Unsupervised Learning of Neural Networks to Explain Neural Networks

Quanshi Zhang[b,a], **Yu Yang**[a], **Yuchen Liu**[c], **Ying Nian Wu**[a], **and Song-Chun Zhu**[a]
[a]University of California, Los Angeles [b]Shanghai Jiao Tong University
[c]University of California, Irvine.

Abstract

This paper presents an unsupervised method to learn a neural network, namely an *explainer*, to interpret a pre-trained convolutional neural network (CNN), *i.e.* explaining knowledge representations hidden in middle conv-layers of the CNN. Given feature maps of a certain conv-layer of the CNN, the explainer performs like an auto-encoder, which first disentangles the feature maps into object-part features and then inverts object-part features back to features of higher conv-layers of the CNN. More specifically, the explainer contains interpretable conv-layers, where each filter disentangles the representation of a specific object part from chaotic input feature maps. As a paraphrase of CNN features, the disentangled representations of object parts help people understand the logic inside the CNN. We also learn the explainer to use object-part features to reconstruct features of higher CNN layers, in order to minimize loss of information during the feature disentanglement. More crucially, we learn the explainer via network distillation without using any annotations of sample labels, object parts, or textures for supervision. We have applied our method to different types of CNNs for evaluation, and explainers have significantly boosted the interpretability of CNN features.

1 Introduction

Convolutional neural networks (CNNs) [18, 16, 10] have achieved superior performance in many visual tasks, such as object classification and detection. In contrast to the significant discrimination power, the model interpretability has always been an Achilles' heel of deep neural networks.

In this paper, we aim to boost the interpretability of feature maps of a middle conv-layer in a CNN. *Without additional human supervision, can we automatically disentangle human-interpretable features from chaotic middle-layer feature maps?* For example, we attempt to force each channel of the disentangled feature map to represent a certain object part. Our task of improving network interpretability by exploring more interpretable middle-layer features is different from the passive visualization [39, 22, 29, 7, 9, 27] and diagnosis [2, 14, 24] of CNN representations.

A major issue with network interpretability is that high interpretability is not necessarily equivalent to, and sometimes conflicts with a high discrimination power [2]. For example, disentangling middle-layer representations of a CNN into object parts [44] may sometimes decrease the classification performance. Thus, people usually have to trade off between the network interpretability and the network performance in real applications.

Tasks: In order to solve the above dilemma, given a pre-trained CNN, we propose to learn an additional neural network, namely an *explainer* network, to explain features inside the CNN. Accordingly, we call the pre-trained CNN a *performer* network. As shown in Fig. 1, the explainer network explains feature maps of a certain conv-layer of the performer network. We attach the explainer onto the performer like an appendix without affecting the original discrimination power of the performer.

Preprint. Work in progress.

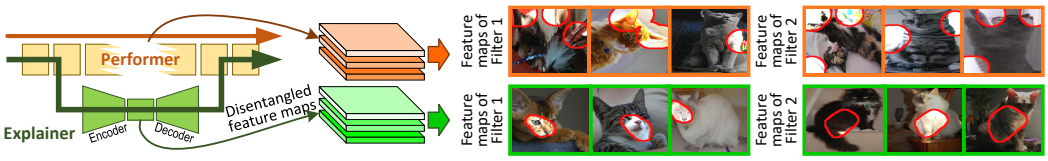


Figure 1: Explainer network. We use an explainer network (green) to disentangle the feature map of a certain conv-layer in a pre-trained performer network (orange). The explainer network disentangles input features into object-part feature maps to explain knowledge representations in the performer, *i.e.* making each filter represent a specific object part. The explainer network can also invert the disentangled object-part features to reconstruct features of the performer without much loss of information. We compare ordinary feature maps in the performer and the disentangled feature maps in the explainer on the right. The orange and green lines indicate information-pass routes for the inference process and the explanation process, respectively.

The explainer performs like an auto-encoder. The encoder in the explainer contains hundreds of interpretable filters, each of which disentangles an object part from input feature maps. The decoder inverts the disentangled object-part features to reconstruct features of upper layers of the performer.

As shown in Fig. 1, the feature map of each filter in the performer usually represents a chaotic mixture of object parts and textures, whereas as a paraphrase of performer features, the disentangled object-part features in the explainer provide an insightful understanding of the performer. People can obtain answers to the following two questions from the disentangled features.

1. What part patterns are encoded in the performer?
2. Which patterns are activated and used for each specific prediction?

