre-trained model				✓	✓
Disentangle explainable signals		✓	✓		✓
Semantic-level explanation		✓	✓		✓
High feature interpretability		✓	✓	—	✓
High discrimination power	✓			Do not affect the pre-trained model	
Potential broad applicability	✓			✓	✓

Table 1: Comparison between our research and other studies. Note that this table can only summarize mainstreams in different research directions considering the huge research diversity. Please see Section 2 for detailed discussions of related work.

Therefore, the disentanglement of object-part information requires us **1.** to mathematically model and distinguish features corresponding to object parts away from textures and noises, without any part annotations; and **2.** to quantify neural activations for object parts and other concepts. For example, 90% signals for inference may be quantified as object parts and treat the rest 10% may be considered as textures and noises.

• Semantic explanations: Furthermore, the disentangled features can be assigned with different object parts with clear semantic meanings. In comparisons, network visualization and diagnosis (Fong & Vedaldi, 2017; Selvaraju et al., 2017; Ribeiro et al., 2016; Lundberg & Lee, 2017) mainly illustrate the appearance corresponding to a network output/filter at the pixel level, without quantitatively summarizing strict semantics. Our method identifies which object parts are used for inference. Compared to previous pixel-level visualization and diagnosis of a CNN, our quantitative disentanglement of object-part features presents a more trustworthy way to diagnose CNNs.

Tasks, learning networks to diagnose networks: We propose a new strategy to boost feature interpretability. Given a pre-trained CNN, we learn another neural network, namely a *explainer* network, to transform and decompose chaotic intermediate-layer features of the CNN into (1) elementary features corresponding to object parts and (2) unexplainable features. Accordingly, the pre-trained CNN is termed a *performer* network.

As shown in Fig. 1, the performer is trained for superior performance, in which each filter usually represents a chaotic mixture of object parts and textures (Bau et al., 2017). The explainer works like an auto-encoder, which is learned to mimic the logic of the performer. Two specific sets of filters in the explainer decomposes performer features into features of object parts and other concepts, respectively. Then, the decomposed features are also trained to reconstruct features of upper layers of the performer. The explainer is attached onto the performer without affecting the original discrimination power of the performer.

A high reconstruction quality ensures the explainer successfully mimics the signal processing in the performer (please see Appendix C for more discussions). Besides, the disentangled object-part features can be treated as a paraphrase of performer feature representations. The proposed method roughly quantifies the ratio of neural activations for object parts, *i.e.* telling us
• how much features (*e.g.* 90%) in the performer can be explained as object parts;
• information of what parts is encoded in the performer;
• for each specific inference score, which object parts activate filters in the performer, and how much they contribute to the inference.

Diagnosing black-box networks vs. learning interpretable[1] networks: As compared in Table 1, diagnosing pre-trained black-box neural networks has distinctive contributions beyond learning semantically meaningful intermediate-layer features, such as (Sabour et al., 2017; Zhang et al., 2018c). It is because traditional black-box networks have exhibited much broader applicability than inter-

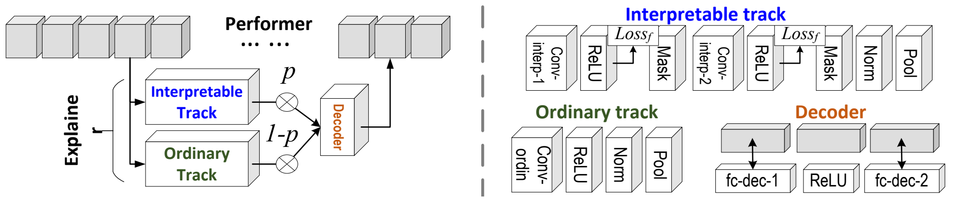


Figure 2: The explainer network (left). Detailed structures within the interpretable[1] track, the ordinary track, and the decoder are shown on the right. People can change the number of conv-layers and FC layers within the encoder and the decoder for their own applications.

pretable networks.
• Model flexibility: Most existing CNNs are black-box models with low interpretability, so diagnosing pre-trained CNNs has broad applicability. In comparison, interpretable neural networks usually have specific requirements for structures (Sabour et al., 2017) or losses (Zhang et al., 2018c), which limit the model flexibility and applicability.
• Interpretability vs. discriminability: Unlike diagnosing pre-trained networks, learning interpretable networks usually suffers from the dilemma between the feature interpretability and its discrimination power. A high interpretability is not equivalent to, and sometimes conflicts with a high discrimination power. As shown in (Sabour et al., 2017), increasing the interpretability of a neural network may affect its discrimination power. Furthermore, filter losses in the interpretable CNN (Zhang et al., 2018c) greatly hurts the classification performance when the network has sophisticated structures (please see Appendix A). People usually have to trade off between the network interpretability and the performance in real applications.

Learning: We learn the explainer by distilling feature representations from the performer to the explainer. No annotations of parts or textures are used to guide the feature disentanglement during the learning process. We add a loss to specific filters in the explainer (see Fig. 2). Without part annotations, the filter loss automatically encourages the filter to be exclusively triggered by a certain object part of a category. This filter is termed an interpretable[1] filter.

Meanwhile, the disentangled object-part features are also required to reconstruct features of upper layers of the performer. Successful feature reconstructions guarantee to avoid significant information loss during the disentanglement of part features.

Contributions of this study are summarized as follows.
• We tackle a new strategy to diagnose pre-trained neural networks, *i.e.* learning an explainer network to disentangle object-part concepts that are used by a pre-trained CNN. Our method roughly quantifies neural activations corresponding to object parts, which sheds new light on understanding black-box models.
• Our method is able to learn the explainer without any annotations of object parts or textures for supervision. Experiments show that our approach has considerably improved the feature interpretability.

2 Related work

Network interpretability: Instead of analyzing network features from a global view (Wolchover, 2017; Schwartz-Ziv & Tishby, 2017; Rauber et al., 2016), (Bau et al., 2017) defined six kinds of semantics for intermediate-layer feature maps of a CNN, *i.e. objects*, *parts*, *scenes*, *textures*, *materials*, and *colors*. Fong and Vedaldi (Fong & Vedaldi, 2018) analyzed how multiple filters jointly represented a certain semantic concept. We can roughly consider the first two semantics as object parts with explicit shapes, and summarize the last four semantics as textures. Our research aims to disentangle object-part information from intermediate layers of the performer network.

Many studies for network interpretability mainly showed visual appearance corresponding to a neural unit inside a CNN (Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015; Simonyan et al., 2013; Dosovitskiy & Brox, 2016; Yosinski et al., 2015; Dong et al., 2017) or extracted image regions that were responsible for network output (Ribeiro et al., 2016; Elenberg et al., 2017; Koh & Liang, 2017; Fong & Vedaldi, 2017; Selvaraju et al., 2017; Kumar et al., 2017). Other studies retrieved mid-level representations with specific meanings from CNNs for various applications (Kolouri et al., 2017;

Lengerich et al., 2017). For example, (Zhou et al., 2015; 2016) selected neural units to describe "scenes". (Simon & Rodner, 2015) discovered objects from feature maps of unlabeled images. In fact, each filter in an intermediate conv-layer usually encodes a mixture of parts and textures, and these studies consider the most notable part/texture component as the semantic meaning of a filter. In contrast, our research uses a filter loss to purify the semantic meaning of each filter (Fig. 1 visualizes the difference between the two types of filters).

A new trend related to network interpretability is to learn networks with disentangled, explainable representations (Hu et al., 2016; Stone et al., 2017; Liao et al., 2016). Many studies learn explainable representations in a weakly-supervised or unsupervised manner. For example, capsule nets (Sabour et al., 2017) and interpretable RCNN (Wu et al., 2017) learned interpretable intermediate-layer features. InfoGAN (Chen et al., 2016) and β-VAE (Higgins et al., 2017) learned meaningful input codes of generative networks. The study of interpretable CNNs (Zhang et al., 2018c) developed a loss to push each intermediate-layer filter towards the representation of a specific object part during the learning process without given part annotations. However, as mentioned in (Bau et al., 2017), an interpretable model cannot always ensure a high discrimination power, which limits the applicability of interpretable models. Therefore, instead of directly boosting the interpretability of the performer network, we propose to learn an explainer network in an unsupervised fashion. (Vaughan et al., 2018) distilled knowledge of a network into an additive model, but this study does not explain the network at a semantic level.

Meta-learning: Our study is also related to meta-learning (Chen et al., 2017; Andrychowicz et al., 2016; Li & Malik, 2016; Wang et al., 2017). Meta-learning uses an additional model to guide the learning of the target model. In contrast, our research uses an additional explainer network to explain intermediate-layer features of the target performer network.

