Interpretable Neural Network Decoupling

Yuchao Li[1], Rongrong Ji[1,2⋆] , Shaohui Lin[3], Baochang Zhang[4], Chenqian Yan[1], Yongjian Wu[5], Feiyue Huang[5], Ling Shao[6,7]

[1]Department of Artificial Intelligence, School of Informatics, Xiamen University, China, [2]Peng Cheng Laboratory, Shenzhen, China, [3]National University of Singapore, Singapore, [4]Beihang University, China, [5]BestImage, Tencent Technology (Shanghai) Co.,Ltd, China, [6]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE, [7]Inception Institute of Artificial Intelligence, Abu Dhabi, UAE

Abstract. The remarkable performance of convolutional neural networks (CNNs) is entangled with their huge number of uninterpretable parameters, which has become the bottleneck limiting the exploitation of their full potential. Towards network interpretation, previous endeavors mainly resort to the single filter analysis, which however ignores the relationship between filters. In this paper, we propose a novel architecture decoupling method to interpret the network from a perspective of investigating its calculation paths. More specifically, we introduce a novel architecture controlling module in each layer to encode the network architecture by a vector. By maximizing the mutual information between the vectors and input images, the module is trained to select specific filters to distill a unique calculation path for each input. Furthermore, to improve the interpretability and compactness of the decoupled network, the output of each layer is encoded to align the architecture encoding vector with the constraint of sparsity regularization. Unlike conventional pixel-level or filter-level network interpretation methods, we propose a path-level analysis to explore the relationship between the combination of filter and semantic concepts, which is more suitable to interpret the working rationale of the decoupled network. Extensive experiments show that the decoupled network achieves several applications, i.e., network interpretation, network acceleration, and adversarial samples detection.

Keywords: Network Interpretation, Architecture Decoupling

1 Introduction

Deep convolutional neural networks (CNNs) have dominated various computer vision tasks, such as object classification, detection and semantic segmentation. However, the superior performance of CNNs is rooted in their complex architectures and huge amounts of parameter, which thereby restrict the interpretation of their internal working mechanisms. Such a contradiction has become a key drawback when the network is used in task-critical applications such as medical diagnosis, automatic robots, and self-driving cars.

⋆ Corresponding author.

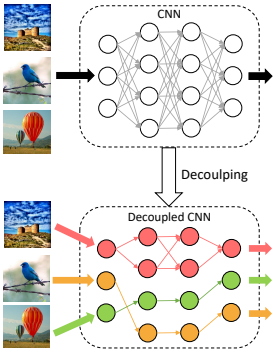


Fig. 1. An example of the neural network architecture decoupling. Each color represents a calculation path of specific input.

To this end, network interpretation have been explored to improve the understanding of the intrinsic structures and working mechanisms of neural networks [40, 2, 28, 41, 20, 26, 5]. Interpreting a neural network involves investigating the rationale behind the decision-making process and the roles of its parameters. For instance, some methods [22, 5] view networks as a whole when explaining their working process. However, these approaches are too coarse-grained for exploring the intrinsic properties in the networks. In contrast, network visualization approaches [40, 39] interpret the role of each parameter by analyzing the pixel-level feature representation, which always require complex trial-and-error experiments. Beyonds, Bau *et al.* [2] and Zhang *et al.* [41] explored the different roles of filters in the decision-making process of a network. Although these methods are more suitable for explaining the network, they characterize semantic concepts using only a single filter, which has been proven to be less effective than using a combination of multiple filters [34, 10]. Under this situation, different combination of filters can be viewed as different calculation paths in the network, which inspires us to investigate the working process of networks based on a path-level analysis. The challenge, however, comes from the fact that each inference involves all filters in the network and has the same calculation process, making it difficult to interpret how each calculation path affects the final result. To overcome this problem, previous methods [36, 37] explore the difference between the calculation paths of different inputs by reducing the number of parameters involved in the calculation process. For instance, Wang *et al.* [36] proposed a post-hoc analysis to obtain a unique calculation path of a specific input based on a pre-trained model, which however involves a huge number of complicated experiments. Moreover, Sun *et al.* [37] learned a network that generates a dynamic calculation path in the last layer by modifying the SGD algorithm. However, it ignores the fact that the responses of filters are also dynamic in the intermediate layers, and thus cannot interpret how the entire network works.

In this paper, we propose an interpretable network decoupling approach, which enables a network to adaptively select a suitable subset of filters to form a calculation path for each input, as shown in Fig. 1. In particular, Our design principle lies in a novel light-weight *architecture controlling module* as well as a novel learning process for network decoupling. Fig. 2 depicts the framework of the proposed method. The architecture controlling module is first incorporated into each layer to dynamically select filters during network inference with a negligible computational burden. Then, we maximize the mutual information between the architecture encoding vector (*i.e.,* the output of the architecture controlling module) and the inherent attributes of the input images during training, which

