Architecture Disentanglement for Deep Neural Networks

Jie Hu[1], Liujuan Cao[1*], Tong Tong[1], Qixiang Ye[2], Shengchuan Zhang[1],
Ke Li[3], Feiyue Huang[3], Ling Shao[4], and Rongrong Ji[1,5,6].
[1]MAC Lab, School of Informatics, Xiamen University, [2]University of Chinese Academy of Sciences,
[3]Tencent Youtu Lab, [4]Inception Institute of Artificial Intelligence,
[5]Institute of Artificial Intelligence, Xiamen University, [6]Peng Cheng Lab.

Abstract

Understanding the inner workings of deep neural networks (DNNs) is essential to provide trustworthy artificial intelligence techniques for practical applications. Existing studies typically involve linking semantic concepts to units or layers of DNNs, but fail to explain the inference process. In this paper, we introduce neural architecture disentanglement (NAD) to fill the gap. Specifically, NAD learns to disentangle a pre-trained DNN into sub-architectures according to independent tasks, forming information flows that describe the inference processes. We investigate whether, where, and how the disentanglement occurs through experiments conducted with handcrafted and automatically-searched network architectures, on both object-based and scene-based datasets. Based on the experimental results, we present three new findings that provide fresh insights into the inner logic of DNNs. First, DNNs can be divided into sub-architectures for independent tasks. Second, deeper layers do not always correspond to higher semantics. Third, the connection type in a DNN affects how the information flows across layers, leading to different disentanglement behaviors. With NAD, we further explain why DNNs sometimes give wrong predictions. Experimental results show that misclassified images have a high probability of being assigned to task sub-architectures similar to the correct ones. Our code is available at `https://github.com/hujiecpp/NAD`.

1. Introduction

A fundamental problem in using deep neural networks (DNNs) is our inability to understand their inner workings, which is crucial in many real-world applications, including healthcare, criminal justice, and administrative regulation [28]. Recent studies also show that DNNs are easily fooled [25, 32, 19, 18] Thus, interpreting DNNs has attracted ever-increasing research attention in recent years. Existing endeavors typically link semantics to units or layers of DNNs to determine the roles of these specific parts. However, the hierarchical inference process is not effectively captured in this way, for two reasons. First, the trained networks entangle information together, and one unit can be responsible for multiple classes [23, 42]. Second, only knowing which unit or layer represents what class is insufficient for understanding the reasoning process in DNNs. The relationship between successive layers is not explored. For example, if we want to explain the inference process for ‘Airplane’, we know from previous works that the bottom layer activates ‘Blue Sky’ and the top layer activates ‘Airplane’, but this still fails to explain how DNNs infer from ‘Blue Sky’ to ‘Airplane’. Thus, if the network architectures could be disentangled into the sub-architectures in terms of tasks, such as classifying ‘Airplane’, the above concerns could be naturally addressed. For example, we can explain that the edges and colors from bottom layers are

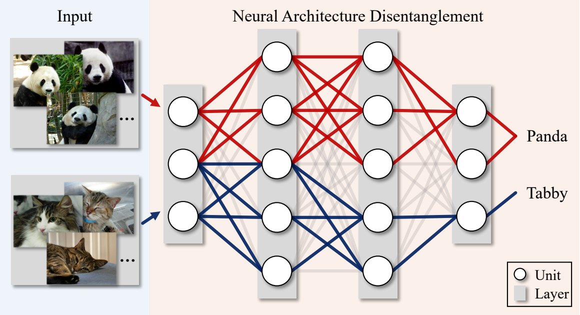


Figure 1: Illustration of the proposed neural architecture disentanglement (NAD). NAD aims to disentangle a pre-trained DNN into sub-architectures, each of which is responsible for only one task. For example, the network in this figure is disentangled into two sub-architectures, *i.e.*, the red one and the blue one, for the task of classifying ‘Panda’ and ‘Tabby’. Note that the sub-architectures can overlap with the same unit for different tasks.

[*]Corresponding author: caoliujuan@xmu.edu.cn

clustered in middle layers to form the parts, such as 'Wing' and 'Window', of 'Airplane', and then the most representative part will be selected to activate the class 'Airplane'.

In this paper, we introduce a new method termed neural architecture disentanglement (NAD), which learns to disentangle a pre-trained DNN into sub-architectures for different tasks. As illustrated in Fig. 1, the sub-architectures form the information flows that describe the inference processes. Inspired by representation disentanglement, we design an objective function constraining the information between successive layers of DNNs. The hidden units are selected to construct the sub-architectures from a DNN's bottom layers to its top layers. Extensive experiments are conducted to investigate whether, where, and how the disentanglement occurs. The architectures used in our experiments range from handcrafted to automatically-searched, *i.e.*, VGG16 [31], ResNet50 [11], DenseNet121 [14] and DARTS-Net [20]. Consistent results are obtained on both object-based and scene-based datasets, *i.e.*, ImageNet [7] and Place365 [41], yielding three new findings that provide fresh insights into the inner logic of DNNs.