3 ALGORITHM

3.1 NETWORK STRUCTURE OF THE EXPLAINER

As shown in Fig. 2, the explainer network has two modules, *i.e.* an encoder and a decoder, which decompose the performer's intermediate-layer features into explainable object-part features and invert object-part features back to features of the performer, respectively. If features of the performer can be well reconstructed, then we can roughly consider that the decomposed features contain nearly the same information as features in the performer.

We applied the encoder and decoder with following structures to all types of performers in experiments. Nevertheless, people can change the layer number of the explainer in their applications.

Encoder: In order to reduce the risk of over-interpreting textures or noises as parts, we design two tracks for the encoder, namely an *interpretable*[1] *track* with interpretable filters and an *ordinary track* with ordinary filters, which model object-part features and other features, respectively. Although as discussed in (Zhang et al., 2018c), a high conv-layer mainly represents parts rather than textures, avoiding over-interpreting is still necessary for the explainer.

The interpretable track disentangles object parts from chaotic features. This track has two interpretable conv-layers (namely *conv-interp-1,conv-interp-2*), each followed by a ReLU layer and a mask layer. The interpretable conv-layer is defined in (Zhang et al., 2018c). Each filter in an interpretable conv-layer is learned to be exclusively triggered by a specific object part, and is termed an interpretable filter. This interpretable filter is learned using both the task loss and an filter loss. The filter loss boosts the interpretability, which will be introduced later. Besides, the ordinary track contains a conv-layer (namely *conv-ordin*), a ReLU layer, and a pooling layer.

We sum up output features of the interpretable track x_{interp} and those of the ordinary track x_{ordin} as the final output of the encoder, *i.e.* $x_{\text{enc}} = p \cdot x_{\text{interp}} + (1-p) \cdot x_{\text{ordin}}$, where a scalar weight p measures the quantitative contribution from the interpretable track. p is parameterized as a sigmoid probability $p = sigmoid(w_p)$, $w_p \in \boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is the set of parameters to be learned. Our method encourages a large p so that most information in x_{enc} comes from the interpretable track.

For example, if $p = 0.9$, we can roughly consider that about 90% feature information from the performer can be represented as object parts due to the use of norm-layers.

Norm-layer: We normalize x_{interp} and x_{ordin} to make the probability p accurately represent the ratio of the contribution from the interpretable track, *i.e.* making each channel of these feature maps produces the same magnitude of activation values. Thus, we add two norm-layers to the interpretable track and the ordinary track (see Fig. 2). For each input feature map $x \in \mathbb{R}^{L\times L\times D}$, the normalization operation is given as $\hat{x}^{(ijk)} = x^{(ijk)}/\alpha_k$, where $\alpha_k \in \boldsymbol{\alpha} \subset \boldsymbol{\theta}$ denotes the average activation magnitude of the k-th channel $\alpha_k = \mathbb{E}_x[\sum_{ij} \max(x^{(ijk)}, 0)]$ through feature maps of all images, where $x^{(ijk)}$ denotes an element in x. We update $\boldsymbol{\alpha}$ during the forward propagation, just like in the learning for batch normalization.

Mask layer: We add mask layers after two interpretable conv-layers to remove activations that are unrelated to the target part. As shown in Fig. 3, each activated feature map after the mask layer always has a single activation peak.

Let $x_f \in \mathbb{R}^{L\times L}$ denote the feature map of an interpretable filter f after the ReLU operation. The mask layer localizes the potential target object part on x_f as the neural unit with the strongest activation $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_f^{(ij)}$, where $\mu = [i, j]$ denotes a neural unit in x_f $(1 \le i, j \le L)$, and $x_f^{(ij)}$ indicates its activation value.

Based on the estimated part location $\hat{\mu}$, the mask layer assigns x_f with a mask $mask_f$ to remove noises, $x_f^{\text{masked}} = x_f \circ mask_f$, where $\circ$ denotes the Hadamard product. The mask *w.r.t.* $\hat{\mu}$ is given as $mask_f = \max(T_{\hat{\mu}}, 0)$, where $T_{\hat{\mu}}$ is a pre-define template with a single activation peak at $\hat{\mu}$. Theoretically, there are various choices for the template shape, but for simplicity, we used part templates in (Zhang et al., 2018c) in this study, *i.e.* $T_{\hat{\mu}}^{(ij)} = \max\{1 - \frac{4\|\hat{\mu}-[i,j]\|_1}{L}, 0\}$. $\|\cdot\|_1$ denotes the L-1 norm. Given $\hat{\mu}$, we treat $mask_f$ as a constant to enable gradient back-propagation.

Decoder: The decoder inverts x_{enc} to x_{dec} to reconstruct performer features. The decoder has two FC layers, namely *fc-dec-1* and *fc-dec-2*, which are used to reconstruct feature maps of two corresponding FC layers in the performer. The reconstruction loss will be introduced later. The better feature reconstruction indicates that the explainer's feature x_{enc} loses less information.

3.2 LEARNING

When we distill knowledge representations from the performer to the explainer, we minimize the following loss for each input image.

$$Loss(\boldsymbol{\theta}) = \sum_{l\in\mathbf{L}} \lambda_{(l)} \|x_{(l)} - x^*_{(l)}\|^2 - \eta \log p + \sum_f \lambda_f \cdot Loss_f(x_f) \tag{1}$$

where $\boldsymbol{\theta}$ denotes the set of parameters to be learned, including filter weights of conv-layers and FC layers in the explainer, w_p for p, and parameters for norm-layers. $\lambda_{(l)}, \lambda_f$ and η are scalar hyper-parameters.

• **The first term** $\|x_{(l)} - x^*_{(l)}\|^2$ is the reconstruction loss, which also minimizes the information loss when we use the explainer's features to mimic the logic of the performer. $x_{(l)}$ denotes the feature of the FC layer l in the decoder, $\mathbf{L} = \{fc-dec-1, fc-dec-2\}$. $x^*_{(l)}$ indicates the corresponding feature in the performer.

• **The second term** $-\log p$ encourages the interpretable track to make more contribution to the inference. In other words, this term forces object-part information in the performer's features go through the interpretable track, instead of going through the traditional track a short-cut path.

• **The third term** $Loss_f(x_f)$ is the loss of filter interpretability. Without annotations of object parts, the filter loss forces each filter x_f to be exclusively triggered by a specific object part of a certain category.

During the back propagation, the interpretable filter f receives gradients from both the filter loss and the reconstruction loss to update its weights. Whereas, ordinary filters in the ordinary track and the decoder only learn from gradients of the reconstruction loss.

The filter loss was formulated in (Zhang et al., 2019) as the minus mutual information between the distribution of feature maps and that of part locations. Given an input image, x_f is learned to satisfy that if the target part appears, then x_f should have a single activation peak at the part location; otherwise, x_f should keep inactivated.

$$\mathbf{Loss}_f = \sum\nolimits_{x_f\in\mathbf{X}} Loss_f(x_f) = -MI(\mathbf{X};\mathbf{P}) = -\sum\nolimits_{\mu\in\mathbf{P}} p(\mu) \sum\nolimits_{x_f\in\mathbf{X}} p(x_f|\mu) \log \frac{p(x_f|\mu)}{p(x_f)} \tag{2}$$

where $MI(\cdot)$ indicates the mutual information. $\mathbf{X}$ denotes a set of feature maps of the filter f, which are extracted from different input images. $\mathbf{P} = \{\mu | \mu = [i,j], 1 \leq i,j \leq L\} \cup \{\emptyset\}$ is referred to as a set of all part-location candidates. As mentioned above, each location $\mu = [i,j]$ is referred to as an activation unit in the feature map. Besides, $\emptyset \in \mathbf{P}$ denotes the case that the target part of the filter does not appear in the input image, and we expect all units in x_f to keep inactivated. The joint probability $p(x_f, \mu)$ describes the compatibility between x_f and μ. Please see (Zhang et al., 2018c) or Appendix H for details of $p(\mu)$ and $p(x_f|\mu)$. The filter loss ensures x_f match only one of all $L^2 + 1$ location candidates.

As shown in Fig. 2, we add a filter loss to each interpretable filter f in the two conv-layers (*conv-interp-1* and *conv-interp-2*). $x_f \in \mathbb{R}^{L\times L}$ denotes the feature map of the interpretable filter after the ReLU operation.

3.3 Analysis of the algorithm

Our algorithm makes a trade-off between the explanation power and the reconstruction accuracy of the explainer. The item of the reconstruction loss in Equation equation 1 ensures a high reconstruction quality, *i.e.* the explainer should successfully mimic the signal processing in the performer. On the other hand, w_p for p is updated based on both the $-\log p$ loss and the reconstruction loss. The $-\log p$ loss encourages a high value of p, in order to push as much object-part information as possible to the interpretable track. Whereas, the reconstruction loss usually requires a moderate value of p to avoid over-interpreting textural features and noises as object parts.