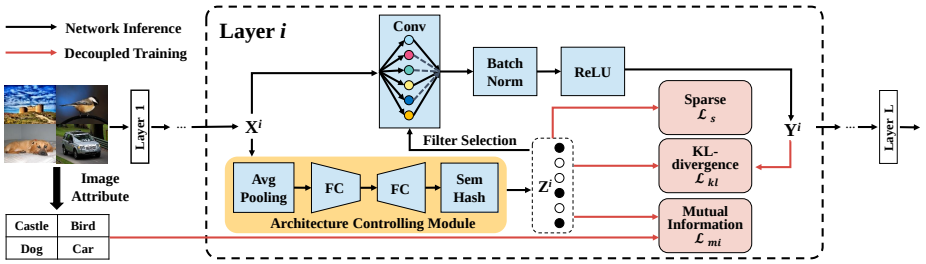


Fig. 2. The framework of the proposed interpretable neural network decoupling. The architecture encoding vector $\mathbf{z}^i$ is first constructed by the architecture controlling module, and then learned to determine the filter selection by Eq. 12. For network inference, we only use the selected filters based on each input. The mutual information loss $\mathcal{L}_{mi}$ is computed between the output of the architecture controlling module $\mathbf{z}^i$ and the attribute of the inputs to decouple the network architecture. The KL-divergence loss $\mathcal{L}_{kl}$ is computed by the output of convolutional layer $\mathcal{Y}^i$ and $\mathbf{z}^i$ to disentangle the filters. The sparse loss $\mathcal{L}_s$ is used to sparsify the result of filter selection.

allows the network to dynamically generate the calculation path related to the input. In addition, to further improve the interpretability of decoupled networks, we increase the similarity between the architecture encoding vector of each convolutional layer and its output by minimizing the KL-divergence between them, making filter only respond to a specific object. Finally, we sparsify the architecture encoding vector to attenuate the calculation path and eliminate the effects of redundant filters for each input. We also introduce an improved semantic hashing scheme to make the discrete architecture encoding vector differentiable, which is therefore capable to be trained directly by stochastic gradient descent (SGD).

Correspondingly, the decoupled network becomes more interpretable, and one can trace the functional processing behavior layer-by-layer to form a hierarchical path towards understanding the working principle of the decoupled network. Meanwhile, each filter is only related to a set of similar input images after the decoupling, thus they also become more interpretable, and the combination of them forms a decoupled sub-architecture, which better characterizes the specific semantic concepts. Such a decoupled architecture further benefits from a low computational cost for network acceleration, as well as good hints for adversarial samples detection, which are subsequently validated in our experiments.

We summarize our three main contributions as follows:

- To interpret neural networks by dynamically selecting the filters for different inputs, we propose a lightweight architecture controlling module, which is differentiable and can be optimized by SGD based on the losses we propose.
- The decoupled network reserves similar performance of the original network and has better interpretable. Thus it enables the functional processing of each calculation path to be well interpreted, which helps better understand

the rationale behind the network inference, as well as explore the relationship between filters and semantic concepts in the decoupled network.
– Our method is generic and flexible, which can be easily employed on the existing network architectures, such as VGGNets [32], ResNets [13], and Inceptions [33]. The decoupled architecture further benefits extensive applications, including network acceleration and adversarial samples detection.

2 Related Work

Network Interpretation. One way to interpret a network is to analyze how it responds to a specific input image for output prediction [20, 26, 22, 42, 5]. This strategy views the network as a whole to interpret the network prediction results by exploring the knowledge blind spots of neural networks [22], or by assigning each output feature an importance value for a particular prediction [26]. Moreover, a decision tree [42] or an explainer network [5] has been used to better understand the classification process. However, these methods only pay attention to the reason behind the network prediction result, and the roles of each parameter are ignored, making it difficult to understand their effects on the network.

To open the black-box of neural network and interpret the role of parameters, several methods [40, 39, 8] have been proposed to visualize the feature representations inside the network. For instance, Zeiler *et al.* [40] visualized the feature maps in the intermediate layers by establishing a deconvolutional network corresponding to the original one. Yoshinski *et al.* [39] proposed two visualization methods to explore the information contained in features: a respective post-hoc analysis on a pre-trained model and learning a network by regularized optimization. Visualizing feature representations is a very direct method to explain the role of parameters in a network, which however requires extensive experiments due to the enormous number of parameters.

In addition to the above methods, the functions of filters are also explored for interpreting networks [38, 2, 28, 41, 37]. They have evaluated the transferability of filters [38] or quantified the relationship between filters and categories [28] to explain their different roles. Compared with using a single filter to represent semantic concepts, methods in [34, 10] have found that the semantic concepts can be better characterized by combining multiple filters. Wang *et al.* [34] further validated that clustering the activations of multiple filters can better represent semantic concepts than using a single filter. Fong *et al.* [10] mapped the semantic concepts into vectorial embeddings based on the responses of multiple filters and found that these embeddings can better characterize the features. Different from these methods, we interpret the working principle of a network based on a path-level analysis by decoupling the network, upon which we further disentangle each intra-layer filter to explore the interpretable semantic concepts across filters on the calculation path. Our method is more in line with the internal working mechanism of the network than these works, and has a better extension to other applications, such as network acceleration and adversarial samples detection.

Conditional Computation. Works on conditional computation tend to concentrate on the selection of model components when generating the calculation path. For instance, the work in [3] explored the influence of stochastic or non-smooth neurons when estimating the gradient of the loss function. Later, an expert network was learned to find a suitable calculation path for each input by reinforcement learning [4] or SGD [6]. However, the requirement of a specific expert network makes these approaches cumbersome. Along another line, a halting score [9] or a differentiable directed acyclic graph [24] has been used to dynamically adjust the model components involved in the calculation process. Recently, a feature boosting and suppression method [11] was introduced to skip unimportant output channels of the convolutional layer for data-dependent inference, which is different from static pruning [**?**,**?**,**?**]. However, it selects the same number of filters for each layer, without considering inter-layer differences. Different from the above works, we employ a novel architecture controlling module to decouple the network by fitting it to the data distribution. After decoupling, the network becomes interpretable, enabling us to visualize its intrinsic structure, accelerate the inference, and detect adversarial samples.