Learning: We learn the explainer by distilling feature representations from the performer to the explainer without any additional supervision. No annotations of sample labels, parts, or textures are used to guide feature disentanglement during the learning process. We add a loss to each interpretable filter in the explainer (see Fig. 2). The filter loss encourages an interpretable filter 1) to represent objects of a single category, and 2) to 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[1]. Thus, the filter loss pushes each interpretable filter towards the representation of an object part.

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

Potential values of the explainer: The high interpretability of a middle conv-layer is of great value when the performer network needs to earn trust from people. As discussed in [43], due to potential dataset bias and representation bias, a high accuracy on testing images cannot fully ensure that the performer encodes correct features. People usually need clear explanations for middle-layer features of a performer, in order to semantically or visually clarify the logic of each prediction made by the performer. For example, in Fig. 5, we use the disentangled object-part features to quantitatively identify which parts are learned and used for the prediction. Given an input image, some previous studies [9, 27, 24] estimated the top contributing image regions in each prediction. In comparisons, our explainer ensures each interpretable filter to exclusively describe a single object part of a category, which provides more fine-grained knowledge structures inside the performer at the part level.

Contributions: We can summarize the contributions of this study as follows. 1) We tackle a new problem, *i.e.* learning an explainer network to mine and clarify potential feature components that are encoded in middle layers of a pre-trained performer network. The explainer disentangles chaotic feature maps of the performer into human-interpretable object parts. Compared to directly letting the performer encode disentangled/interpretable features, learning an additional explainer does not affect the discrimination power of the performer, thereby ensuring broader applicability. 2) We propose a simple yet effective method to learn the explainer without any annotations of object parts or textures for supervision. 3) Theoretically, besides the proposed explainer, our explainer-performer network

[1] *E.g.*, we consider the left and right eyes as two different parts, because they have different contexts.

structure also supports knowledge distillation into new explainers with novel interpretability losses. Meanwhile, the explainer can be broadly applied to different CNN performers. 4) Experiments show that our approach has considerably improved the feature interpretability.

2 Related work

Network interpretability: Recent studies of the interpretability of visual network are reviewed in [46]. Instead of analyzing network features from a global view [36, 26, 23], Bau *et al.* [2] defined six kinds of semantics for middle-layer feature maps of a CNN, *i.e. objects*, *parts*, *scenes*, *textures*, *materials*, and *colors*. We can roughly consider the first two semantics as object-part patterns with specific shapes, and summarize the last four semantics as texture patterns. In this study, we use the explainer to disentangle object-part patterns from feature maps.

Many studies for network interpretability mainly visualized image appearance corresponding to a neural unit inside a CNN [39, 22, 29, 7, 38, 6] or extracted image regions that were responsible for network output [24, 8, 14, 9, 27, 17]. Other studies retrieved mid-level representations with specific meanings from CNNs for various applications [15, 35, 32, 19]. For example, Zhou *et al.* [47, 48] selected neural units to describe "scenes". Simon *et al.* discovered objects from feature maps of unlabeled images [28]. Zhang *et al.* [41, 42] extracted certain neural units to describe an object part in a weakly-supervised manner. Zhang *et al.* [40] also disentangled feature representations in middle layers of a CNN into a graphical model of object parts in an unsupervised manner. In fact, each filter in a middle 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, interpretable representations [12, 31, 21]. Many studies learn interpretable representations in a weakly-supervised or unsupervised manner. For example, capsule nets [25] and interpretable RCNN [37] learned interpretable middle-layer features. InfoGAN [3] and β-VAE [11] learned meaningful input codes of generative networks. The study of interpretable CNNs [44] developed a loss to push each middle-layer filter towards the representation of a specific object part during the learning process without given part annotations, which is the closest to our research. However, as mentioned in [2], an interpretable model cannot always ensure a high discrimination power, which limits the applicability of above interpretable models. Therefore, instead of directly boosting the interpretability of the performer network, we propose to learn an explainer network in an unsupervised fashion.

Meta-learning: Our study is also related to meta-learning [5, 1, 20, 34]. Meta-learning uses an additional model to guide the learning of the target model. In contrast, our research uses an additional explainer network to interpret middle-layer features of the target performer network.

3 Algorithm

3.1 Explainer structure

As shown in Fig. 2, the explainer network has two modules, *i.e.* an encoder and a decoder, which transform performer features into interpretable object-part features and invert object-part features back to features of the performer, respectively. We applied the encoder and decoder with following structures to all types of performers in all experiments. However, people can edit the number of conv-layers and fully-connected (FC) layers in their specific applications.