Equation equation 1 also boosts the mutual information between the feature map and all part locations. In other words, each filter should be triggered by a single region (part) of the object, rather than repetitively appear on different regions of an object. We assume that repetitive shapes on various regions are more likely to describe low-level textures (*e.g.* colors and edges) than high-level parts. We consider the left and right eyes as two different parts, because they have different contexts. Thus, the filter loss pushes each interpretable filter towards the representation of an object part.

4 Experiments

In experiments, we trained explainers for six types of performer networks to demonstrate the broad applicability of our method. Performer networks were pre-trained using object images in two different benchmark datasets for object classification. We visualized feature maps of interpretable filters in the explainer to illustrate semantic meanings of these filters. Experiments showed that interpretable filters in the explainer generated more semantically meaningful feature maps than conv-layers in the performer.

Benchmark datasets: Because the evaluation of filter interpretability required ground-truth annotations of object landmarks[3] (parts), we used two benchmark datasets with part annotations for training and testing, *i.e.* the CUB200-2011 dataset (Wah et al., 2011) and the Pascal-Part dataset (Chen et al., 2014). Note that previous studies (Chen et al., 2014; Zhang et al., 2018c) usually selected animal categories to test part localization, because animals usually contain non-rigid parts, which present great challenges for part localization. Therefore, we followed the experimental design in (Zhang et al., 2018c) that selected the seven animal categories in the two datasets for evaluation. Both the datasets provide object bounding boxes. The CUB200-2011 dataset (Wah et al., 2011) contains 11.8K bird images of 200 species with center positions of fifteen bird landmarks. Here, we considered all 200 bird species in the CUB200-2011 dataset as a single category. The Pascal-Part dataset (Chen et al., 2014) provides ground-truth segmentations of a total of 107 object parts for six animal categories.

Six types of CNNs as performers: We applied our method to six types of performers, including the ResNet-101, ResNet-152 (He et al., 2016), AlexNet (Krizhevsky et al., 2012), the VGG-M (Si-

[3]To avoid ambiguity, a landmark is referred to as the central position of a semantic part with an explicit name (*e.g.* a head, a tail). In contrast, the part corresponding to an interpretable filter does not have an explicit name. We followed experiment settings in (Zhang et al., 2018c), which selected the *head*, *neck*, and *torso* of each category in the Pascal-Part dataset (Chen et al., 2014) as the landmarks and used the *head*, *back*, *tail* of birds in the CUB200-2011 dataset (Wah et al., 2011) as landmarks. It was because these landmarks appeared on testing images most frequently.

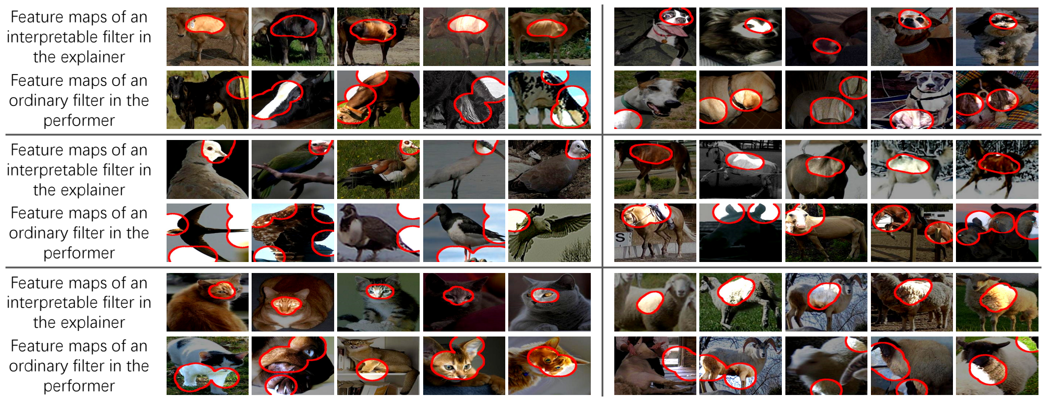

Feature maps of an interpretable filter in the explainer
Feature maps of an ordinary filter in the performer
Feature maps of an interpretable filter in the explainer
Feature maps of an ordinary filter in the performer
Feature maps of an interpretable filter in the explainer
Feature maps of an ordinary filter in the performer

Figure 3: Visualization of interpretable filters in the explainer and ordinary filters in the performer. As discussed in (Bau et al., 2017), the top conv-layer of a CNN is more likely to represent object parts than low conv-layers. We compared filters in the top conv-layer of the performer and interpretable filters in the *conv-interp-2* layer of the explainer. We used (Zhou et al., 2015) to visualize the RF[4] of neural activations in a feature map after a ReLU layer and a mask layer. Ordinary filters are usually activated by repetitive textures, while interpretable filters always represent the same part through different images, which are more semantically meaningful. Please see Appendix B for more results.

monyan & Zisserman, 2015), the VGG-S (Simonyan & Zisserman, 2015), the VGG-16 (Simonyan & Zisserman, 2015).

Two experiments: We followed experimental settings in (Zhang et al., 2018c) to conduct two experiments, *i.e.* an experiment of single-category classification and an experiment of multi-category classification. For single-category classification, we trained six performers with structures of the AlexNet (Krizhevsky et al., 2012), VGG-M (Simonyan & Zisserman, 2015), VGG-S (Simonyan & Zisserman, 2015), VGG-16 (Simonyan & Zisserman, 2015), ResNet-101 (He et al., 2016), and ResNet-152 (He et al., 2016) for the seven animal categories in the two benchmark datasets. Thus, we trained 40 performers, each of which was learned to classify objects of a certain category from other objects. We cropped objects of the target category based on their bounding boxes as positive samples. Images of other categories were regarded as negative samples. For multi-category classification, we trained the VGG-M (Simonyan & Zisserman, 2015), VGG-S (Simonyan & Zisserman, 2015), and VGG-16 (Simonyan & Zisserman, 2015) to classify the six animal categories in the Pascal-Part dataset (Chen et al., 2014).

Experimental details: As discussed in (Bau et al., 2017), high conv-layers in a CNN (performer) are more likely to represent object parts, while low conv-layers mainly encode textures. Therefore, we used feature maps before the top conv-layer of the performer as the input of the explainer, *i.e.* the *relu4* layer of the AlexNet/VGG-M/VGG-S (the 12th/12th/11th layer of the AlexNet/VGG-M/VGG-S) and the *relu5-2* layer of the VGG-16 (the 28th layer). Note that we did not select feature maps of the top conv-layer to enable a fair comparison, because we can parallel the top conv-layer of the performer to the *conv-ordin* layer of the explainer. For ResNets, we used the output feature of the last residual block as the input of the explainer. We trained the explainer network to disentangle these feature maps for testing.

For ordinary CNNs, the output of the explainer reconstructed the feature of the *fc7* layer of the AlexNet/VGG-M/VGG-S/VGG-16 performer and was fed back to the performer. Thus, a reconstruction loss matched features between the *fc-dec-2* layer of the explainer and the *fc7* layer of the performer. Another reconstruction loss connected the *fc-dec-1* layer of the explainer and the previous *fc6* layer of the performer. For ResNets, feature maps of *fc-dec-1* and *fc-dec-2* correspond to outputs of the second last residual block and the last residual block, respectively, although they are not FC layers. Each conv-layer in the explainer had D filters with a $3\times3\times D$ kernel and a biased term, where D is the channel number of its input feature map. We used zero padding to ensure the output feature map had the same size of the input feature map. The *fc-dec-1* and *fc-dec-2* layers in the explainer copied filter weights from the *fc6* and *fc7* layers of the performer, respectively. Pooling layers in the explainer were also parameterized according to the last pooling layer in the performer.

	Single-category							Multi
	bird	cat	cow	dog	horse	sheep	Avg.	Avg.
AlexNet	0.153	0.131	0.141	0.128	0.145	0.140	0.140	–
Explainer	**0.104**	**0.089**	**0.101**	**0.083**	**0.098**	**0.103**	**0.096**	–
VGG-M	0.152	0.132	0.143	0.130	0.145	0.141	0.141	0.135
Explainer	**0.106**	**0.088**	**0.101**	**0.088**	**0.097**	**0.101**	**0.097**	**0.097**
VGG-S	0.152	0.131	0.141	0.128	0.144	0.141	0.139	0.138
Explainer	**0.110**	**0.085**	**0.098**	**0.085**	**0.091**	**0.096**	**0.094**	**0.107**
VGG-16	0.145	0.133	0.146	0.127	0.143	0.143	0.139	0.128
Explainer	**0.095**	**0.089**	**0.097**	**0.085**	**0.087**	**0.089**	**0.090**	**0.109**
ResNet-101	0.147	0.134	0.142	0.127	0.144	0.142	0.139	–
Explainer	**0.098**	**0.088**	**0.099**	**0.088**	**0.088**	**0.093**	**0.092**	–
ResNet-152	0.148	0.134	0.143	0.128	0.145	0.142	0.140	–
Explainer	**0.097**	**0.088**	**0.098**	**0.089**	**0.088**	**0.092**	**0.092**	–

Table 2: Location instability of feature maps between performers and explainers that were trained using the Pascal-Part dataset (Chen et al., 2014). A low location instability indicates a high filter interpretability. Please see Appendix F for comparisons with more baselines.