First, we provide evidence that DNNs can be divided according to independent tasks, *i.e.*, DNNs can be disentangled. We compare the classification results of the original architectures and the disentangled sub-architectures in Fig. 2. After disentanglement, the Top@1 classification accuracies are squashed into the interval of $(90, 100]$ from $(70, 90]$ on ImageNet and $(40, 70]$ on Place365, which indicates that the sub-architectures correlate to the assigned tasks. Second, we find that deeper layers do not necessarily correspond to higher semantics, *i.e.*, the disentanglement can end before the last layer. Previous studies [2, 37] show that the bottom layers of DNNs extract low-level features (*e.g.*, edges and colors), while the top layers extract high-level features (*e.g.*, object parts) for classification. Our new observation is that the disentanglement of high-level information can end before the last layer in architectures with skip-connections. For example, as shown in Figs. 4b and 4d, the top class hit rates of ResNet50 [11] (which has plain skip-connections) and DARTS-Net [20] (which has automatically-searched skip-connections) appear in the 16th and 15th layer, respectively, instead of the last layer. Third, the connection type in a DNN affects how the information flows across layers, leading to different disentanglement behaviors. Intuitively, direct connections successively transmit information layer by layer, while skip-connections amortize the information over all layers. This makes the overall inference processes behave differently in the architectures with direct connections and with skip-connections. Importantly, dense skip-connections severely mix up the information for classification. For example, as shown in Fig. 4c, the visualized feature maps of DenseNet121 activate less useful patterns in some layers. However, the high-level information can still be extracted from the last layer, suggesting that the valuable information for classification is amortized into each layer. Furthermore, from the perspective of NAD, we provide an explanation for why DNNs sometimes give wrong predictions. Experimental results show that misclassified images have a high probability of being assigned to tasks with similar sub-architectures to the correct ones.

In summary, the contributions of this study include:

- We propose a new method, termed neural architecture disentanglement (NAD), to understand the inference process of DNNs.
- We study the properties of NAD with network architectures ranging from handcrafted to automatically-searched. Results on scene-based and object-based datasets yield three new findings for the inner workings of DNNs.

2. Related Work

Visualization-Based Interpretability. Visualization-based interpretability aims to reveal the interpretability of DNNs by visualization [8, 30, 36, 26, 24, 4]. Many studies have been conducted to explain DNNs by visualizing feature maps and then assigning semantic concepts to individual units or layers [10, 37]. For example, activation maximization [8, 27] optimizes a randomly initialized input to maximize a specific unit and then assigns a concept to this unit by observing the optimized input. Network dissection [2, 3] aligns individual hidden units with a set of semantic concepts using predefined pixel-level labels. The concept activation vector [15] interprets the internal layers of DNNs in terms of concepts by learning a normal vector of a plane that differentiates the selected concept from others. Concept whitening [5] tries to align layers to the activated concepts using image classes. However, the relationships between successive layers are not explored in the above methods. In contrast, our method explores how the features of the previous layers are selected for the latter layers, thus showing the overall inference processes for classifying specific classes.

Disentanglement-Based Interpretability. An interpretable DNN can also be trained by enforcing the disentanglement of representations [12, 38, 13] or distilling DNNs into more interpretable models [39, 9, 33, 21]. The work in [1] quantifies the disentanglement of representations based on information theory. The key idea of disentanglement-based interpretability is to align the learned representations' distribution to a standard Gaussian distribution, and then assign concepts to each independent dimension of the representations. Instead of learning disentangled representations from scratch, we focus on disentangling the entire architecture of a pre-trained DNN to interpret its inner workings. To this end, an objective function is constructed to constrain the information between successive layers of DNNs for the architecture disentanglement.

3. Method

3.1. Problem Formulation

NAD aims to decompose a pre-trained DNN into a set of sub-architectures for different tasks. As disentangling the decision-making process in the classifier leads to fixed one-hot vectors in the output layer, we focus on the information flows in the feature extractor of one DNN.

Let c denote the class of the target classification task to be disentangled from the DNNs. The image $\boldsymbol{x}^c$ is sampled from the set of images $\boldsymbol{X}^c$ labeled with $\boldsymbol{y}^c$. We define the layers in the feature extractor as n functions $f_0(\cdot), f_1(\cdot), ..., f_{n-1}(\cdot)$ from bottom to top, where $f_n(\cdot)$ is the classifier. For the input $\boldsymbol{x}^c$, we can obtain the feature maps in the original network as:

$$\begin{aligned}\boldsymbol{r}^c_0, \boldsymbol{r}^c_1, ..., \boldsymbol{r}^c_{n-1}, \widetilde{\boldsymbol{y}}^c =& f_0(\boldsymbol{x}^c), f_1\big(f_0(\boldsymbol{x}^c)\big),\\ &...,f_n\big(f_{n-1}(...f_0(\boldsymbol{x}^c)...)\big),\end{aligned} \tag{1}$$

where $\boldsymbol{r}^c_0, \boldsymbol{r}^c_1, ..., \boldsymbol{r}^c_{n-1}$ denote the output feature maps in the feature extractor, and $\widetilde{\boldsymbol{y}}^c$ denotes the output of the classifier. As there is a one-to-one relationship between the feature maps and the filters of the convolutional layers, selecting the feature maps of each layer is equivalent to selecting the filters for constructing the sub-architectures. Therefore, NAD finds the minimal combinations of feature maps in the feature extractor, so that the classifier has the maximal possibility to predict the label $\boldsymbol{y}^c$. The sub-architecture for the task of classifying c is constructed using the selected filters.