3 Architecture Decoupling

Formally speaking, the l-th convolutional layer in a network with a batch normalization (BN) [17] and a ReLU layer [29] transforms $\mathcal{X}^l \in \mathbb{R}^{C^l \times H^l_{in} \times W^l_{in}}$ to $\mathcal{Y}^l \in \mathbb{R}^{N^l \times H^l_{out} \times W^l_{out}}$ using the weight $\mathcal{W}^l \in \mathbb{R}^{N^l \times C^l \times D^l \times D^l}$, which is defined as:

$$\mathcal{Y}^l = \Big(BN\big(Conv(\mathcal{X}^l, \mathcal{W}^l)\big) \Big)_+, \tag{1}$$

where $(\cdot)_+$ represents the ReLU layer, and $\mathrm{Conv}(\cdot,\cdot)$ denotes the standard convolution operator. (H^l_{in}, W^l_{in}) and (H^l_{out}, W^l_{out}) are the spatial size of the input and output in the l-th layer, respectively. D^l is the kernel size.

3.1 Architecture Controlling Module

For an input image, the proposed architecture controlling module selects the filters and generates the calculation path during network inference. In particular, we aim to predict which filters need to participate in the convolutional computation *before* the convolutional operation to accelerate network inference. Therefore, for the l-th convolutional layer, the architecture encoding vector $\mathbf{z}^l$ (*i.e.,* the output of the architecture controlling module) only relies on the input $\mathcal{X}^l$ instead of the output $\mathcal{Y}^l$, which is defined as $\mathbf{z}^l = G^l(\mathcal{X}^l)$. Inspired by the effectiveness of the squeeze-and-excitation (SE) block [16], we select a similar SE-block to predict the importance of each filter. Thus, we first squeeze the global spatial information via global average pooling, which transforms each input channel $X^l_i \in \mathbb{R}^{H^l_{in} \times W^l_{in}}$ to a scalar s^l_i. We then design a sub-network structure $\bar{G}^l(\mathbf{s}^l)$ to determine the filter selection based on $\mathbf{s}^l \in \mathbb{R}^{C^l}$, which is

formed by two fully connected layers, *i.e.,* a dimensionality-reduction layer with weights $\mathbf{W}_1^l$ and a dimensionality-increasing layer with weights $\mathbf{W}_2^l$:

$$\bar{G}^l(\mathbf{s}^l) = \mathbf{W}_2^l \cdot (\mathbf{W}_1^l \cdot \mathbf{s}^l)_+, \tag{2}$$

where $\mathbf{W}_1^l \in \mathbb{R}^{\frac{C^l}{\gamma} \times C^l}$, $\mathbf{W}_2^l \in \mathbb{R}^{N^l \times \frac{C^l}{\gamma}}$ and $\cdot$ represents the matrix multiplication. We ignore the bias for simplicity. To reduce the module complexity, we empirically set the reduction ratio γ to 4 in our experiments. The output of $\bar{G}^l(\mathbf{s}^l)$ is a real vector, while we need to binarize $\bar{G}^l(\mathbf{s}^l)$ to construct a binary vector $\mathbf{z}^l$, which represents the result of filter selection. However, a simple discretization using the sign function is not differentiable, which prevents the corresponding gradients from being directly obtained by back-propagation. Thus, we further employ an *Improved SemHash* method [19] to transform the real vector in $\bar{G}^l(\mathbf{s}^l)$ to a binary vector by a simple rounding bottleneck, which also makes the discretization become differentiable.

Improved SemHash. The proposed scheme is based on the different operations for training and testing. During training, we first sample a noise $\alpha \sim \mathcal{N}(0,1)^{N^l}$, which is added to $\bar{G}^l(\mathbf{s}^l)$, and then obtain $\widetilde{\mathbf{s}}^l = \bar{G}^l(\mathbf{s}^l) + \alpha$. After that, we compute a real vector and a binary vector by:

$$\mathbf{v}_1^l = \sigma'(\widetilde{\mathbf{s}}^l), \mathbf{v}_2^l = \mathbf{1}(\widetilde{\mathbf{s}}^l > 0), \tag{3}$$

where σ' is a saturating Sigmoid function [18] denoted as:

$$\sigma'(x) = \max\Big(0, \min\big(1, 1.2\sigma(x) - 0.1\big)\Big). \tag{4}$$

Here, σ is the Sigmoid function. $\mathbf{v}_1^l \in \mathbb{R}^{C^l}$ is a real vector with all elements falling in the interval $[0,1]$, and we calculate its gradient during back-propagation. $\mathbf{v}_2^l \in \mathbb{R}^{C^l}$ represents the discretized vector, which cannot be involved in the gradient calculation. Thus, we randomly use $\mathbf{z}^l = \mathbf{v}_1^l$ for half of the training samples and $\mathbf{z}^l = \mathbf{v}_2^l$ for the rest in the forward-propagation. We then mask the output channels using the architecture encoding vector (*i.e.,* $\mathcal{Y}^l * \mathbf{z}^l$) as the final output of this layer. In the backward-propagation, the gradient of $\mathbf{z}^l$ is the same as the gradient of $\mathbf{v}_1^l$.