	Pascal-Part dataset		CUB200
	Single-class	Multi-class	dataset
AlexNet	–	0.7137	0.5810
VGG-M	0.9012	0.8066	0.8611
VGG-S	0.9270	0.8996	0.9533
VGG-16	0.8593	0.8718	0.9579
ResNet-101	0.9727	–	–
ResNet-152	0.9833	–	–

Table 3: Average p values of explainers. p measures the quantitative contribution from the interpretable track. When we used an explainer to diagnose feature maps of a VGG network, about 86%–96% activation scores came from explainable features.

We set $\eta = 1.0 \times 10^6$ for the AlexNet, VGG-M, and VGG-S, because these three CNNs have similar numbers of layers. We set $\eta = 1.5 \times 10^5$ for the VGG-16 and ResNets, since they have more conv-layers. For each type of CNN performers, the parameter setting was uniformly applied to the learning of different performers for various categories. We set $\lambda_{(l)} = 5 \times 10^4 / \mathbb{E}_{x^*_{(l)}}[\| \max(x^*_{(l)}, 0)\|]$, where the expectation was averaged over features of all images.

Evaluation metric: We compared the object-part interpretability between feature maps of the explainer and those of the performer. To obtain a convincing evaluation, we both visualized filters (see Fig. 3) and used the objective metric of location instability to measure the fitness between a filter f and the representation of an object part.

The metric of location instability was widely used (Zhang et al., 2018c). As discussed in (Zhang et al., 2018c), compared to the interpretability metric in (Bau et al., 2017), the location instability is a more reasonable metric when the filter is automatically learned without ground-truth annotations or scales of parts (see Appendix G for details). Given a feature map x_f, we localized the part at the unit $\hat{\mu}$ with the highest activation. We used (Zhou et al., 2015) to project the part coordinate $\hat{\mu}$ on the feature map onto the image plane and obtained $\mathbf{p}_{\hat{\mu}}$. We assumed that if the filter f consistently represented the same object part of a certain category through different images, then distances between the inferred part location $\mathbf{p}_{\hat{\mu}}$ and some object landmarks[3] of the category should not change a lot among different objects. For example, if f always represented the head part on different objects, then the distance between the localized part $\mathbf{p}_{\hat{\mu}}$ (*i.e.* the dog head) and the ground-truth landmark of the shoulder should keep stable, although the head location $\mathbf{p}_{\hat{\mu}}$ changes in different images. Thus, for single-category classification, we computed the deviation of the distance between $\mathbf{p}_{\hat{\mu}}$ and a specific landmark through objects of the category, and we used the average deviation *w.r.t.* various landmarks to evaluate the location instability of f. The location instability was reported as the average deviation, when we computed deviations using all pairs of filters and landmarks of the category. For multi-category classification, we first determined the target category of each filter f and then computed the location instability based on objects of the target category. We assigned each interpretable filter in the explainer to the category whose images can activate the filter most. Please see (Zhang et al., 2018c) for computational details of this evaluation metric.

According to network structures used in experiments, we can parallel the explainer to the top conv-layer of the performer, because they both receive features from the *relu4*/*relu5-2* layer of the performer and output features to the upper layers of the performer. Crucially, as discussed in (Bau et al., 2017), low conv-layers in a CNN usually represent colors and textures, while high conv-layers mainly represent object parts; the top conv-layer of the CNN is most likely to model object parts among all conv-layers. Therefore, to enable a fair comparison, we compared feature maps of the *conv-interp-2* layer of the explainer with feature maps of the top conv-layer of the performer.

4.1 Experimental results and analysis

Tables 2 and 4 compare the interpretability between feature maps in the performer and feature maps in the explainer. Feature maps in our explainers were much more explainable than feature maps in

	AlexNet	VGG-M	VGG-S	VGG-16
Performer	0.1502	0.1476	0.1481	0.1373
Explainer	**0.0906**	**0.0815**	**0.0704**	**0.0490**

Table 4: Location instability of feature maps in performers and explainers that were trained using the CUB200-2011 dataset (Wah et al., 2011). A low location instability indicates a high filter interpretability. Please see Appendix F for comparisons with more baselines.

	Performer	Explainer	Δ Error
VGG-M	6.12%	6.62%	0.5%
VGG-S	5.95%	6.97%	1.02%
VGG-16	2.03%	2.17%	0.14%
ResNet-101	1.67%	3.19%	1.52%
ResNet-152	0.71%	1.55%	0.84%

Table 5: Multi-category classification errors using features of performers and explainers based on the Pascal-Part dataset (Chen et al., 2014). Please see Appendix J for more results of performers and explainers.

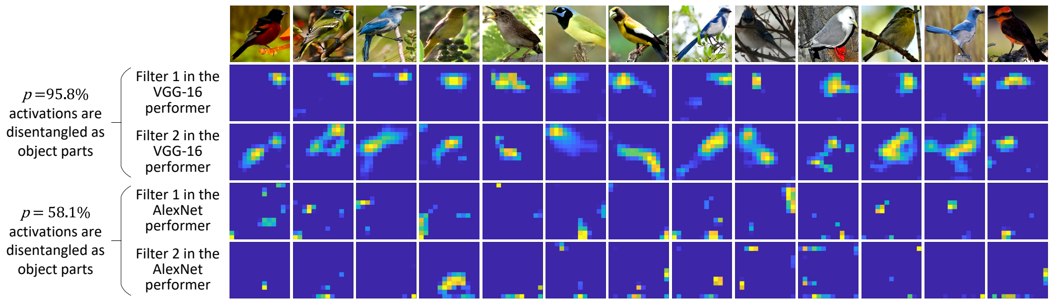

p=95.8% activations are disentangled as object parts
Filter 1 in the VGG-16 performer
Filter 2 in the VGG-16 performer
p = 58.1% activations are disentangled as object parts
Filter 1 in the AlexNet performer
Filter 2 in the AlexNet performer

Figure 4: Comparisons of feature maps of different performers corresponding to different values of p. 95.8% of VGG-16 features were disentangled as object parts, while only 58.1% of AlexNet features were disentangled as object parts. Note that each visualized filter of the performer does not strictly represent a single object part, which is different from interpretable filters in the explainer.

performers in all comparisons. The explainer exhibited only 35.7%–68.8% of the location instability of the performer, which means that interpretable filters in the explainer more consistently described the same object part through different images than filters in the performer.

The p value of an explainer indicates the quantitative ratio of the contribution from explainable features. Table 3 lists p values of explainers that were learned for different performers. p measures the quantitative contribution from the interpretable track. For example, the VGG-16 network learned using the CUB200-2011 dataset has a p value $p = 0.9579$, which means that about 95.8% feature information of the performer can be represented as object parts, and only about 4.2% feature information comes from textures and noises. In contrast, the AlexNet is not so powerful in learning object-part features. Only about 58.1% feature information describes object parts, when the AlexNet is learned the CUB200-2011 dataset.

Fig. 4 visualizes feature maps of the VGG-16 and AlexNet to demonstrate the difference between different conv-layers in part information. We found that the explainer disentangled more features from the VGG-16 network as object parts than those from the AlexNet. Accordingly, the visualized feature maps of the VGG-16 network were also more localized and more related to part patterns. Note that p is just a rough measurement of object-part information. *Accurately* disentangling semantic information from a CNN is still a significant challenge.

To evaluate feature reconstructions of an explainer, we fed the reconstructed features back to the performer for classification. As shown in Table 5, we compared the classification accuracy of explainer's reconstructed features with the accuracy based on original performer features. Performers outperformed explainers in object classification. We used the explainer's increase of classification errors *w.r.t.* the performer (*i.e.* "Δ Error" in Table 5) to measure the information loss during feature transformation in the explainer.