Encoder: The encoder contains two tracks, namely an *interpretable track* and an *ordinary track*. The interpretable track disentangles performer features into object parts. 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 layer contains interpretable filters, and each interpretable filter can be only triggered by a specific object part. Since exclusively using object-part features is not enough to reconstruct performer features, we also design an ordinary track as a supplement to the interpretable track, in order to represent textural features that cannot be modeled by the interpretable track. The ordinary track contains a conv-layer (namely *conv-ordin*), a ReLU layer, and a pooling layer.

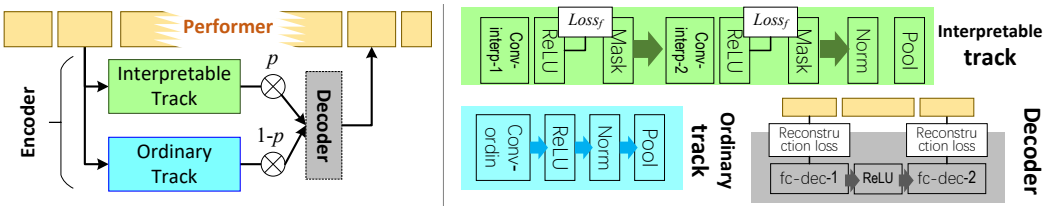


Figure 2: Overall structure of the explainer network (left). Detailed structures within the interpretable track, the ordinary track, and the decoder are shown on the right. Theoretically, people can edit the number of conv-layers and FC layers within the encoder and the decoder for their own applications.

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 softmax 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.

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 the tensor x. We can update $\boldsymbol{\alpha}$ during the learning process, which is similar to the learning for batch normalization.

Mask layer: We add mask layers after each of the two interpretable conv-layers to further remove noisy activations that are unrelated to the target object part. 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 the coordinate of a neural unit in x_f ($1 \le i, j \le L$), and $x_f^{(ij)}$ indicates the activation value of this unit. Based on the part-localization result $\hat{\mu}$, the mask layer assigns a mask $mask_f$ to x_f to remove noises. *I.e.* $x_f^{\text{masked}} = x_f \circ mask_f$ is the output feature map, where $\circ$ denotes the Hadamard (element-wise) 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 that will be introduced later. When $mask_f$ is determined, we treat $mask_f$ as a constant to enable gradient back-propagation.

Decoder: The decoder inverts x_{enc} to x_{dec}, which reconstructs performer features. The decoder has two FC layers, which followed by two ReLU layers. We use the two FC layers, namely *fc-dec-1* and *fc-dec-2*, to reconstruct feature maps of two corresponding FC layers in the performer. The reconstruction loss will be introduced later. The better reconstruction of the FC features indicates that the explainer loses less information during the computation of x_{enc}.

3.2 Learning

When we distill knowledge representations from the performer to the explainer, we consider the following three terms: 1) the quality of network distillation, *i.e.* the explainer needs to well reconstruct feature maps of upper layers in the performer, thereby minimizing the information loss; 2) the interpretability of feature maps of the interpretable track, *i.e.* each filter in *conv-interp-2* should exclusively represent a certain object part; 3) the relative contribution of the interpretable track *w.r.t.* the ordinary track, *i.e.* we hope the interpretable track to make much more contribution to the final CNN prediction than the ordinary track. Therefore, we design the following loss for each input image to learn the explainer.

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

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 $\boldsymbol{\alpha}$ for norm-layers. $\lambda_{(l)}, \lambda_f$ and η are scalar weights.

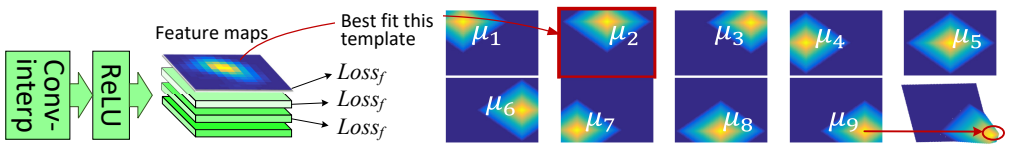


Figure 3: Templates of $\{T_{\mu_i}\}$ for the toy case of $L = 3$. Each template T_{μ_i} represents the ideal activation shape when the target part is localized at the i-th unit of x_f. In real applications, we usually use 6^2–14^2 templates since $6 \leq L \leq 14$.