3.2. Representation Disentanglement

The target of NAD is similar to that of the representation disentanglement with the information bottleneck theory [34]. Given the input $\boldsymbol{x}^c \in \boldsymbol{X}^c$ with its representations $\boldsymbol{r}^c_{n-1} \in \boldsymbol{R}^c_{n-1}$ and label $\boldsymbol{y}^c \in \boldsymbol{Y}^c$, the objective function of the information bottleneck is defined as:

$$\mathcal{L}_{IB} = \beta \cdot \mathcal{I}(\boldsymbol{x}^c; \boldsymbol{r}^c_{n-1}) - \mathcal{I}(\boldsymbol{r}^c_{n-1}; \boldsymbol{y}^c), \tag{2}$$

where $\mathcal{I}(\cdot;\cdot)$ denotes the mutual information shared between two random variables, and $\beta > 0$ is a hyperparameter. The first term of the objective function indicates that the representations should use the input information as little as possible, while the second term ensures that they also keep as much information as possible for accomplishing downstream tasks. After derivation,[1] the variational upper bound of Eq. 2 is:

$$\begin{aligned}\widetilde{\mathcal{L}}_{IB} =& \mathbb{E}_{\boldsymbol{x}^c \sim P(\boldsymbol{x}^c)} \Big[\beta \cdot KL\big[P(\boldsymbol{r}^c_{n-1}|\boldsymbol{x}^c)||Q(\boldsymbol{r}^c_{n-1})\big]\\ &-\mathbb{E}_{\boldsymbol{r}^c_{n-1} \sim P(\boldsymbol{r}^c_{n-1}|\boldsymbol{x}^c)}\big[\log Q(\boldsymbol{y}^c|\boldsymbol{r}^c_{n-1})\big]\Big],\end{aligned} \tag{3}$$

where KL denotes the Kullback-Leibler (KL) divergence between two distributions, and $Q(\cdot)$ is the predefined distribution for variational approximation. Eq. 3 has the same formulation as the objective function in the standard variational autoencoder (VAE) [16]. By defining $Q(\cdot)$ as the Gaussian distribution and enlarging β for regularization, β-VAE [12] disentangles the factors into every single dimension of the representations under a self-supervised setting.

3.3. Architecture Disentanglement

For the architecture disentanglement, we modify the representation disentanglement from a single constraint to multiple constraints, and disentangle the factors between successive layers. Inspired by [6, 35], the networks can be interpreted as a Markov chain between successive representations:

$$\boldsymbol{y}^c \rightarrow \boldsymbol{x}^c \rightarrow \boldsymbol{r}^c_0 \rightarrow \boldsymbol{r}^c_1 \rightarrow ... \rightarrow \boldsymbol{r}^c_{n-1} \rightarrow \widetilde{\boldsymbol{y}}^c, \tag{4}$$

where the trained network extracts the representations for predicting the labels. The goal is to keep the information for classifying class c, while removing the redundant information for other classes. Therefore, between each adjacent layer, we reward the information related to the classification of class c, while penalizing the information of other classes.

For the i-th hidden layer, *i.e.*, when $1 \leq i < n$, the objective function can be written as:

$$\mathcal{L}^c_i = \beta \cdot \mathcal{I}(\boldsymbol{r}^c_{i-1}; \widetilde{\boldsymbol{r}}^c_i) - \mathcal{I}(\boldsymbol{r}^c_i; \widetilde{\boldsymbol{r}}^c_i), \tag{5}$$

where $\widetilde{\boldsymbol{r}}^c_i$ denotes the representation after constraining the information. As with Eq. 3, the variational upper bound of Eq. 5 can be derived as:

$$\begin{aligned}\widetilde{\mathcal{L}}^c_i = & \mathbb{E}_{\boldsymbol{r}^c_{i-1} \sim P(\boldsymbol{r}^c_{i-1})} \Big[\beta \cdot KL\big[P(\widetilde{\boldsymbol{r}}^c_i|\boldsymbol{r}^c_{i-1})||Q(\widetilde{\boldsymbol{r}}^c_i)\big]\\ &-\mathbb{E}_{\boldsymbol{r}^c_i \sim P(\boldsymbol{r}^c_i|\boldsymbol{r}^c_{i-1})}\big[\log Q(\widetilde{\boldsymbol{r}}^c_i|\boldsymbol{r}^c_i)\big]\Big].\end{aligned} \tag{6}$$