Visualization of filters: We used the visualization method proposed by (Zhou et al., 2015) to compute the receptive field (RF) of neural activations of an interpretable filter (after ReLU and mask operations), which was scaled up to the image resolution. As mentioned in (Zhou et al., 2015), the computed RF represented image regions that were responsible for neural activations of a filter,

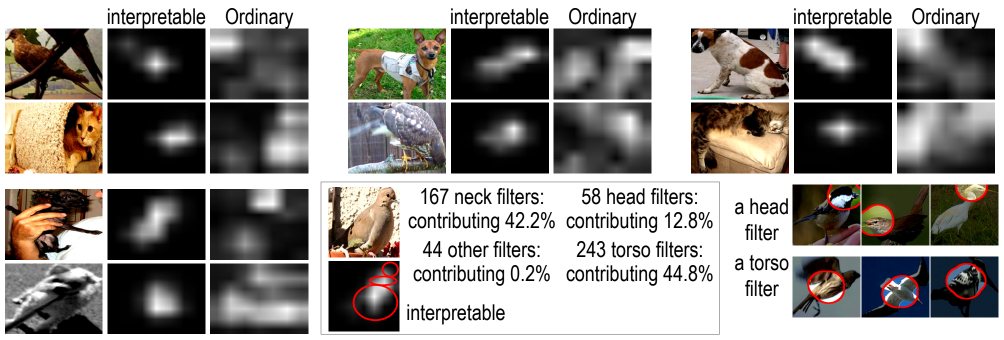

interpretable
Ordinary
interpretable
Ordinary
interpretable
Ordinary
167 neck filters: contributing 42.2%
58 head filters: contributing 12.8%
44 other filters: contributing 0.2%
243 torso filters: contributing 44.8%
interpretable
a head filter
a torso filter

Figure 5: Grad-CAM attention maps and quantitative analysis. We used (Selvaraju et al., 2017) to compute grad-CAM attention maps of interpretable feature maps in the explainer and ordinary feature maps in the performer. Interpretable filters focused on a few distinct object parts, while ordinary filters separated its attention to both textures and parts. We can assign each interpretable filter with a semantic part. *E.g.* the network learned 58, 167, and 243 filters in the *conv-interp-2* layer to represent the head, neck, and torso of the bird, respectively. We used the linear model in (Zhang et al., 2018b) to estimate contributions of different filters to the classification score. We summed up contributions of a part's filters as the part's quantitative contribution. Please see Appendix D for more results.

which was much smaller than the theoretical size of the RF. Fig. 3 used RFs[4] to visualize interpretable filters in the *conv-interp-2* layer of the explainer and ordinary filters in the top conv-layer of the performer. Fig. 5 compares grad-CAM attention maps (Selvaraju et al., 2017) of the *conv-interp-2* layer in the explainer and those of the top conv-layer of the performer. Interpretable filters in an explainer mainly represented an object part, while feature maps of ordinary filters were usually activated by different image regions without clear meanings.

5 Conclusion and discussions

In this paper, we have proposed a new network-diagnosis strategy, *i.e.* learning an explainer network to disentangle object-part features and other features that are used by a pre-trained performer network. We have developed a simple yet effective method to learn the explainer, which guarantees the high interpretability of feature maps without using annotations of object parts or textures for supervision. Theoretically, our explainer-performer structure supports knowledge distillation into new explainer networks with different losses. People can revise network structures inside the ordinary track, the interpretable track, and the decoder and apply novel interpretability losses to the interpretable track.

We divide the encoder of the explainer into an interpretable track and an ordinary track to reduce the risk of over-interpreting textures or noises as parts. Fortunately, experiments have shown that most of signals in the performer can be explained as parts.

Directly learning interpretable features (Zhang et al., 2018c) may hurt the discrimination power, especially when the CNN has complex structures (*e.g.* residual networks, see Section 1). In comparison, learning an explainer does not affect the performer, thereby ensuring the broad applicability.

We have applied our method to different types of performers, and experimental results show that our explainers can disentangle most information in the performer into object-part feature maps, which significantly boosts the feature interpretability. *E.g.* for explainers for VGG networks, more than 80% signals go through the interpretable track, so they can be explained as object-part information.

[4]When an ordinary filter in the performer does not have consistent contours, it is difficult for (Zhou et al., 2015) to align different images to compute the average RF. Thus, for performers, we simply used a round RF for each activation. We overlapped all activated RFs in a feature map to compute the final RF.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *In NIPS*, 2016.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *In CVPR*, 2017.

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.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *In NIPS*, 2016.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. *In ICML*, 2017.

Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *In arXiv:1708.05493*, 2017.

Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. *In CVPR*, 2016.

Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. *In NIPS*, 2017.

Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *In CVPR*, 2018.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *In ICCV*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *In CVPR*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-vae: learning basic visual concepts with a constrained variational framework. *In ICLR*, 2017.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P. Xing. Harnessing deep neural networks with logic rules. *In arXiv:1603.06318v2*, 2016.

PangWei Koh and Percy Liang. Understanding black-box predictions via influence functions. *In ICML*, 2017.

Soheil Kolouri, Charles E. Martin, and Heiko Hoffmann. Explaining distributed neural activations via unsupervised learning. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017.

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

Devinder Kumar, Alexander Wong, and Graham W. Taylor. Explaining the unexplained: A class-enhanced attentive response (clear) approach to understanding deep neural networks. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017.

Yann LeCun, Lèon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *In Proceedings of the IEEE*, 1998.

Benjamin J. Lengerich, Sandeep Konam, Eric P. Xing, Stephanie Rosenthal, and Manuela Veloso. Visual explanations for convolutional neural networks via input resampling. *In ICML Workshop on Visualization for Deep Learning*, 2017.

Ke Li and Jitendra Malik. Learning to optimize. *In arXiv:1606.01885*, 2016.

Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. *In NIPS*, 2016.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *In NIPS*, 2017.

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

Paulo E. Rauber, Samuel G. Fadel, Alexandre X. Falc ao, and Alexandru C. Telea. Visualizing the hidden activity of artificial neural networks. *In Transactions on PAMI*, 23(1):101–110, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. *In KDD*, 2016.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. *In NIPS*, 2017.

Ravid Schwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *In arXiv:1703.00810*, 2017.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In ICCV*, 2017.

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

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

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. *In arXiv:1312.6034*, 2013.

Austin Stone, Huayan Wang, Yi Liu, D. Scott Phoenix, and Dileep George. Teaching compositionality to cnns. *In CVPR*, 2017.

Joel Vaughan, Agus Sudjianto, Erind Brahimi, Jie Chen, and Vijayan N. Nair. Explainable neural networks based on additive index models. *In arXiv:1806.01933*, 2018.

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.

J.X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J.Z.d Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *In arXiv:1611.05763v3*, 2017.

Natalie Wolchover. New theory cracks open the black box of deep learning. *In Quanta Magazine*, 2017.

Tianfu Wu, Xilai Li, Xi Song, Wei Sun, Liang Dong, and Bo Li. Interpretable r-cnn. *In arXiv:1711.05226*, 2017.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *In ICML Deep Learning Workshop*, 2015.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014.

Q. Zhang, R. Cao, F. Shi, Y.N. Wu, and S.-C. Zhu. Interpreting cnn knowledge via an explanatory graph. *In AAAI*, 2018a.

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

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *In CVPR*, 2018c.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable cnns. *In arXiv:1901.02413*, 2019.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016.

A Conflicts between the feature interpretability and the discrimination power

Unlike our learning explainers to diagnose pre-trained CNNs, directly learning explainable intermediate-layer features usually hurts the discrimination power of the neural network, especially when the neural network has sophisticated structures.

To prove this assertion, we did further experiments. We followed experimental settings in (Zhang et al., 2018c) to add an interpretable conv-layer with filter losses above the last residual block and learn interpretable feature representations. We trained ResNets for the binary classification for each animal category in the VOC Part dataset (Chen et al., 2014), just like experiments in (Zhang et al., 2018c).

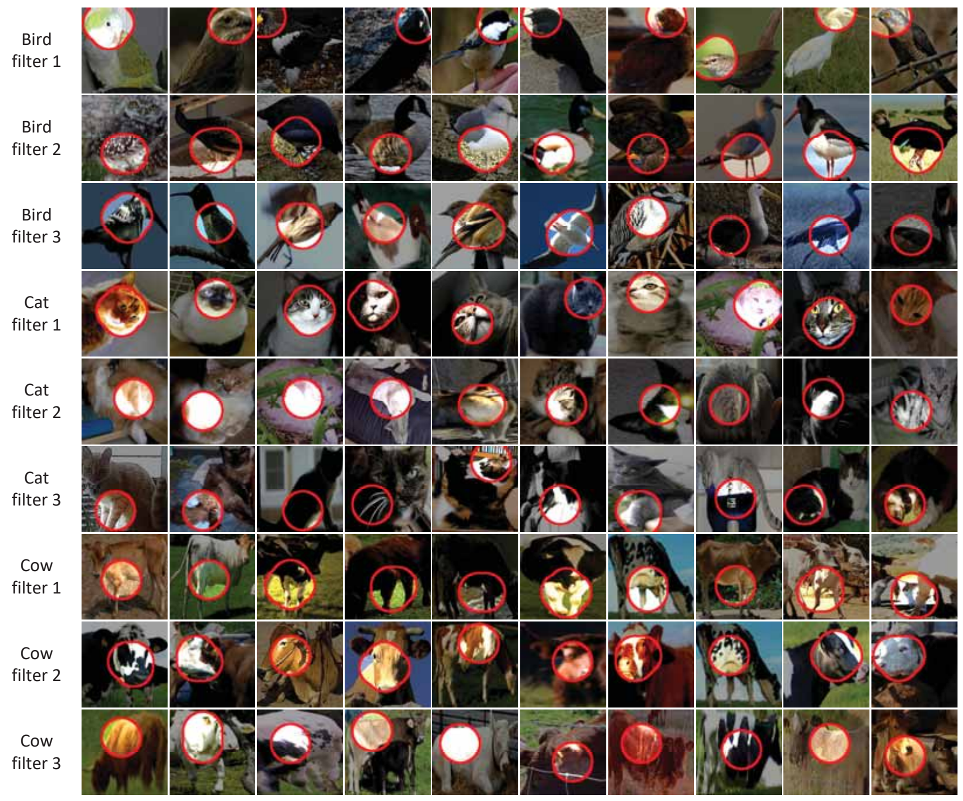
In the following table, we compared the classification errors between the original ResNet-101/ResNet-152 and the revised ResNet-101/ResNet-152 with an interpretable layer.