The first term $\|x_{(l)} - x^*_{(l)}\|^2$ is the reconstruction loss, where $x_{(l)}$ 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 CNN prediction. **The third term** $Loss_f(x_f)$ is the loss of filter interpretability. Without annotations of object parts, the filter loss forces x_f to be exclusively triggered by a specific object part of a certain category. 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. Note that w_p for p is updated based on both the $-\log p$ loss and other losses. The $-\log p$ loss encourages a high value of p, while the reconstruction loss usually requires a moderate value of p to balance neural activations from the interpretable and ordinary tracks to ensure a good reconstruction.

3.2.1 Filter loss

As a prior measurement of the fitness between a filter and an object part, the filter loss was formulated in [44] as the minus mutual information between feature maps and a set of pre-defined templates.

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

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. $\mathbf{T} = \{T_{\mu_1}, T_{\mu_2}, \ldots, T_{\mu_{L^2}}, T^-\}$ is referred to as a set of pre-defined templates. $p(T)$ is a constant prior probability of a template T, and $p(x_f|T)$ measures the fitness between x_f and T. Thus, we get $Loss_f(x_f) = -\sum_{T\in\mathbf{T}} p(x_f, T) \log \frac{p(x_f|T)}{p(x_f)}$.

The first L^2 templates, *i.e.* $T_{\mu_1}, T_{\mu_2}, \ldots, T_{\mu_{L^2}}$. $T_{\mu_i} \in \mathbb{R}^{L\times L}$ represents the ideal distributions of neural activations in the feature map x_f when the target part mainly triggers the unit μ_i in x_f ($i \in \{1, 2, \ldots, L^2\}$). Fig. 3 illustrates the L^2 templates for a toy case of $L = 3$. Besides the L^2 positive templates, we use a negative template T^- ($T^{-,(ij)} = -\tau$) for the case that the input image does not contain the target object so that the feature map should remain inactivated. τ is a positive constant. **1)** When the input image contains the target object, we expect x_f to be activated by a single part at a single location in the feature map. Thus, x_f should match only one of the L^2 positive templates. **2)** When the input image does not contain the target object, we expect x_f not to be triggered, thereby matching the negative template T^-. In this way, for each input image, its feature map x_f needs to match exactly one of the $(L^2 + 1)$ templates. Thus, high mutual information in Equation (2) indicates a high probability that the filter f represents the same object part through different images.

More specifically, $p(x_f|T)$ is given as $p(x_f|T) = \frac{1}{Z_T}\exp[tr(x_f \cdot T)]$, where $Z_T = \sum_{x_f\in\mathbf{X}} \exp[tr(x_f \cdot T)]$. $tr(\cdot)$ indicates the trace of a matrix, *i.e.* $tr(x_f \cdot T) = \sum_{ij} x_f^{(ij)} T^{(ji)}$. $p(x_f) = \sum_T p(T)p(x_f|T)$.

4 Experiments

In experiments, we learned explainers for performer networks with three types of structures 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 pre-trained performer.

Benchmark datasets: Because the evaluation of filter interpretability required ground-truth annotations of object landmarks[2] (parts), we used two benchmark datasets with part annotations for training and testing, *i.e.* the CUB200-2011 dataset [33] and the Pascal-Part dataset [4]. Note that previous studies [4, 41, 44] 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 [41, 44] that selected the seven animal categories in the two datasets for evaluation. Both the datasets provide object bounding boxes. The CUB200-2011 dataset [33] 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 [4] provides ground-truth segmentations of a total of 107 object parts for six animal categories.

Four types of CNNs as performers: We applied our method to four types of performers, including the AlexNet [16], the VGG-M network [30], the VGG-S network [30], the VGG-16 network [30]. For residual networks [10] and dense networks [13], skip connections in these networks usually make a single feature map contain rich features from different conv-layers, which increases the difficulty of understanding its feature maps. Thus, to simplify the story, we did not test the performance on residual networks and dense networks.

Two experiments: We followed experimental settings in [44] to conduct two experiments, *i.e.* an experiment of single-category classification and an experiment of multi-category classification. For single-category classification, we learned four performers with structures of the AlexNet [16], VGG-M [30], VGG-S [30], and VGG-16 [30] for each of the seven animal categories in the two benchmark datasets. Thus, we learned 28 performers, and each performer 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 learned the VGG-M [30], VGG-S [30], and VGG-16 [30] to classify the six animal categories in the Pascal-Part dataset [4].