To optimize Eq. 6, we specify the parametric forms of the distributions, which is usually done by assuming a Gaussian distribution for regression and a multinomial distribution for classification. Therefore, we assume $Q(\widetilde{\boldsymbol{r}}^c_i), Q(\widetilde{\boldsymbol{r}}^c_i|\boldsymbol{r}^c_i)$ as the standard Gaussian distribution for regressing the representations, and define $\widetilde{\boldsymbol{r}}^c_i$ as:

$$\begin{aligned}\widetilde{\boldsymbol{r}}^c_i &= (\boldsymbol{\mu}^c_i + \boldsymbol{\epsilon}_i \cdot \boldsymbol{\sigma}^c_i) \cdot \boldsymbol{r}^c_i\\ &= (\boldsymbol{\mu}^c_i + \boldsymbol{\epsilon}_i \cdot \boldsymbol{\sigma}^c_i) \cdot f_i(\boldsymbol{r}^c_{i-1}),\end{aligned} \tag{7}$$

[1] Due to space limitations, we only show the final formulations in the main body of this paper. The derivations can be found in the supplementary material.

where $\boldsymbol{\epsilon}_i$ is random noise sampled from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $\boldsymbol{\mu}_i^c, \boldsymbol{\sigma}_i^c$ are the learnable parameters. With Eq. 7, we can correlate the information in successive layers as well as expanding the KL divergence term in Eq. 6, which defines the conditional probability as:

$$\begin{aligned} P(\widetilde{\boldsymbol{r}}_i^c|\boldsymbol{r}_{i-1}^c) &= \mathcal{N}\Big(f_i(\boldsymbol{r}_{i-1}^c) \cdot \boldsymbol{\mu}_i^c, diag\big[(f_i(\boldsymbol{r}_{i-1}^c) \cdot \boldsymbol{\sigma}_i^c)^2\big]\Big) \\ &= \mathcal{N}\Big(\boldsymbol{r}_i^c \cdot \boldsymbol{\mu}_i^c, diag\big[(\boldsymbol{r}_i^c \cdot \boldsymbol{\sigma}_i^c)^2\big]\Big). \end{aligned} \tag{8}$$

Combining the parametric forms of the distributions $Q(\widetilde{\boldsymbol{r}}_i^c), Q(\widetilde{\boldsymbol{r}}_i^c|\boldsymbol{r}_i^c)$, and $P(\widetilde{\boldsymbol{r}}_i^c|\boldsymbol{r}_{i-1}^c)$, the constraint in Eq. 6 for the i-th hidden layer can be derived as:

$$\begin{aligned} \widetilde{\mathcal{L}}_i^c = &\frac{\beta}{2}\big((\boldsymbol{r}_i^c \cdot \boldsymbol{\sigma}_i^c)^2 - \log{(\boldsymbol{r}_i^c \cdot \boldsymbol{\sigma}_i^c)^2} + (\boldsymbol{r}_i^c \cdot \boldsymbol{\mu}_i^c)^2 - 1\big) \\ &+ \frac{1}{2}\big(||\boldsymbol{r}_i^c - \boldsymbol{\mu}_i^c \cdot \boldsymbol{r}_i^c||_2^2 + \log 2\pi + \log{(\boldsymbol{r}_i^c \cdot \boldsymbol{\sigma}_i^c)^2}\big). \end{aligned} \tag{9}$$

For the classifier's output, *i.e.*, when $i = n$, we maximize the information between $\widetilde{\boldsymbol{y}}^c$ and $\boldsymbol{y}^c$ to guarantee that the constrained representations will keep the information for class c. The objective function is:

$$\mathcal{L}_n^c = -\mathcal{I}(\widetilde{\boldsymbol{y}}^c; \boldsymbol{y}^c). \tag{10}$$

Correspondingly, the variational upper bound of Eq. 10 is:

$$\widetilde{\mathcal{L}}_n^c = \mathbb{E}_{\boldsymbol{r}_{n-1}^c \sim P(\boldsymbol{r}_{n-1}^c)}\Big[\mathbb{E}_{\widetilde{\boldsymbol{y}}^c \sim P(\widetilde{\boldsymbol{y}}^c|\boldsymbol{r}_{n-1}^c)}\big[-\log Q(\boldsymbol{y}^c|\widetilde{\boldsymbol{y}}^c)\big]\Big]. \tag{11}$$

By assuming the variational distribution $Q(\boldsymbol{y}^c|\widetilde{\boldsymbol{y}}^c)$ as the multinomial distribution for classifying $\boldsymbol{y}^c$, we obtain the cross entropy loss:

$$\begin{aligned} \widetilde{\mathcal{L}}_n^c = &-\boldsymbol{y}^c \log f_n(\boldsymbol{r}_{n-1}^c) \\ &- (1 - \boldsymbol{y}^c) \log\big(1 - f_n(\boldsymbol{r}_{n-1}^c)\big) \\ = &-\boldsymbol{y}^c \log \widetilde{\boldsymbol{y}}^c - (1 - \boldsymbol{y}^c) \log(1 - \widetilde{\boldsymbol{y}}^c). \end{aligned} \tag{12}$$