During evaluation/testing, we directly use the sign function in the forward-propagation as:

$$\mathbf{z}^l = \mathbf{1}\big(\bar{G}^l(\mathbf{s}^l) > 0\big). \tag{5}$$

After that, we select suitable filters involved in the convolutional computation based on $\mathbf{z}^l$ to achieve fast inference.

3.2 Network Training

We expect the network architecture to be gradually decoupled during training, where the essential problem is how to learn an architecture encoding vector that fits the data distribution. To this end, we propose three loss functions for network decoupling.

Mutual Information Loss. When the network architecture is decoupled, different inputs should select their related sets of filters. We adopt mutual information $I(a;\mathbf{z}^l)$ between the result of filter selection $\mathbf{z}^l$ and the attribute of an input image a (*i.e.,* the unique information contained in the input image) to measure the correlation between the architecture encoding vector and its input image. $I(a;\mathbf{z}^l)=0$ means that the result of filter selection is independent to the input image, *i.e.,* all the inputs share the same filter selection. In contrast, when $I(a;\mathbf{z}^l)\neq 0$, filter selection depends on the input image. Thus, we maximize the mutual information between a and $\mathbf{z}^l$ to achieve architecture decoupling. Formally speaking, we have:

$$\begin{aligned} I(a;\mathbf{z}^l) &= H(a)-H(a|\mathbf{z}^l) \\ &= \sum_a\sum_{\mathbf{z}^l} P(a,\mathbf{z}^l)logP(a|\mathbf{z}^l)+H(a) \\ &= \sum_a\sum_{\mathbf{z}^l} P(\mathbf{z}^l)P(a|\mathbf{z}^l)logP(a|\mathbf{z}^l)+H(a). \end{aligned} \tag{6}$$

The mutual information $I(a;\mathbf{z}^l)$ is difficult to directly maximize, as it is hard to obtain $P(a|\mathbf{z}^l)$. Thus, we use $Q(a|\mathbf{z}^l)$ as a variational approximation to $P(a|\mathbf{z}^l)$ [1]. In fact, the KL-divergence is positive, so we have:

$$\begin{aligned} KL\big(P(a|\mathbf{z}^l),Q(a|\mathbf{z}^l)\big)\geq 0 \Rightarrow &\sum_a P(a|\mathbf{z}^l)logP(a|\mathbf{z}^l) \\ &\geq \sum_a P(a|\mathbf{z}^l)logQ(a|\mathbf{z}^l). \end{aligned} \tag{7}$$

We then obtain the following equation:

$$\begin{aligned} I(a;\mathbf{z}^l) &\geq \sum_a\sum_{\mathbf{z}^l} P(\mathbf{z}^l)P(a|\mathbf{z}^l)logQ(a|\mathbf{z}^l)+H(a) \\ &\geq \sum_a\sum_{\mathbf{z}^l} P(\mathbf{z}^l)P(a|\mathbf{z}^l)logQ(a|\mathbf{z}^l) \\ &= \mathbb{E}_{\mathbf{z}^l\sim G^l(\mathcal{X}^l)}[\mathbb{E}_{a\sim P(a|\mathbf{z}^l)}[logQ(a|\mathbf{z}^l)]]. \end{aligned} \tag{8}$$

Eq. 8 provides a lower bound for the mutual information $I(a;|\mathbf{z}^l)$. By maximizing this bound, the mutual information $I(a;\mathbf{z}^l)$ will also be maximized accordingly. In our paper, we use the class label as the attribute of the input image c in the classification task. Moreover, we reparametrize $Q(a|\mathbf{z}^l)$ as a neural network $\tilde{Q}(\mathbf{z}^l)$ that contains a fully connected layer and a softmax layer. Thus, maximizing the mutual information in Eq. 8 is achieved by minimizing the following loss:

$$\mathcal{L}_{mi}=-\sum_{l=1}^{L} A_X * log\tilde{Q}(\mathbf{z}^l), \tag{9}$$

where A_X represents the label of the input image X. $\tilde{Q}(\mathbf{z}^l)$ is defined as $\mathbf{W}^l_{cla}\cdot\mathbf{z}^l$ with a fully connected weight $\mathbf{W}^l_{cla}\in\mathbb{R}^{K\times N^l}$, where K represents the number of categories in image classification.

KL-divergence Loss. After decoupling the network architecture, we guarantee that the filter selection depends on the input image. However, it is uncertain whether the filters become different (*i.e.,* detect different objects), which obstructs us from further interpreting the network. If a filter only responds to a specific semantic concept, it will not be activated when the input does not contain this feature. Thus, by limiting filters to only respond to specific category, they can be disentangled to detect different categories. To achieve this goal, we minimize the KL-divergence between the output of the current layer and its corresponding architecture encoding vector, which ensures that the overall responses of filters have a similar distribution to the responses of the selected subset. To align the dimension of the convolution output and architecture encoding vector, we further downsample $\mathcal{Y}^l$ to $\mathbf{y}^l \in \mathbb{R}^N$ using global average pooling. Then, the KL-divergence loss is defined as:

$$\mathcal{L}_{kl} = \sum_{l=1}^{L} KL(\mathbf{z}^l || \mathbf{y}^l). \tag{10}$$

As the output of filter is limited by the result of filter selection, it will be unique and only detects the specific object. Thus, all filters are different from each other, *i.e.,* each one performs its function.