Experimental details: We considered 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 layers. We sent feature maps of the target layer in each performer network to an explainer network. We learned the explainer network to disentangle these feature maps for testing. The output of the explainer reconstructed the feature of the *fc7* layer of the performer (the 19th/19th/18th/35th layer of the AlexNet/VGG-M/VGG-S/VGG-16) 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. Each conv-layer in the explainer had D filters with a $3 \times 3 \times D$ 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. We initialized the *conv-interp-2* and *conv-ordin* layers with random weights. The *conv-interp-1* layer in the explainer was initialized using weights of the *conv5*/*conv5*/*conv5*/*conv5-3* layer of the AlexNet/VGG-M/VGG-S/VGG-16. 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 considering the last pooling layer in the performer.

Theoretically, we can use different activation shapes for T_{μ_i} to construct the filter loss. For simplicity, we used part templates defined in [44] in our experiments. We set $\eta = 1.0 \times 10^6$ for the AlexNet, VGG-M, and VGG-S and set $\eta = 1.5 \times 10^5$ for the VGG-16, since the VGG-16 has more conv-layers than the other networks. We set $\lambda_{(l)} = 5 \times 10^4 / \mathbb{E}_{x^*_{(l)}}[\|\max(x^*_{(l)}, 0)\|]$ in all experiments, where the expectation was averaged over features of all images. We followed [44] to set templates $\mathbf{T}$ with the parameter τ and $p(T) = \frac{1}{L^2+1}$. Directly minimizing $Loss_f(x_f)$ is time-consuming, and [44]

[2]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 [44], which selected the *head*, *neck*, and *torso* of each category in the Pascal-Part dataset [4] as the landmarks and used the *head*, *back*, *tail* of birds in the CUB200-2011 dataset [33] as landmarks. It was because these landmarks appeared on testing images most frequently.

Table 1: Location instability of feature maps in performers and explainers for the evaluation of filter interpretability. Please see the appendix for comparisons with more baselines.

Results based on the Pascal-Part dataset [4]

	Single-category							Multi-category
	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**

CUB200-2011 dataset [33]

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

Table 2: Average p values of explainers in different experiments.

	Pascal-Part [4]		CUB200
	Single	Multi	-2011 [33]
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

Table 3: Multi-category classification errors using features of performers and explainers based on the Pascal-Part dataset [4]. We evaluated loss of feature information in explainers based on the classification-error gap between performers and explainers. Please see the appendix for more results.

	Performer	Explainer	Δ Error	Performer+cls	Δ Error
VGG-M	6.12%	6.62%	0.5%	5.22%	-0.9%
VGG-S	5.95%	6.97%	1.02%	5.43%	-0.52%
VGG-16	2.03%	2.17%	0.14%	2.49%	0.46%

formulated approximate gradients of $\lambda_f Loss_f(x_f)$ to speed up the computation. Please see the appendix for more details.

Evaluation metric: We compared the object-part interpretability between feature maps of the explainer and those of the performer. According to above network structures, 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 [2], 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.

We used the location instability as the evaluation metric, which has been widely used to measure the fitness between a filter f and the representation of a specific object part [44]. Given a feature map x_f, we localized the part at the unit $\hat{\mu}$ with the highest activation. We used [47] 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[2] 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 $\mathbf{p}_{\hat{\mu}}$ may change 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 [44] for computational details of this evaluation metric.

4.1 Experimental results and analysis

Table 1 compares the interpretability between feature maps in the performer and feature maps in the explainer. Feature maps in our explainers were much more interpretable than feature maps in 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

Figure 4: Visualization of interpretable filters in the explainer and ordinary filters in the performer. As discussed in [2], the top conv-layer of a CNN is the more likely to represent object parts than low conv-layers. We visualized and compared filters in the top conv-layer of the performer and interpretable filters in the *conv-interp-2* layer of the explainer. We used [47] to estimate the RF[3] of neural activations to illustrate a filter's semantics. Interpretable filters are much more semantically meaningful than ordinary filters. Please see the appendix for more results.

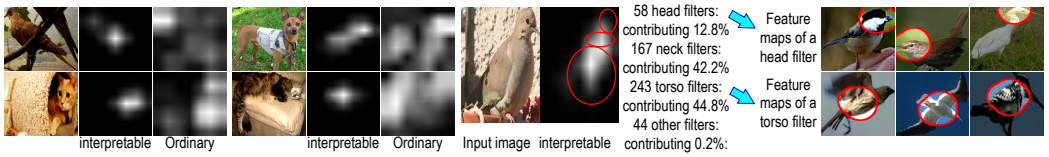


Figure 5: Grad-CAM attention maps and quantitative analysis. We used [27] to compute grad-CAM attention maps of interpretable feature maps in the explainer and ordinary feature maps in the performer. Interpretable filters focused on a few distinct object parts, while ordinary filters separated its attention to both textures and parts. We can assign each interpretable filter with a semantic part. For example, we 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 [45] to compute quantitative contributions of different parts to the classification score on the right. Please see the appendix for more results.