3.4. Optimization of Objective Function

After expanding the cost function to Eq. 9, we can see that the optimization of μ_i^c has no relationship with σ_i^c. That is to say, when optimizing μ_i^c, the terms with σ_i^c in Eq. 9 can be seemed as constants. To simplify the optimization, we can just optimize μ_i^c and ignore the σ_i^c. After optimizing μ_i^c, we can reduce the noise level by fixing ϵ_i to its mean value, *i.e.*, 0, and calculate the $\widetilde{r}_i^c$ via Eq. 7, which frees σ_i^c to be any vector. Therefore, Eq. 9 becomes a much easier formulation to optimize:

$$\widetilde{\mathcal{L}}_i^c = \beta \cdot (\boldsymbol{r}_i^c \cdot \boldsymbol{\mu}_i^c)^2 + ||\boldsymbol{r}_i^c - \boldsymbol{\mu}_i^c \cdot \boldsymbol{r}_i^c||_2^2, \tag{13}$$

Algorithm 1 Neural Architecture Disentanglement

```
Input: The pre-trained network to be disentangled, the data $\boldsymbol{X}^c$ labeled with class $c$, the parameter $\boldsymbol{\mu}^c$ to be learned, the learning rate $\alpha$, and the hyperparameter $\beta$.

Output: The learned $\boldsymbol{\mu}^c$ for class $c$.
Initialize $\boldsymbol{\mu}^c$ randomly.
repeat
    Sample $\boldsymbol{x}^c$ from $\boldsymbol{X}^c$.
    For $i = 1, ..., n-1$, calculate $\widetilde{\boldsymbol{y}}^c$ and $\boldsymbol{r}_i^c$ by Eq. 1.
    For $i = 1, ..., n-1$, calculate $\widetilde{\boldsymbol{r}}_i^c$ by Eq. 7.
    Calculate the loss by Eq. 14 with Eq. 12 and Eq. 13.
    For $i = 1, ..., n-1$, update $\boldsymbol{\mu}_i^c$ by Eq. 15.
until Convergence
For $i = 1, ..., n-1$, discretize $\boldsymbol{\mu}_i^c$ by Eq. 16.
```

where the first term regularizes $\boldsymbol{\mu}_i$, and the second term constrains the reconstruction error for $\boldsymbol{r}_i^c$. Combining with Eq. 12 and Eq. 13, the overall objective function is:

$$\widetilde{\mathcal{L}}^c = \mathbb{E}_{\boldsymbol{x}^c \in \boldsymbol{X}^c}\Big[\sum_{i=0}^{n} \widetilde{L}_i^c\Big]. \tag{14}$$

To constrain the values to $(0, 1)$, we activate $\boldsymbol{\mu}_i^c$ using the Sigmoid function. Stochastic gradient descent is adopted to update $\boldsymbol{\mu}_i^c$, and L2 normalization is applied to the gradient for fast convergence. The update of $\boldsymbol{\mu}_i^c$ is defined as:

$$\boldsymbol{\mu}_i^{c'} = \boldsymbol{\mu}_i^c - \alpha \cdot norm(\frac{\partial \widetilde{\mathcal{L}}^c}{\partial \boldsymbol{\mu}_i^c}), \tag{15}$$

where α is the learning rate. The pre-trained networks are fixed during optimization. After convergence, we discretize the continuous $\boldsymbol{\mu}_i^c$ to binary values to form the disentangled sub-architecture for classifying classifier c:

$$\boldsymbol{\mu}_i^c = \begin{cases} 1, & \text{if } \boldsymbol{\mu}_i^c > 0.5 \\ 0, & \text{if } \boldsymbol{\mu}_i^c \leq 0.5, \end{cases} \tag{16}$$

where 0.5 is the threshold from the Sigmoid function. After obtaining $\boldsymbol{\mu}_i^c$ with $i = 0, .., n-1$ for all classes $c \in C$, we have naturally disentangled the original network. The identification of the disentangled portion is learned in training via μ_i^c. Therefore, the cost in Eq. 13 tries to find the optimal μ_i^c for class c. After discretize μ_i^c to 1 or 0, the portion with value 1 is used for class c. Alg. 1 summarizes the overall procedure of NAD.

4. Experiments

In this section, we investigate whether, where, and how NAD occurs to understand the inner workings of DNNs. We first check the classification results to ensure the information of classifying specific classes is disentangled with the