Sparse Loss. An ℓ_1-regularization on $\mathbf{z}^l$ is further introduced to encourage the architecture encoding vector to be sparse, which makes the calculation path of each input becomes thinner. Thus, the sparse loss is defined as:

$$\mathcal{L}_s = \sum_{l=1}^{L} |\|\mathbf{z}^l\|_1 - R * N^l|, \tag{11}$$

where R represents the target compression ratio. Since z^l falls in the interval $[0, 1]$, the maximum value of $\|\mathbf{z}^l\|_1$ is N^l, and the minimum value is 0, where N^l is the number of filters. For example, we set R to 0.5 if activating only half of the filters.

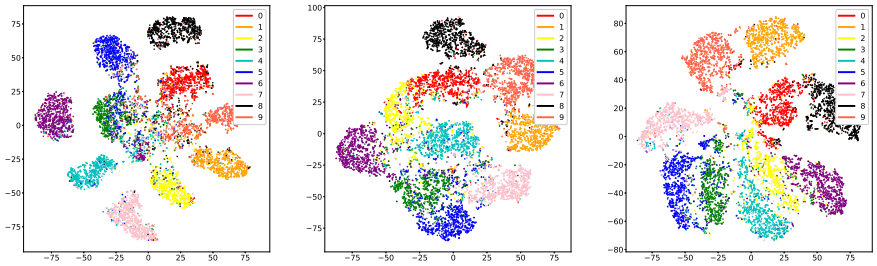
Therefore, we obtain the overall loss function as follows:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_m * \mathcal{L}_{mi} + \lambda_k * \mathcal{L}_{kl} + \lambda_s * \mathcal{L}_s, \tag{12}$$

where $\mathcal{L}_{ce}$ is the network classification loss. λ_m, λ_k and λ_s are the hyper-parameters. Eq. 12 can be effectively solved via SGD.

4 Experiments

We evaluate the effectiveness of the proposed neural network architecture decoupling scheme on three kinds of networks, *i.e.*, VGGNets [32], ResNets [13], and Inceptions [33]. For network acceleration, we conduct comprehensive experiments on three datasets, *i.e.*, CIFAR-10, CIFAR-100 [21] and ImageNet 2012 [31]. For quantifying the network interpretability, we use the interpretability of filters [41] and the representation ability of semantic features [10] on BRODEN dataset [2] to evaluate the original and our decoupled models.



(a) VGGNet (b) ResNet-56 (c) GoogleNet

Fig. 3. Visualization of the distribution of the integral calculation path in different networks on CIFAR-10.

4.1 Implementation Details

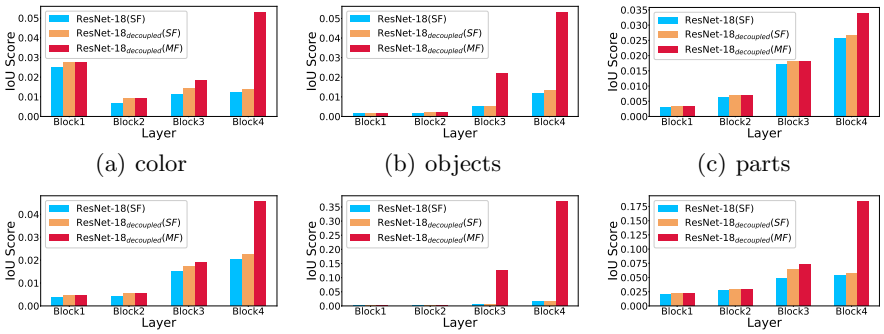
We implement our method using PyTorch [30]. The weights of decoupled networks are initialized using the weights from their corresponding pre-trained models. We add the architecture controlling module to all convolutional layers except the first and last ones. All networks are trained using stochastic gradient descent with a momentum of 0.9. For CIFAR-10 and CIFAR-100, we train all the networks over 200 epochs using a mini-batch size of 128. The learning rate is initialized by 0.1, which is divided by 10 at 50% and 75% of the total number of epochs. For ImageNet 2012, we train the networks over 120 epochs with a mini-batch size of 64 and 256 for VGG-16 and ResNet-18, respectively. The learning rate is initialized as 0.01 and is multiplied by 0.1 after the 30-th, 60-th and 90-th epoch. The real speed on the CPU is measured by a single-thread AMD Ryzen Threadripper 1900X. Except for the experiments on network acceleration, we automatically learn sparse filters by setting R to 0 in Eq. 11.

4.2 Network Interpretability

Architecture Encoding. We collect the calculation paths from three different networks (*i.e.,* VGGNet, ResNet-56 and GoogleNet) to verify that the proposed network decoupling method can successfully decouple the network and ensure that it generates different calculation paths for different images. We first reduce the dimension of the calculation path (*i.e.,* the concatenation of architecture encoding vectors $\mathbf{z}^l$ across all layers) to 300 using Principal Component Analysis (PCA), and then visualize the calculation path by t-SNE [27]. As shown in Fig. 3, each color represents one category and each dot is a calculation path corresponding to an input. We can see that the network architecture is successfully decoupled after training by our method, where different categories of images have different calculation paths.

Model	Top1-Acc	Top5-Acc	Conv2_2	Conv3_3	Conv4_3	Conv5_3
VGG-16	71.59	90.38	0.0637	0.0446	0.0627	0.0787
VGG-16$_{decoupled}$	71.51	90.32	**0.0750**	**0.0669**	**0.0643**	**0.0879**
Model	Top1-Acc	Top5-Acc	Block1	Block2	Block3	Block4
ResNet-18	69.76	89.08	0.0527	0.0212	0.0477	0.0521
ResNet-18$_{decoupled}$	67.62	87.78	**0.1062**	**0.0268**	**0.0580**	**0.0618**

Table 1. The average interpretability score of filters in the different layers of original networks and decoupled networks on BRODEN. The higher score is better.