indicates the quantitative ratio of the contribution from interpretable features. Table 2 lists p values of explainers that were learned for different performers. When we used an explainer to interpret feature maps of a VGG network, about 80%–96% activation scores came from interpretable features. To evaluate feature reconstructions of an explainer, we fed the reconstructed features back to the performer for classification. As shown in Table 3, 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 3) to measure the information loss during feature transformation in the explainer. Furthermore, we added another baseline (*Explainer+cls*), which used the classification loss to replace the reconstruction loss to learned explainers. *Explainer+cls* learned from the classification loss, rather than distill knowledge from performers, so we did not consider feature maps in *Explainer+cls* as the explanation of performer features. However, with the help of the classification loss, *Explainer+cls* outperformed the performer in classification, which indicates potential boundaries of the discrimination power of explainers.

Visualization of filters: We used the visualization method proposed by Zhou *et al.* [47] 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 [47], the computed RF represented image regions that were responsible for neural activations of a filter, which was much smaller than the theoretical size of the RF. Fig. 4 used RFs[3] 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 [27] 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 semantic meanings.

[3]When an ordinary filter in the performer does not have consistent contours, it is difficult for [47] 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.

5 Conclusion and discussions

In this paper, we have proposed a theoretical solution to a new task, *i.e.* learning an explainer network to disentangle and explain feature maps of a pre-trained performer network. Learning an explainer besides the performer does not decrease the discrimination power of the performer, which ensures the broad applicability. 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 for each specific application.

We have applied our method to different types of performers, and experimental results show that our explainers can disentangle most information of input feature maps into object-part feature maps, which significantly boosts the feature interpretability. *E.g.* for explainers for VGG networks, more than 80% signals came from interpretable filters. The explainer can also invert object-part feature maps back to reconstruct feature maps of the performer without losing much information.

References

[1] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to learn by gradient descent by gradient descent. *In NIPS*, 2016.

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

[3] 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.

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

[5] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas. Learning to learn without gradient descent by gradient descent. *In ICML*, 2017.

[6] Y. Dong, H. Su, J. Zhu, and F. Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *In arXiv:1708.05493*, 2017.

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

[8] E. R. Elenberg, A. G. Dimakis, M. Feldman, and A. Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. *In NIPS*, 2017.

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

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

[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.

[12] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. P. Xing. Harnessing deep neural networks with logic rules. *In arXiv:1603.06318v2*, 2016.

[13] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *In CVPR*, 2017.

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

[15] S. Kolouri, C. E. Martin, and H. Hoffmann. Explaining distributed neural activations via unsupervised learning. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017.

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

[17] D. Kumar, A. Wong, and G. 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.

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

[19] B. J. Lengerich, S. Konam, E. P. Xing, S. Rosenthal, and M. Veloso. Visual explanations for convolutional neural networks via input resampling. *In ICML Workshop on Visualization for Deep Learning*, 2017.

[20] K. Li and J. Malik. Learning to optimize. *In arXiv:1606.01885*, 2016.

[21] R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. *In NIPS*, 2016.