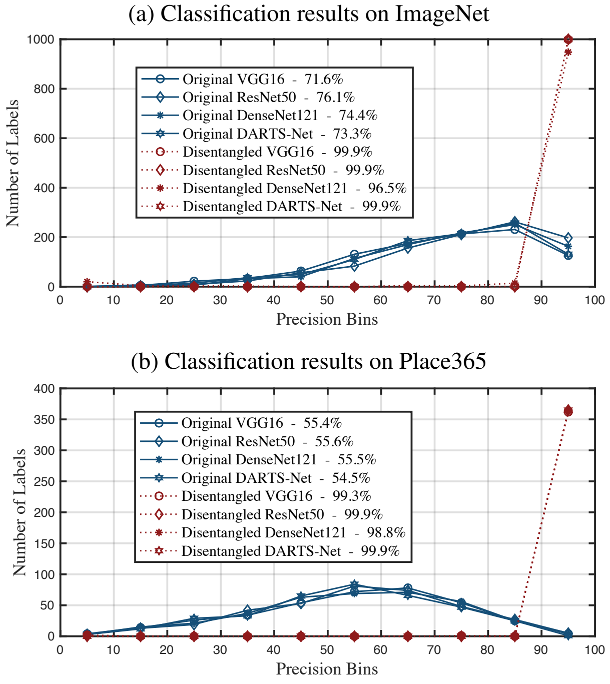


Figure 2: Histograms of the Top@1 classification accuracies (%) of the original architectures and the disentangled sub-architectures. The horizontal axis denotes the bins of Top@1 classification precision for one single label, in which the values are discretized into ten intervals, *i.e.*, $[0, 10], (10, 20], ..., (90, 100]$. The vertical axis denotes the number of classes whose classification accuracies fall into the same discretized bins. The solid lines show the results of the original architectures, and the dotted lines show the results of the disentangled sub-architectures. The average top@1 classification accuracies are listed in the legends. The classification results indicate that the disentangled sub-architectures are linked with their corresponding classes.

sub-architectures. Then, two indexes measuring the similarity and class hit rate of sub-architectures are designed to study where the networks begin to disentangle. Together with the visualizations of the activated feature maps, we try to understand how the entire networks work. In the end, we provide an explanation for why DNNs sometimes give wrong predictions.

4.1. Experimental Settings

Datasets and Network Architectures. We conduct experiments on both object-based and scene-based datasets, *i.e.*, ImageNet [7] and Place365 [41]. These datasets are organized in terms of the WordNet hierarchy [22], with each node being depicted by hundreds of images. We disentangle the original architectures using the training set, and the validation set is used to study the properties of DNNs. We select four network architectures, *i.e.*, VGG16 [31], ResNet50 [11], DenseNet121 [14] and DARTS-Net [20], to conduct our experiments. The models are pre-trained on the above two datasets, and their parameters are fixed when disentangling the architectures. The connection types include the direct connection in VGG, the plain skip-connection in ResNet, the dense skip-connection in DenseNet, and the automatically-searched skip-connection in DARTS-Net. The architecture of DARTS-Net is searched on the CIFAR10 [17] dataset.

Implementation Details. We set the learning rate $\alpha = 0.1$ and the iterations $N = 20$ to disentangle the DNNs for each class. We use the Elbow method to determine the hyper-parameter β for balancing the regularization term and the reconstruction term in the final objective function.[2] Following the experimental settings in [29, 40], feature map visualization is performed by bilinearly interpolating the size of feature maps to the size of input images and visualizing the top 5% of activated pixels.

4.2. Properties of NAD

4.2.1 Does Disentanglement Occur?

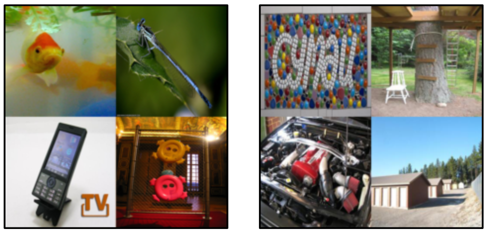
We check the single-class classification results for the disentangled sub-architectures and the original architectures, to investigate whether the disentanglement occurs in DNNs. Specifically, we perform classification with images from the same classes, by inputting them into their corresponding sub-architectures and the original architectures. The Top@1 classification accuracies (%) are recorded for each class, and the values are discretized into bins of $[0, 10], (10, 20], ..., (90, 100]$. We accumulate the number of classes whose accuracies fall into the same bins. From Fig. 2, we can see that the accuracies increase after disentanglement. For example, on ImageNet, the distributions of the original architectures mainly fall into the interval of $(70, 90]$, while those of the disentangled sub-architectures are squashed into the interval of $(90, 100]$. These results suggest that the sub-architectures can be disentangled to precisely relate to the assigned classes.

4.2.2 Where Does the Disentanglement Occur?

Considering that the sub-architectures can output the target classes regardless of which images are given, we design hit rate experiments to check if the disentangled sub-architectures can distinguish the corresponding classes from the randomly-combined images. To do so, we introduce two indexes, *i.e.*, the similarity and the hit rate for the sub-architectures.

Similarity Between Sub-Architectures. The first index is the similarity measurement. As the sub-architectures can be regarded as sets combined with selected filters, the similarity can be calculated by the Jaccard coefficient, which is

[2] More detailed results can be found in the supplementary material.