(a) color (b) objects (c) parts

(d) materials (e) scenes (f) textures

Fig. 5. Average representation ability of different concepts in ResNet-18 on BRODEN. SF/MF represents use single/multiple filters characterizing the semantic features.

Filter State. After decoupling the network architecture, the state of a filter in the network has three possibilities: it responds to all the input samples, it does not respond to any input samples, or it responds to the specific inputs. These three possibilities are termed as *energetic filter*, *silent filter*, and *dynamic filter*, respectively. As shown in Fig. 4, we collect different states of filters in different layers. We can see that the proportion of dynamic filters increases with network depth increasing. This phenomenon demonstrates that filters in the top layer tend to detect high-level semantic features, which are highly related to the input images. In contrast, filters in the bottom layer tend to detect low-level features, which are always shared across images. For more detailed analysis, refer to Section A.1 of the supplementary material.

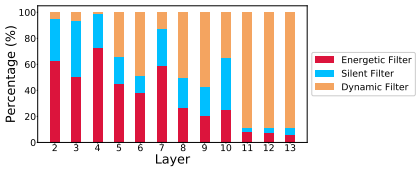


Fig. 4. The distribution of filters with different states in each layer of VGG-16 on ImageNet2012.

Interpretable Quantitative Analysis. Following the works [2, 41, 10], we select the interpretability of filters and the representation ability of semantic features to measure the network interpretability. Specifically, we first select the original and our decoupled models which trained on ImageNet2012, and compute

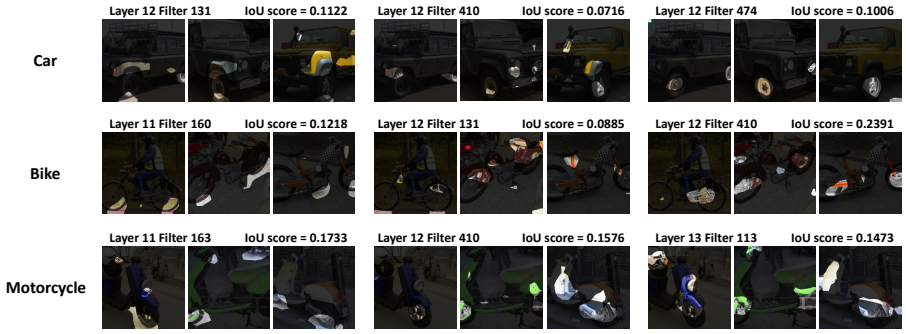


Fig. 6. Visualization of the receptive fields of filters which are inactivated because of the lack of semantic feature in images. We occlude the specific semantic feature (*i.e.,* wheel) in different images (*i.e.,* car, bike and motorcycle) on ImageNet and then collect the filters become inactivated due to the lack of the semantic feature.

the activation map of each filter/unit on BRODEN dataset. Then, the top quantile level threshold is determined over all spatial locations of feature maps. After that, low-resolution activation maps of all filters are scaled up to input-image resolution using bilinear interpolation and thresholded into a binary segmentation, so as to obtain the receptive fields of filters. The score of each filter f as segmentation for the semantic concept t in the input image I is reported as an intersection-over-union score $IoU^I_{f,t} = \frac{|S^I_f \cap S^I_t|}{|S^I_f \cup S^I_t|}$, where S^I_f and S^I_t denote the receptive field of filter f and the ground-truth mask of the semantic concept t in the input image, respectively. Given an image I, we associated filter f with the t-th part if $IoU^I_{f,t} > 0.01$. Finally, we measure the relationship between the filter f and concept t by $P_{f,t} = mean_I \mathbf{1}(IoU^I_{f,t} > 0.01)$ across all the input images. Based on [41], we can report the highest association between the filter and concept as the final interpretability score of filter f by $max_t P_{f,t}$. As shown in Table. 1, the value in each layer is obtained by averaging the final interpretability score across all the corresponding filters. For ResNet-18, we collect the filters from the first convolutional layers in the last unit of each block. Compared to the original networks, our decoupled networks have the better interpretability under the similar classification accuracy. For instance, we achieve $1.2\times \sim 2\times$ score improvement of the filter interpretability than the original ResNet-18.

We further investigate the representation ability of network for specific semantic features before and after network decoupling. For the representation of semantic features from a single filter, we evaluate the highest association between each semantic feature in BRODEN (which has $1,197$ semantic features) and the filters using $max_{I,f} IoU^I_{f,t}$ as the representation ability of specific semantic features, based on [10]. For the representation of semantic features from multiple filters, we first occlude the semantic features in the original image and then collect the number of M filters by comparing the difference between the

calculation path of the original image and the occluded image, where these filters are activated on the original image but inactivated due to the lack of specific semantic features. After that, we merge their receptive field and calculate the value of IoU $IoU^I_{f\in M,t} = \frac{|S^I_{f\in M} \cap S^I_t|}{|S^I_{f\in M} \cup S^I_t|}$ as the representation ability of semantic feature t. As shown in Fig. 5, we average the representation ability of semantic features belonging to the same concepts in the different layers. The results demonstrates that our decoupled network has the better representation ability of semantic feature than the original ResNet-18. The combination of multiple filters, which collected by our path-level disentangling, achieves about $3\times$ improvement in the representation ability than the single ones. Moreover, we find that the bottom layers in the decoupled network always use the single filters to characterize the semantic features based on our path-level analysis, so the representation ability of semantic features in the bottom layers is similar in the single filter and multiple filters.