	bird	cat	cow	dog	horse	sheep	Avg.
original ResNet-101	**0.50%**	**2.00%**	**0.51%**	**1.50%**	**5.00%**	**0.51%**	**1.67%**
interpretable ResNet-101	0.50%	2.75%	0.51%	3.00%	12.50%	8.10%	4.56%
original ResNet-152	**0.25%**	**0.75%**	**0.00%**	**1.75%**	**1.26%**	**0.25%**	**0.71%**
interpretable ResNet-152	2.24%	2.25%	8.38%	2.50%	9.09%	6.06%	5.09%

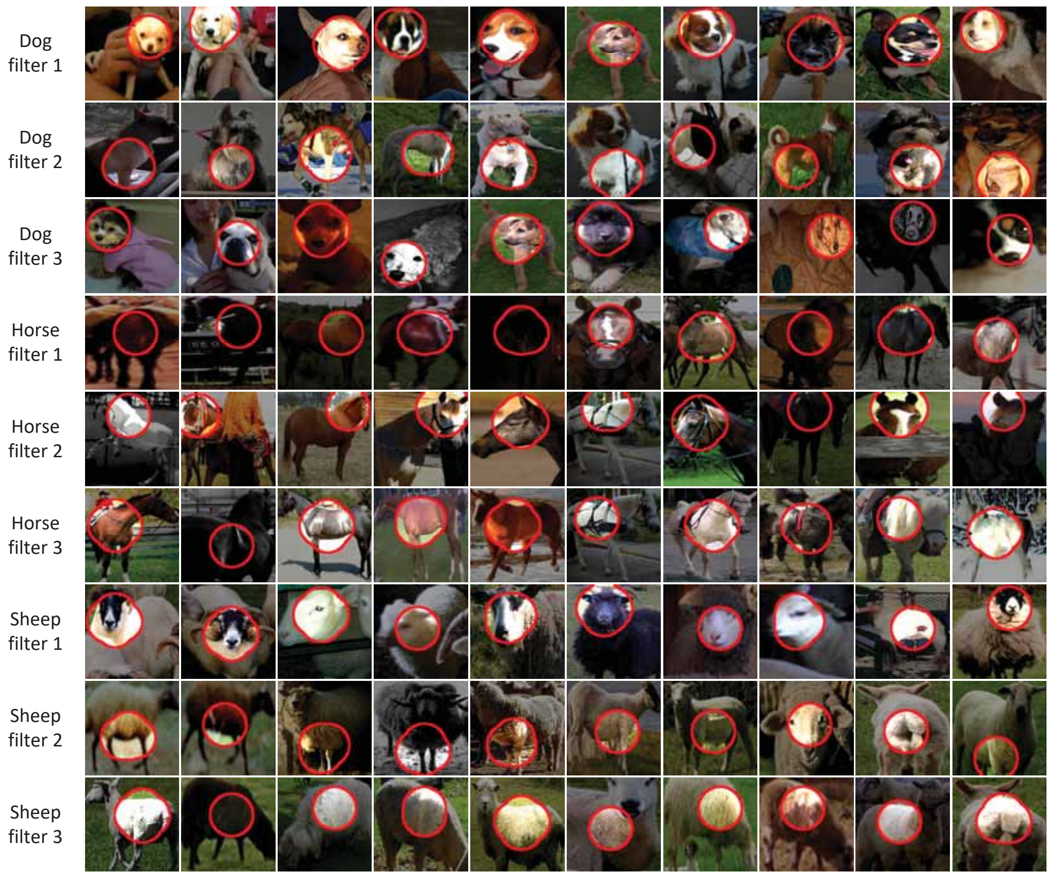
Table 6: Classification errors of original ResNets and interpretable ResNets

Therefore, our learning explainers to diagnose a pre-trained CNN without hurting the discrimination power of the CNN has distinctive contributions beyond learning models with interpretable features.

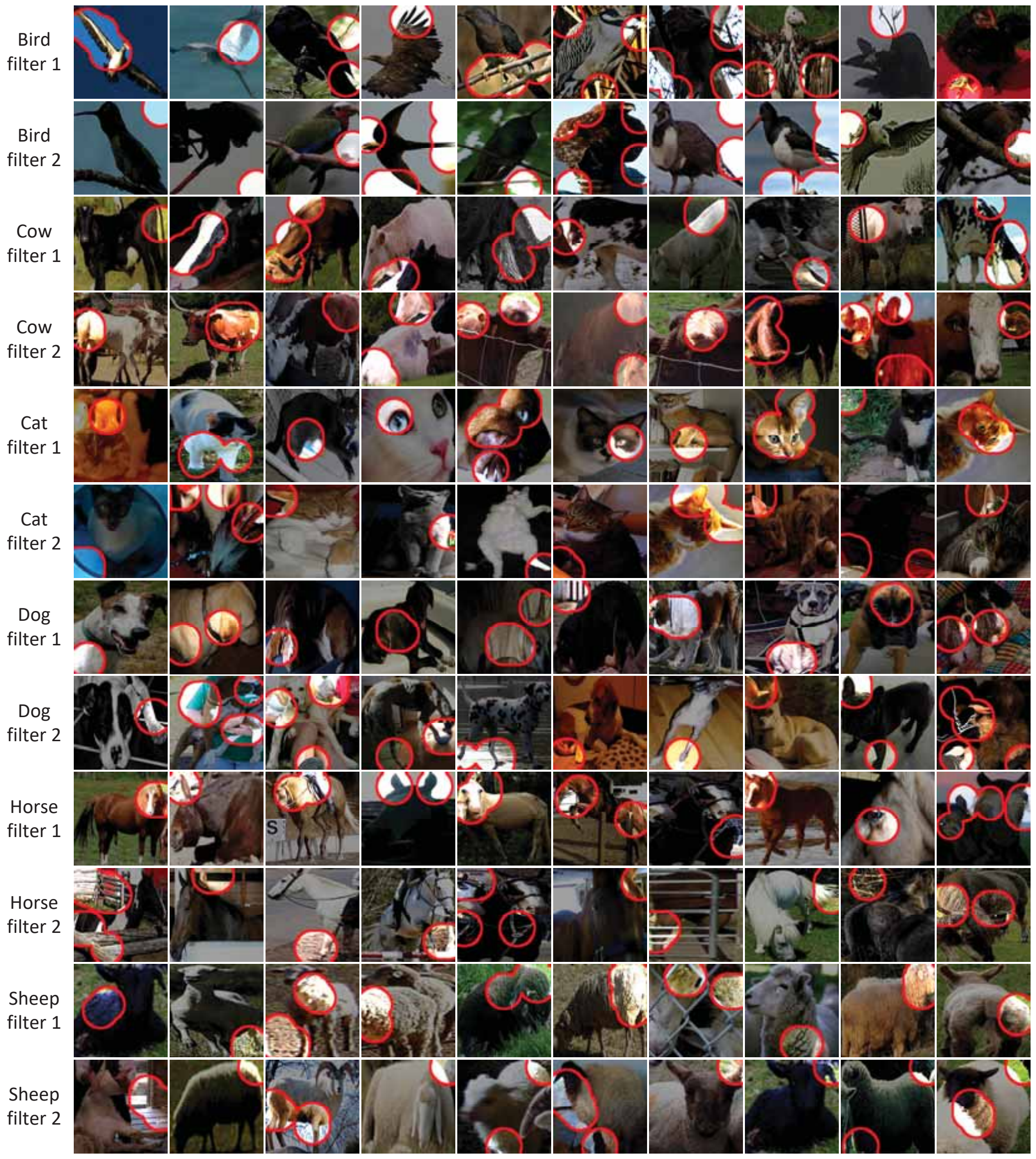
B Visualization of feature maps of the explainer and feature maps of the performer



Visualization of feature maps in the *conv-interp-2* layer of the explainer. Each row corresponds to feature maps of a filter in the *conv-interp-2* layer. We simply used a round RF for each neural activation and overlapped all RFs for visualization.