(a) ImageNet (b) Place365

Figure 3: Examples of randomly combined images for calculating the hit rate of the disentangled sub-architectures. Example (a) combines the images with 'label-concept': '1-Goldfish', '320-Damselfly', '487-MobilePhone', and '489-ChainLinkFence' from the validation set of ImageNet. Example (b) combines the images with 'label-concept': '34-BallPit', '339-TreeHouse', '28-AutoFactory', and '221-ManufacturedHome' from the validation set of Place365.

defined as the intersection size divided by the union size between two sets. We compute the average similarity for each layer between all pairs of disentangled sub-architectures. The results are shown in the first rows of Figs. 4a, 4b, 4c and 4d. Overall, the similarity between bottom layers is higher than that of top layers. Correspondingly, classes share low-level information in the bottom layers, and high-level semantics are gradually combined in the middle layers for classification. Therefore, the disentanglement tends to start in the middle layers, and the concrete concepts will gradually emerge from the middle layers to the top layers.

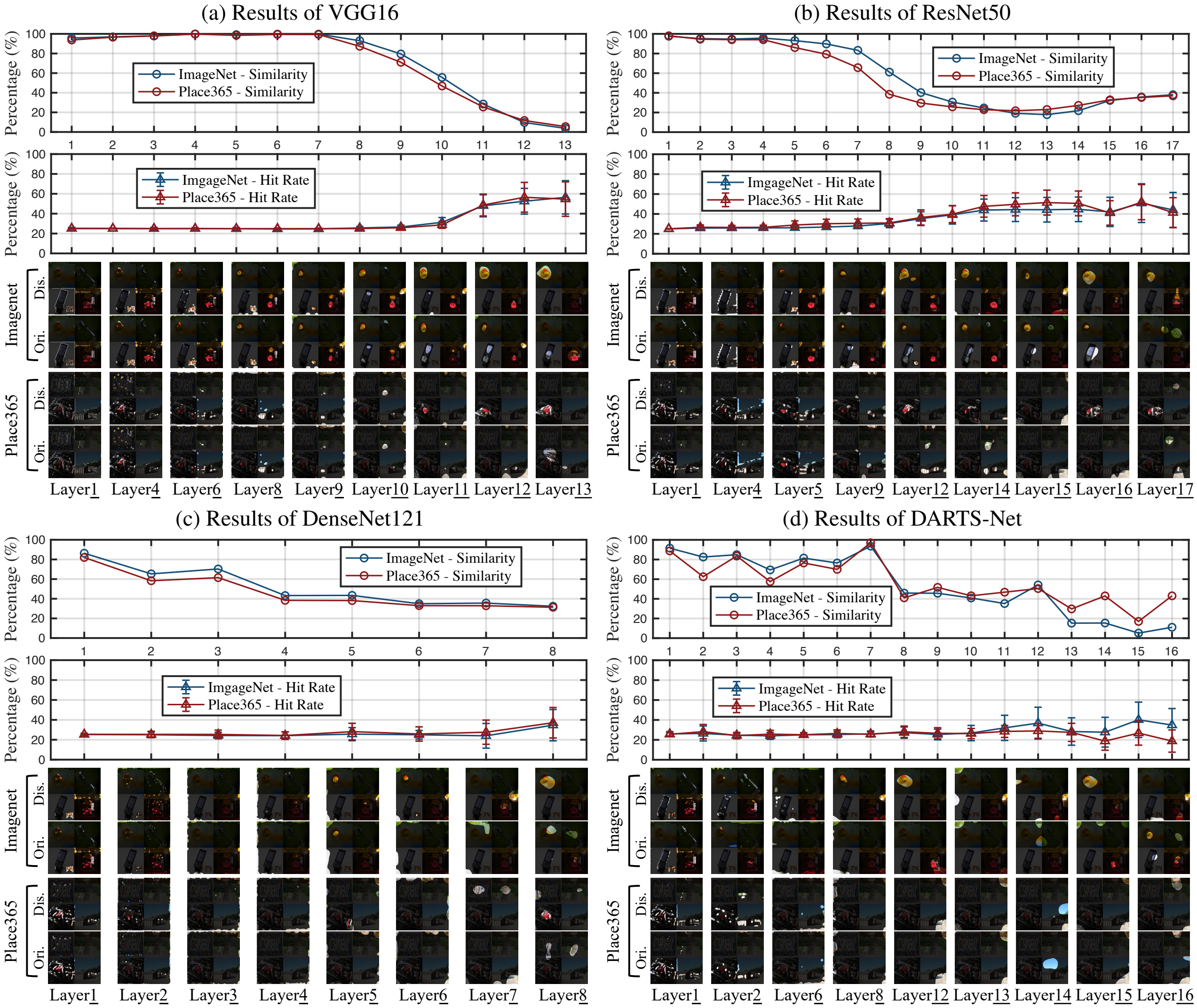
For VGG16 in Fig. 4a, the similarity is roughly above 90% in the first seven layers, which indicates that the low-level information extraction is conducted in these layers. Meanwhile, in the architectures with skip-connections, the results become more complex to analyze. For ResNet50 in Fig. 4b, the similarity begins to decrease early in the 4th layer and reaches the lowest point between the 11th and 14th layer. For DenseNet121 in Fig. 4c, the similarity is low in all layers. For DARTS-Net in Fig. 4d, the similarity values change drastically, especially in the 8th, 12th, and 15th layers. We believe that these results arise from the information fusion caused by the skip-connections. The high-level semantic information extracted in the top layers is brought forward to the middle or bottom layers by the skip-connection. To further verify our statement, we design the second index.