Semantic Concept Analysis. We further investigate the relationship between semantic concepts and calculation paths. To this end, we occlude the areas that contain similar semantic features (*i.e.,* wheels) in the images from different categories (*i.e,.* car, bike and motorcycle) to analyze the characterization of the same semantic concept in different categories. After that, we collect the filters which in the different parts of calculation path between the original images and the semantic lacked images. Our experiments only collect the three filters with highest IoU score in the last three convolutional layers of VGG-16. We find that the existence of a single semantic concept affects the state of multiple filters. For example, as shown in the first row of Fig. 6, when we only occlude the wheels of the car with black blocks, the 131-th, 410-th and 474-th filters in the 12-th convolutional layer become inactived, which makes the calculation path change. To further analyze the relationship between each filter and semantic concept, we visualize the receptive fields of filters on the input image to obtain the specific detection location of each one, and calculate the IoU score between the receptive fields of filters and the location area of the semantic concept. We find that different filters are responsible for different parts of the same semantic concept. For instance, the 131-th, 410-th and 474-th filters in the 12-th convolutional layer of VGG-16 are responsible for the features in the different parts of the wheel in "car" images, respectively. Therefore, the combination of these filters has the better representation ability of the wheel than the single ones.

4.3 Network Acceleration

In this subsection, we evaluate how our method can facilitate network acceleration. We decouple three different network architectures (*i.e.,* ResNet-56, VGGNet and GoogleNet) on CIFAR-10 and CIFAR-100, and set $R = 0$ to allow the networks to be learned automatically. The VGGNet in our experiments is the same as the network in [25]. As shown in Table 2, our method achieves the best trade-off between accuracy and speedup/compression rate, compared with static pruning [15, 23, 25] and dynamic pruning [35]. For instance, we achieve a

Model	CIFAR-10		CIFAR-100	
	FLOPs	Top-1 Acc(%)	FLOPs	Top-1 Acc(%)
ResNet-56	125M	93.17	125M	70.43
CP [15]	63M	91.80	-	-
L1 [23]*	90M	93.06	86M	69.38
Skip [35]*	103M	92.50	-	-
Ours	**63M**	**93.08**	**41M**	**69.72**
VGGNet	398M	93.75	398M	72.98
L1 [23]*	199M	93.69	194M	72.14
Slim [25]	196M	93.80	250M	73.48
Ours	**141M**	**93.82**	**191M**	**73.84**
GoogleNet	1.52B	95.11	1.52B	77.99
L1 [23]*	1.02B	94.54	0.87B	77.09
Ours	**0.39B**	**94.65**	**0.75B**	**77.28**

Table 2. Results of the different networks on CIFAR-10 and CIFAR-100. $*$ represents the result based on our implementation.

Model	Top-1 Acc↓ (%)	Top-5 Acc↓ (%)	FLOPs Reduction	CPU Time Reduction
SFP [14]	3.18	1.85	1.72×	1.38×
DCP [43]	2.29	1.38	1.89×	1.60×
LCL [7]	3.65	2.30	1.53×	1.25×
FBS [11]	2.54	1.46	1.98×	1.60×
Ours	**2.14**	**1.30**	**2.03×**	**1.64×**

Table 3. Results of ResNet-18 on ImageNet2012. The baseline in our method has an 69.76% top-1 accuracy and 89.08% top-5 accuracy with 1.81B FLOPs and an average 180 ms testing on CPU based an image by running the whole of the validation dataset.

2× FLOPs reduction with only a 0.09% drop in top-1 accuracy for ResNet-56 on CIFAR-10. For ImageNet 2012, the results of accelerating ResNet-18 are summarized in Table 3. When setting R to 0.6, our method also achieves the best performance with a 1.64× real CPU running speedup and 2.03× reduction in FLOPs compared with the static pruning [14, 43] and dynamic pruning [7, 11], while only decreasing by 1.30% in top-5 accuracy. The detail of hyper-parameter settings are presented in Section B of the supplementary material.

4.4 Adversarial Samples Detection

We further demonstrate that the proposed architecture decoupling can help to detect the adversarial samples. Recently, several works [12] have concluded that neural networks are vulnerable to adversarial examples, where adding a slight amount of noise to an input image can disturb their robustness. We add noise to images belonging to the "dog" category to make the network predicts as "truck" and visualize the distribution of the calculation path between the original images and adversarial samples in ResNet-56 on CIFAR-10, as shown in Fig. 7. The result demonstrates that the calculation path of the adversarial samples "dog→truck" is different from that of the original "dog" and "truck" images. In other words, adversarial samples do not completely deceive our decoupled network, which can detect them by analyzing their calculation paths. More examples are given in Section C.1 of the supplementary material.

Based on the above observation, we use random forest, adaboost and gradient boosting as the binary classifier to determine whether the calculation paths are from real or adversarial samples. As shown in Table 4, we randomly select 1, 5 and 10 images from each class in the ImageNet 2012 training set to organize

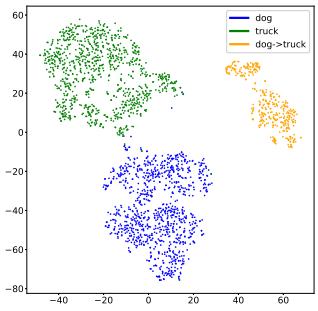


Fig. 7. The distribution of the integral calculation path of original images and adversarial samples in ResNet-56 on CIFAR-10.