Visualization of feature maps in the *conv-interp-2* layer of the explainer. Each row corresponds to feature maps of a filter in the *conv-interp-2* layer. We simply used a round RF for each neural activation and overlapped all RFs for visualization.

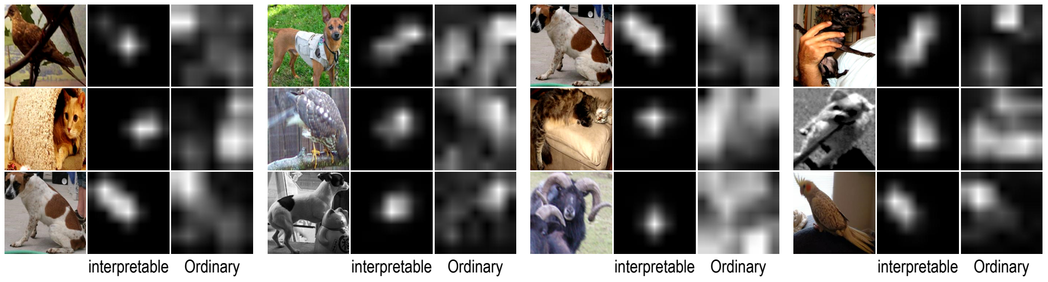


Visualization of feature maps in the top conv-layer of the performer. Each row corresponds to feature maps of a filter in the top conv-layer. We simply used a round RF for each neural activation and overlapped all RFs for visualization.

C Ensuring the explainer to mimic the performer

A high reconstruction quality ensures the explainer successfully mimics the signal processing in the performer. Let us assume the explainer mistakenly uses an object part for prediction, which the performer does not use. Then, when we add or remove the target part from an testing image, the explainer will generate a obviously different feature, but the performer will not; vice versa. In this way, given sufficient training samples, a high reconstruction quality can ensures the explainer approximate the logic of signal processing in the performer.

D Grad-CAM attention maps



Grad-CAM attention maps. We used (Selvaraju et al., 2017) to compute grad-CAM attention maps of interpretable features of the *conv-interp-2* layer in the explainer and those of ordinary features of the top conv-layer in the performer. Interpretable filters in the *conv-interp-2* layer focused on distinct object parts, while ordinary filters in the performer separated its attention to both textures and parts.

The quantitative analysis in Figure 4 of the paper shows an example of how to use the disentangled object-part features to quantitatively evaluate contributions of different parts to the output score of object classification. Based on prior semantic meanings of the interpretable filters, we show a prior explanation of the logic in the classification without manually checking activation distributions of each channel of the feature map. Thus, this is different from the visualization of CNN representations, which requires people to manually check the explanation based on visualization results.

E Detailed results of p values

	Pascal-Part dataset (Chen et al., 2014)								CUB200-2011 (Wah et al., 2011)
	Single-category							Multi-category	
	bird	cat	cow	dog	horse	sheep	Avg.	Avg.	Avg.
AlexNet	0.75	0.72	0.69	0.72	0.70	0.70	0.71	–	0.5810
VGG-M	0.81	0.80	0.81	0.80	0.81	0.81	0.81	0.9012	0.8611
VGG-S	0.91	0.90	0.90	0.90	0.90	0.90	0.90	0.9270	0.9533
VGG-16	0.88	0.89	0.87	0.87	0.86	0.88	0.87	0.8593	0.9579
ResNet-101	0.98	0.97	0.97	0.97	0.98	0.97	0.97	–	–
ResNet-152	0.98	0.98	0.98	0.98	0.98	0.99	0.98	–	–

Table 7: p values of explainers.

F More results of location instability

In this supplementary material, we add another baseline for comparison. Because we took feature maps of the *relu4* layer of the AlexNet/VGG-M/VGG-S (the 12th/12th/11th layer of the AlexNet/VGG-M/VGG-S) and the *relu5-2* layer of the VGG-16 (the 28th layer) as target feature maps to be explained, we sent feature maps of these layers feature into explainer networks to disentangle them. Thus, we measured the location instability of these target feature maps as the new baseline.

The following two tables show that our explainer networks successfully disentangled these target feature maps, and the disentangled feature maps in the explainer exhibited much lower location instability.

	Single-category						
	bird	cat	cow	dog	horse	sheep	Avg
AlexNet (the *relu4* layer)	0.152	0.130	0.140	0.127	0.143	0.139	0.139
AlexNet (the top conv-layer)	0.153	0.131	0.141	0.128	0.145	0.140	0.140
Explainer	**0.104**	**0.089**	**0.101**	**0.083**	**0.098**	**0.103**	**0.096**
VGG-M (the *relu4* layer)	0.148	0.127	0.138	0.126	0.140	0.137	0.136
VGG-M (the top conv-layer)	0.152	0.132	0.143	0.130	0.145	0.141	0.141
Explainer	**0.106**	**0.088**	**0.101**	**0.088**	**0.097**	**0.101**	**0.097**
VGG-S (the *relu4* layer)	0.148	0.127	0.136	0.125	0.139	0.137	0.135
VGG-S (the top conv-layer)	0.152	0.131	0.141	0.128	0.144	0.141	0.139
Explainer	**0.110**	**0.085**	**0.098**	**0.085**	**0.091**	**0.096**	**0.094**
VGG-16 (the *relu5-2* layer)	0.151	0.128	0.145	0.124	0.146	0.146	0.140
VGG-16 (the top conv-layer)	0.145	0.133	0.146	0.127	0.143	0.143	0.139
Explainer	**0.095**	**0.089**	**0.097**	**0.085**	**0.087**	**0.089**	**0.090**

Table 8: Location instability of feature maps in performers and explainers. Performers are learned based on the Pascal-Part dataset (Chen et al., 2014)

AlexNet (*relu4* layer)	0.1542
AlexNet (the top conv-layer)	0.1502
Explainer	**0.0906**
VGG-M (*relu4* layer)	0.1484
VGG-M (the top conv-layer)	0.1476
Explainer	**0.0815**
VGG-S (*relu4* layer)	0.1518
VGG-S (the top conv-layer)	0.1481
Explainer	**0.0704**
VGG-16 (*relu5-2* layer)	0.1444
VGG-16 (the top conv-layer)	0.1373
Explainer	**0.0490**

Table 9: Location instability of feature maps in performers and explainers. Performers are learned based on the CUB200-2011 dataset (Wah et al., 2011)

G ABOUT EVALUATION METRIC

The location instability is designed to evaluate the fitness between an intermediate-layer filter f and the representation of a specific object part, and it has been widely used in (Zhang et al., 2018c;a). Therefore, we used this metric for evaluation in our experiments. In fact, there is another metric to identify semantics of CNN filters, which was proposed by (Bau et al., 2017). This study annotated pixel-level labels for six kinds of semantics (*objects*, *parts*, *scenes*, *textures*, *materials*, and *colors*) on testing images. Then, given a feature map of a filter f, they used the intersection-of-union (IoU) between activation regions in the feature map and image regions of each kind of semantics to identify the semantic meaning of this filter. *I.e.* for filters oriented to representations of object parts, this metric measures whether or not activation regions in a feature map greatly overlap to ground-truth segment of a specific object part. However, in this study, we disentangled original feature maps into object parts without any ground-truth annotations of object parts. The disentangled object parts usually represent joint regions of ground-truth parts, sub-regions of ground-truth parts, or combinations of small ground-truth parts, although each disentangled object part consistently describes the same part through different objects. Therefore, when filters are learned without ground-truth part annotations, the metric in (Bau et al., 2017) is less suitable to evaluate the object-part semantics than the metric of location instability (Zhang et al., 2018a).

H UNDERSTANDING OF FILTER LOSSES

According to (Zhang et al., 2019), we can re-write the filter loss as

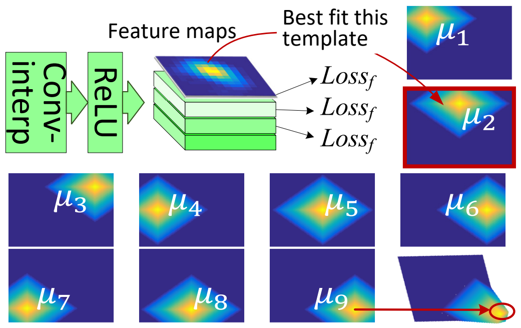
$$\mathbf{Loss}_f = -H(\mathbf{P}) + H(\mathbf{P}'|\mathbf{X}) + \sum\nolimits_{x_f\in\mathbf{X}} p(\mathbf{P}^+, x_f) H(\mathbf{P}^+|X = x_f)$$

where $\mathbf{P}' = \{\emptyset, \mathbf{P}^+\}$. $H(\mathbf{P}) = -\sum_{\mu \in \mathbf{P}} p(\mu) \log p(\mu)$ is a constant prior entropy of part-location candidates (here $\mu = \emptyset$ is a dummy location candidate).