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

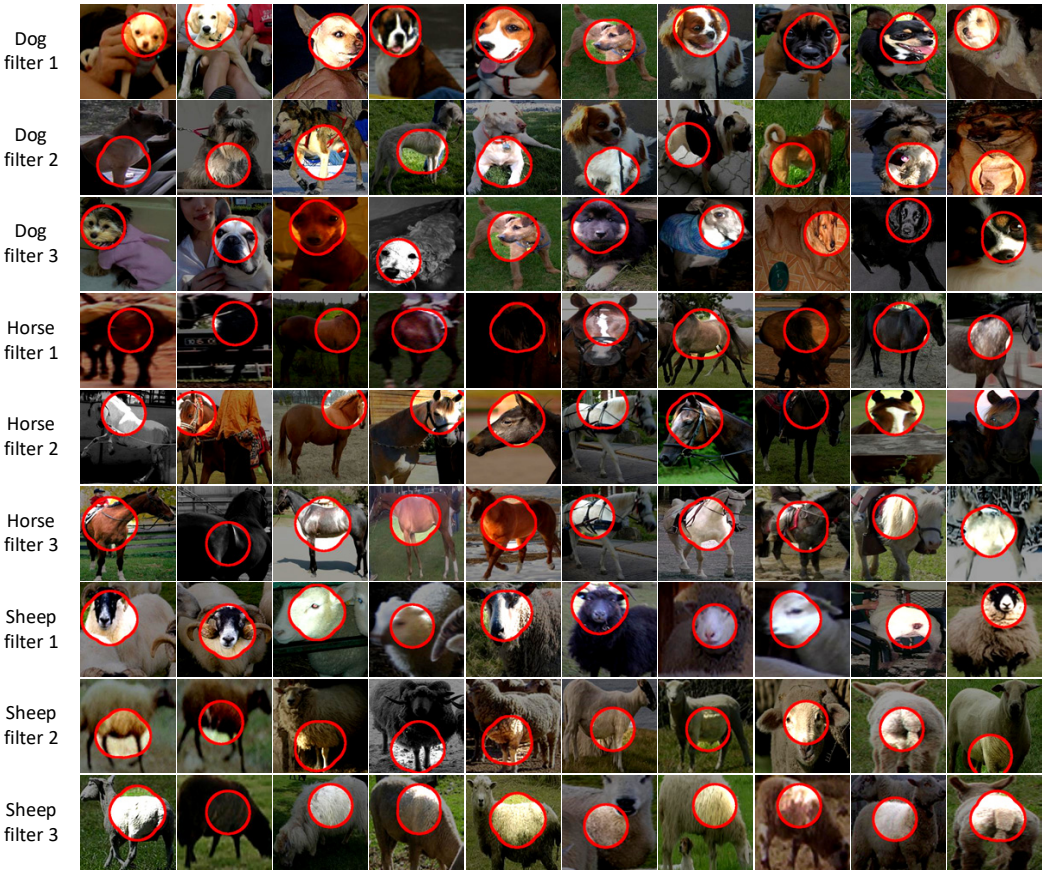
[23] P. E. Rauber, S. G. Fadel, A. X. F. ao, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *In Transactions on PAMI*, 23(1):101–110, 2016.

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

[25] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. *In NIPS*, 2017.
[26] R. Schwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *In arXiv:1703.00810*, 2017.
[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In ICCV*, 2017.
[28] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *In ICCV*, 2015.
[29] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: visualising image classification models and saliency maps. *In arXiv:1312.6034*, 2013.
[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.
[31] A. Stone, H. Wang, Y. Liu, D. S. Phoenix, and D. George. Teaching compositionality to cnns. *In CVPR*, 2017.
[32] C. Ventura, D. Masip, and A. Lapedriza. Interpreting cnn models for apparent personality trait regression. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017.
[33] 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.
[34] J. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *In arXiv:1611.05763v3*, 2017.
[35] A. S. Wicaksana and C. C. S. Liem. Human-explainable features for job candidate screening prediction. *In CVPR Workshop on Explainable Computer Vision and Job Candidate Screening Competition*, 2017.
[36] N. Wolchover. New theory cracks open the black box of deep learning. *In Quanta Magazine*, 2017.
[37] T. Wu, X. Li, X. Song, W. Sun, L. Dong, and B. Li. Interpretable r-cnn. *In arXiv:1711.05226*, 2017.
[38] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *In ICML Deep Learning Workshop*, 2015.
[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *In ECCV*, 2014.
[40] Q. Zhang, R. Cao, F. Shi, Y. Wu, and S.-C. Zhu. Interpreting cnn knowledge via an explanatory graph. *In AAAI*, 2018.
[41] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Growing interpretable part graphs on convnets via multi-shot learning. *In AAAI*, 2016.
[42] Q. Zhang, R. Cao, Y. N. Wu, and S.-C. Zhu. Mining object parts from cnns via active question-answering. *In CVPR*, 2017.
[43] Q. Zhang, W. Wang, and S.-C. Zhu. Examining cnn representations with respect to dataset bias. *In AAAI*, 2018.
[44] Q. Zhang, Y. N. Wu, and S.-C. Zhu. Interpretable convolutional neural networks. *In CVPR*, 2018.
[45] Q. Zhang, Y. Yang, Y. N. Wu, and S.-C. Zhu. Interpreting cnns via decision trees. *In arXiv:1802.00121*, 2018.
[46] Q. Zhang and S.-C. Zhu. Visual interpretability for deep learning: a survey. *in Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.
[47] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *In ICRL*, 2015.
[48] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016.