Classifier	Method	Num. of samples		
		1	5	10
random forest	[36]	0.879	0.894	0.904
	Ours	**0.903**	**0.941**	**0.953**
adaboost	[36]	0.887	0.905	0.910
	Ours	**0.909**	**0.931**	**0.940**
gradient boosting	[36]	0.905	0.919	0.915
	Ours	**0.927**	**0.921**	**0.928**

Table 4. The Area-Under-Curve (AUC) score on adversarial samples detection. Higher is better.

three different scales training datasets. The testing set is collected by selecting 1 image from each class in the ImageNet validation dataset. Each experiment is run five times independently. The results show that our method achieves an AUC score of 0.049 gain over Wang *et al.* [36] (*i.e.,* 0.953 *vs.* 0.904), when the number of training samples is 10 on random forest. It also demonstrates that the calculation paths obtained by our method are better than Wang *et al.* [36], with higher discriminability.

5 Conclusion

In this paper, we propose a novel architecture decoupling method to obtain an interpretable network and explore the rationale behind its overall working process based on a novel path-level analysis. In particular, an architecture controlling module is introduced and embedded into each layer to dynamically identify the activated filters. Then, by maximizing the mutual information between the architecture encoding vector and the input image, we decouple the network architecture to explore the functional processing behavior of each calculation path. Meanwhile, to further improve the interpretability of the network and inference, we limit the output of the convolutional layers and sparsifying the calculation path. Experiments show that our method can successfully decouple the network architecture with several merits, *i.e.,* network interpretation, network acceleration and adversarial samples detection.

Acknowledgements. This work is supported by the Nature Science Foundation of China (No.U1705262, No.61772443, No.61572410, No.61802324 and No.61702136), National Key R&D Program (No.2017YFC0113000, and No.2016YFB1001503), Key R&D Program of Jiangxi Province (No. 20171ACH80022) and Natural Science Foundation of Guangdong Provice in China (No.2019B1515120049).

References

1. Agakov, D.B.F.: The im algorithm: a variational approach to information maximization. NeurIPS (2004)
2. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. CVPR (2017)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
4. Bolukbasi, T., Wang, J., Dekel, O., Saligrama, V.: Adaptive neural networks for efficient inference. ICML (2017)
5. Chen, R., Chen, H., Huang, G., Ren, J., Zhang, Q.: Explaining neural networks semantically and quantitatively. ICCV (2019)
6. Chen, Z., Li, Y., Bengio, S., Si, S.: You look twice: Gaternet for dynamic filter selection in cnns. CVPR (2019)
7. Dong, X., Huang, J., Yang, Y., Yan, S.: More is less: A more complicated network with less inference complexity. CVPR (2017)
8. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. CVPR (2016)
9. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. CVPR (2017)
10. Fong, R., Vedaldi, A.: Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. CVPR (2018)
11. Gao, X., Zhao, Y., Dudziak, L., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. ICLR (2018)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. ICLR (2015)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR (2016)
14. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. IJCAI (2018)
15. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. ICCV (2017)
16. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. CVPR (2018)
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning (2015)
18. Kaiser, Ł., Bengio, S.: Can active memory replace attention? NeurIPS (2016)
19. Kaiser, Ł., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., Shazeer, N.: Fast decoding in sequence models using discrete latent variables. ICML (2018)
20. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. ICML (2017)
21. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
22. Lakkaraju, H., Kamar, E., Caruana, R., Horvitz, E.: Identifying unknown unknowns in the open world: Representations and policies for guided exploration. AAAI (2017)
23. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. ICLR (2016)

24. Liu, L., Deng, J.: Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. AAAI (2018)
25. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. ICCV (2017)
26. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. NeurIPS (2017)
27. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research (2008)
28. Morcos, A.S., Barrett, D.G., Rabinowitz, N.C., Botvinick, M.: On the importance of single directions for generalization. ICLR (2018)
29. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. International Conference on Machine Learning (2010)
30. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. NeurIPS Workshop (2017)
31. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV (2015)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. CVPR (2015)
34. Wang, J., Zhang, Z., Xie, C., Premachandran, V., Yuille, A.: Unsupervised learning of object semantic parts from internal states of cnns by population encoding. arXiv preprint arXiv:1511.06855 (2015)
35. Wang, X., Yu, F., Dou, Z.Y., Darrell, T., Gonzalez, J.E.: Skipnet: Learning dynamic routing in convolutional networks. ECCV (2018)
36. Wang, Y., Su, H., Zhang, B., Hu, X.: Interpret neural networks by identifying critical data routing paths. CVPR (2018)
37. Yiyou, S., Sathya N., R., Vikas, S.: Adaptive activation thresholding: Dynamic routing type behavior for interpretability in convolutional neural networks. ICCV (2019)
38. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? NeurIPS (2014)
39. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. International Conference on Machine Learning Workshop (2015)
40. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. ECCV (2014)
41. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. CVPR (2018)
42. Zhang, Q., Yang, Y., Wu, Y.N., Zhu, S.C.: Interpreting cnns via decision trees. CVPR (2019)
43. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware channel pruning for deep neural networks. NeurIPS (2018)