To compute above mutual information, $p(\mu)$ is defined as constant. $p(x_f|\mu)$ is given as

$$p(x_f|\mu) = p(x_f|T_\mu) = \frac{1}{Z_\mu} \exp\left[tr(x_f \cdot T_\mu)\right] \tag{3}$$

where T_μ denotes the part template corresponding to the part location μ, as shown in the following figure. $Z_\mu = \sum_{x_f} \exp[tr(x_f \cdot T_\mu)]$. $tr(\cdot)$ indicates the trace of a matrix.



Part template T_μ corresponding to each part location μ (Zhang et al., 2019)

Low inter-category entropy: The second term $H(\mathbf{P}' = \{\emptyset, \mathbf{P}^+\}|\mathbf{X})$ is computed as $H(\mathbf{T}' = \{\emptyset, \mathbf{P}^+\}|\mathbf{X}) = -\sum_{x_f} p(x_f) \sum_{\mu \in \{\emptyset, \mathbf{P}^+\}} p(\mu|x_f) \log p(\mu|x_f)$, where $\mathbf{P}^+ = \{\mu_1, \mu_2, \ldots, \mu_{L^2}\} \subset \mathbf{P}$ and $p(\mathbf{P}^+|x_f) = \sum_{i=1}^{L^2} p(\mu_i|x_f)$. We define the set of all valid part locations in $\mathbf{P}^+$ as a single label to represent the category c. We use a negative template $\emptyset$ to denote other categories. This term encourages a low conditional entropy of inter-category activations, *i.e.* a well-learned filter f needs to be exclusively activated by a certain category c and keep silent on other categories. The feature map x_f can usually identify whether the input image belongs to category c or not, *i.e.* x_f fitting to either a valid part location $\hat{\mu} \in \mathbf{P}^+$ or $\emptyset$, without great uncertainty.

Low spatial entropy: The third term is given as $H(\mathbf{P}^+|X = x_f) = \sum_{i=1}^{L^2} \tilde{p}(\mu_i|x_f) \log \tilde{p}(\mu_i|x_f)$, where $\tilde{p}(\mu_i|x_f) = \frac{p(\mu_i|x_f)}{p(\mathbf{P}^+|x_f)}$. This term encourages a low conditional entropy of spatial distribution of x_f's activations. *I.e.* given an image $I \in \mathbf{I}_c$, a well-learned filter should only be activated by a single region $\hat{\mu} \in \mathbf{P}^+$ of the feature map x_f, instead of being repetitively triggered at different locations.

Optimization of filter losses: The computation of gradients of the filter loss *w.r.t.* each element $x_f^{(ij)}$ of feature map x_f is time-consuming. (Zhang et al., 2019) computes an approximate but efficient gradients to speed up the computation, as follows.

$$\begin{aligned}\frac{\partial \mathbf{Loss}_f}{\partial x_f^{(ij)}} &= \sum_{\mu \in \mathbf{P}} \frac{p(\mu) t_{ij} e^{tr(x_f \cdot T_\mu)}}{Z_\mu} \left\{ tr(x_f \cdot T_\mu) - \log\left[Z_\mu p(x_f)\right] \right\} \\ &\approx \frac{p(\hat{\mu}) \hat{t}_{ij}}{Z_{\hat{\mu}}} e^{tr(x_f \cdot \hat{T})} \left\{ tr(x_f \cdot \hat{T}) - \log[Z_{\hat{\mu}} p(x_f)] \right\}\end{aligned}$$

where $\hat{T}$ is the target template for feature map x_f. Let us assume that there are multiple object categories C. We simply assign each filter f with the category $\hat{c} \in C$ whose images activate f the most, *i.e.* $\hat{c} = \text{argmax}_c \mathbb{E}_{x_f : I \in \mathbf{I}_c} \sum_{ij} x_f^{(ij)}$. If the input image I belongs to the filter f's target category, then $\hat{T} = T_{\hat{\mu}}$, where $\hat{\mu} = \text{argmax}_{\mu=[i,j]} x_f^{(ij)}$. If image I belongs to other categories, then $\hat{T} = \emptyset$. Considering $\forall \mu \in \mathbf{P} \setminus \{\hat{\mu}\}$, $e^{tr(x_f \cdot \hat{T})} \gg e^{tr(x_f \cdot T_\mu)}$ after initial learning epoches, we can make approximations in the above equation.

Note that above assignments of object categories are also used to compute location instability for intermediate-layer filters to evaluate their interpretability.

Inspired by optimization tricks in (Zhang et al., 2019), we updated the parameter $\lambda_f = \frac{1}{300N}\mathbb{E}_{x_f}[\|\frac{\partial Loss_{rec}}{\partial x_f}\|]/\mathbb{E}_{x_f}[\|\frac{\partial Loss_f}{\partial x_f}\|]$ for the N-th learning epoch in an online manner, where $\frac{\partial Loss_{rec}}{\partial x_f}$ denotes gradients of reconstruction losses obtained from upper layers. In particular, given performers for single-category classification, we simply used feature maps of positive images (*i.e.* objects of the target category) to approximately estimate the parameter $\boldsymbol{\alpha}$ for the norm-layer, because positive images can much more strongly trigger interpretable filters than negative images. Thus, computing $\boldsymbol{\alpha}$ based on positive images made p accurately measure the contribution ratio of the interpretable track when the network made predictions to positive images. We will clarify all these settings when the paper is accepted. In experiments, for interpretable filters in the *conv-interp-2* layer, we added the filter loss to $x'_f = p \cdot x_f + (1-p) \cdot \hat{x}_{\text{ordin}}$, where $\hat{x}_{\text{ordin}} \in \mathbb{R}^{L \times L}$ denotes a channel of x_{ordin} that corresponds to the channel of filter f. We found that this modification achieved more robust performance than directly applying the filter loss to x_f. In this case, the filter loss encouraged a large value of p and trained the interpretable filter f, but we did not pass gradients of the filter loss to the ordinary track.

I About parameter settings

In the experimental section, we have clarified settings for parameters. We simply set $\eta = 1.0 \times 10^6$ for the AlexNet, VGG-M, and VGG-S without sophisticatedly turning the value of η. We set $\eta = 1.5 \times 10^5$ for the VGG-16, since the VGG-16 has more conv-layers than the other networks. For each type of CNNs, the same value of η was uniformly applied to various CNNs for different categories. Our method is not sensitive to η. When the VGG-16 used the η value of the AlexNet, it only changed an average location instability of 0.003 over all experiments.

J Evaluating the reconstruction quality based on the object-classification accuracy

In order to evaluate the feature-reconstruction quality, we used the classification accuracy based on explainer features as an evaluation metric. We fed output features of the explainer back to the performer for classification. Theoretically, a high classification accuracy may demonstrate that the explainer can well reconstruct performer features without losing much information. Note that explainers were learned to reconstruct feature maps of performers, rather than optimizing the classification loss, so explainers could only approximate the classification performance of performers but could not outperform performers.

In addition, we added another baseline, namely *Explainer+cls*, which used the object-classification loss to replace the reconstruction loss to learned explainer networks. Thus, output features of *Explainer+cls* exhibited higher classification accuracy than features of the original explainer.

The following table compares the classification accuracy between the performer and the explainer. For multi-category classification, the performance of explainers was quite close to that of performers. Learning explainers with classification losses exhibited significantly better classification performance than learning explainers with reconstruction losses. Because *Explainer+cls* directly learned from the classification loss, *Explainer+cls* sometimes even outperformed the performer.

	Pascal-Part (Chen et al., 2014)						CUB200 (Wah et al., 2011)		
	Multi-category			Single-category					
	Performer	Explainer	Explainer+cls	Performer	Explainer	Explainer+cls	Performer	Explainer	Explainer+cls
AlexNet	–	–	–	4.60%	8.20%	2.88%	4.41%	10.98%	3.57%
VGG-M	6.12%	6.62%	5.22%	3.18%	8.58%	3.40%	2.66%	6.84%	2.54%
VGG-S	5.95%	6.97%	5.43%	2.26%	10.97%	3.86%	2.76%	8.53%	2.72%
VGG-16	2.03%	2.17%	2.49%	1.34%	6.12%	1.76%	1.09%	6.04%	0.90%

Table 10: Classification errors based on feature maps of performers and explainers.