Appendix: Visualization of feature maps of the explainer and feature maps of the performer

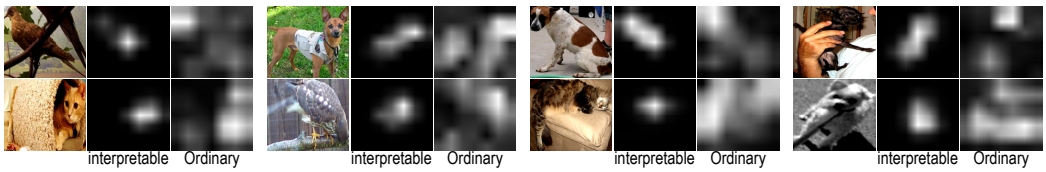
Visualization of feature maps in the *conv-interp-2* layer of the explainer. Each row corresponds to feature maps of a filter in the *conv-interp-2* layer. We simply used a round RF for each neural activation and overlapped all RFs for visualization.



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.

Appendix: Grad-CAM attention maps



Grad-CAM attention maps. We used [27] 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 5 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.

Appendix: Detailed results of p values

Table 4: p values of explainers.

	Pascal-Part dataset [4]								CUB200-2011 [33]
	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

Appendix: More results of location instability

In this section, we add another baseline for comparison. Because we considered 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.

Table 5: Location instability of feature maps in performers and explainers. Performers are learned based on the Pascal-Part dataset [4]

	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**

Appendix: Understanding of filter losses

We can re-write the filter loss as

$$\textbf{Loss}_f = -H(\mathbf{T}) + H(\mathbf{T}'|\mathbf{X}) + \sum\nolimits_{x_f \in \mathbf{X}} p(\mathbf{T}^+, x_f) H(\mathbf{T}^+|X = x_f)$$

where $\mathbf{T}' = \{T^-, \mathbf{T}^+\}$. $H(\mathbf{T}) = -\sum_{T \in \mathbf{T}} p(T) \log p(T)$ is a constant prior entropy of part templates.

Low inter-category entropy: The second term $H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X})$ is computed as $H(\mathbf{T}' = \{T^-, \mathbf{T}^+\}|\mathbf{X}) = -\sum_{x_f} p(x_f) \sum_{T \in \{T^-, \mathbf{T}^+\}} p(T|x_f) \log p(T|x_f)$, where $\mathbf{T}^+ =$

Table 6: Location instability of feature maps in performers and explainers. Performers are learned based on the CUB200-2011 dataset [33]

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**

$\{T_{\mu_1}, T_{\mu_2}, \ldots, T_{\mu_{L^2}}\} \subset \mathbf{T}$ and $p(\mathbf{T}^+|x_f) = \sum_\mu p(T_\mu|x_f)$. We define the set of all positive templates $\mathbf{T}^+$ as a single label to represent category c. We use the negative template T^- 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. We can use a feature map x_f of f to identify whether the input image belongs to category c or not, *i.e.* x_f fitting to either $T_{\hat{\mu}}$ or T^-, without great uncertainty.

Low spatial entropy: The third term is given as $H(\mathbf{T}^+|X = x_f) = \sum_\mu \tilde{p}(T_\mu|x_f) \log \tilde{p}(T_\mu|x_f)$, where $\tilde{p}(T_\mu|x_f) = \frac{p(T_\mu|x_f)}{p(\mathbf{T}^+|x_f)}$. 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 $\hat{\mu}$ 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. [44] computes an approximate but efficient gradients to speed up the computation, as follows.

$$\frac{\partial \mathbf{Loss}_f}{\partial x_f^{(ij)}} = \sum_T \frac{p(T) t_{ij} e^{tr(x_f \cdot T)}}{Z_T} \Big\{ tr(x_f \cdot T) - \log\left[Z_T p(x_f)\right] \Big\} \approx \frac{p(\hat{T}) \hat{t}_{ij}}{Z_{\hat{T}}} e^{tr(x_f \cdot \hat{T})} \Big\{ tr(x_f \cdot \hat{T}) - \log[Z_{\hat{T}} p(x_f)] \Big\}$$

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 target category of filter f, 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 $\hat{T} = T^-$. Considering $\forall T \in \mathbf{T} \setminus \{\hat{T}\}$, $e^{tr(x_f \cdot \hat{T})} \gg e^{tr(x_f \cdot T)}$ after initial learning episodes, we can make approximations in the above equation.

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

Inspired by optimization tricks in [44], 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.

Appendix: Evaluating the reconstruction quality based on the object-classification accuracy

Table 7: Classification errors based on feature maps of performers and explainers.

	Pascal-Part [4]						CUB200 [33]		
	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%

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.