Hit Rate of Sub-Architectures. For the hit rate, we first combine four images with randomly selected labels to obtain the test images. Fig. 3 shows two examples of the combined images. We input the combined images to the disentangled sub-architectures and visualize the activated feature maps. We accumulate the number of correct hits at each pixel of the feature maps, and divide them by all the activated pixels. A correct hit means the activated pixel is located on the image of the correct class in the combined image. We carry out this experiment $1,000$ times using the randomly selected classes, and the mean values are computed for each layer. Note that using the original architectures in this experiment makes no sense, as the activated feature maps for the four selected classes in the combined images are the same. The hit rate results are shown in the second rows of Figs. 4a, 4b, 4c and 4d. The third and fourth rows show examples of the visualized feature maps for the ImageNet and Place365 datasets, respectively. 'Dis.' denotes the feature maps from the disentangled sub-architectures for the class 'Goldfish' in ImageNet and 'AutoFactory' in Place365. 'Ori.' denotes the visualized feature maps of the original architectures.

		VGG16	**ResNet50**	**DenseNet121**	**DARTS-Net**
Img.	Mis.	78.59	59.38	53.35	60.02
	Oth.	73.74	57.09	51.41	52.97
	Dif.	**-4.85**	**-2.29**	**-1.94**	**-7.05**
Plc.	Mis.	81.87	65.49	43.95	61.96
	Oth.	78.75	60.34	42.02	55.97
	Dif.	**-3.12**	**-5.15**	**-1.93**	**-5.99**

Table 1: Average similarity of two different pairs of sub-architectures. 'Mis.' denotes the average similarity between the sub-architectures of the correct labels and the sub-architectures of the misclassified labels. 'Oth.' denotes the average similarity between the sub-architectures of the correct labels and the sub-architectures of all other labels except the misclassified ones. 'Dif.' denotes the difference between 'Mis.' and 'Oth.'. 'Img.' denotes the ImageNet dataset, and 'Plc.' denotes Place365. All the values of 'Dif.' are greater than zero, which indicates that the misclassified input images have a high probability of being assigned to classes with similar sub-architectures to the correct ones.

For VGG16 in Fig. 4a, the hit rate begins to increase from the 8th layer and reaches the top point in the last two layers. We also find that the classes 'Goldfish' and 'AutoFactory' are gradually selected among the activated object parts in the visualization. This supports our claim that the disentanglement in VGG16 starts in the middle layers, and the classes are selected in the top layers. However, in the architectures with skip-connections, the results become far more complicated than VGG16, which has rarely been discussed in previous studies. For ResNet50 in Fig. 4b, the highest hit rate is not in the last layer but the 16th layer. The 12th, 13th, and 14th layers also get high hit rates. From the visualization, we also find that the classes 'Goldfish' and 'AutoFactory' are selected before the last layer, and the best hit rate correspondingly emerges in the 16th layer. For DenseNet121 in Fig. 4c, the hit rate gradually increases, but the values are not high in all layers. This could result from the dense skip-connection severely mixing up the information from different layers. The most complex results come


(a) Results of VGG16

(b) Results of ResNet50

(c) Results of DenseNet121

(d) Results of DARTS-Net

Figure 4: Results of the similarity, hit rate and feature map visualization. Best viewed in color, zoomed in for details.

from the automatically-searched architecture DARTS-Net in Fig. 4d. It yields different results on the two datasets. On ImageNet, the hit rate is high in the 12th and 15th layers, and the class 'Goldfish' is selected in the corresponding layers. However, the results are not good on Place365. We believe this is because the architecture of DARTS-Net is searched on the CIFAR10 dataset, which has similar classes to ImageNet but a large domain gap with Place365.

In summary, we find that the skip-connections in ResNet50 and DARTS-Net can make the disentanglement end early, while the dense skip-connection in DenseNet121 severely mixes information up. Specifically, the high-level semantic information can be extracted in the middle layers and sent to the top layers by the skip-connections in ResNet50 and DARTS-Net, which is quite different from what occurs in the VGG16 architecture with direct connections. Meanwhile, the dense skip-connection in DenseNet121 severely amortizes the information in every layer, making it perform similarly to VGG16 but more challenging to disentangle.

4.2.3 How Does the Disentanglement Occur?

We now study how the disentanglement occurs and investigate the inference process in DNNs. Generally, different networks have different connection types, causing different inference processes. Intuitively, direct connections successively transmit information layer by layer, while skip-connections amortize the information over all layers. In VGG16 with direct connections, the patterns start to group into concrete semantics in the middle layers, *i.e.*, the 8th layer to 10th layer, and the combined semantics are disentangled in the top layers, *i.e.*, the 11th layer to 13th layer. Such a process also emerges in the middle layers of ResNet50, *i.e.*, the 10th layer to the 16th layer. The differ-

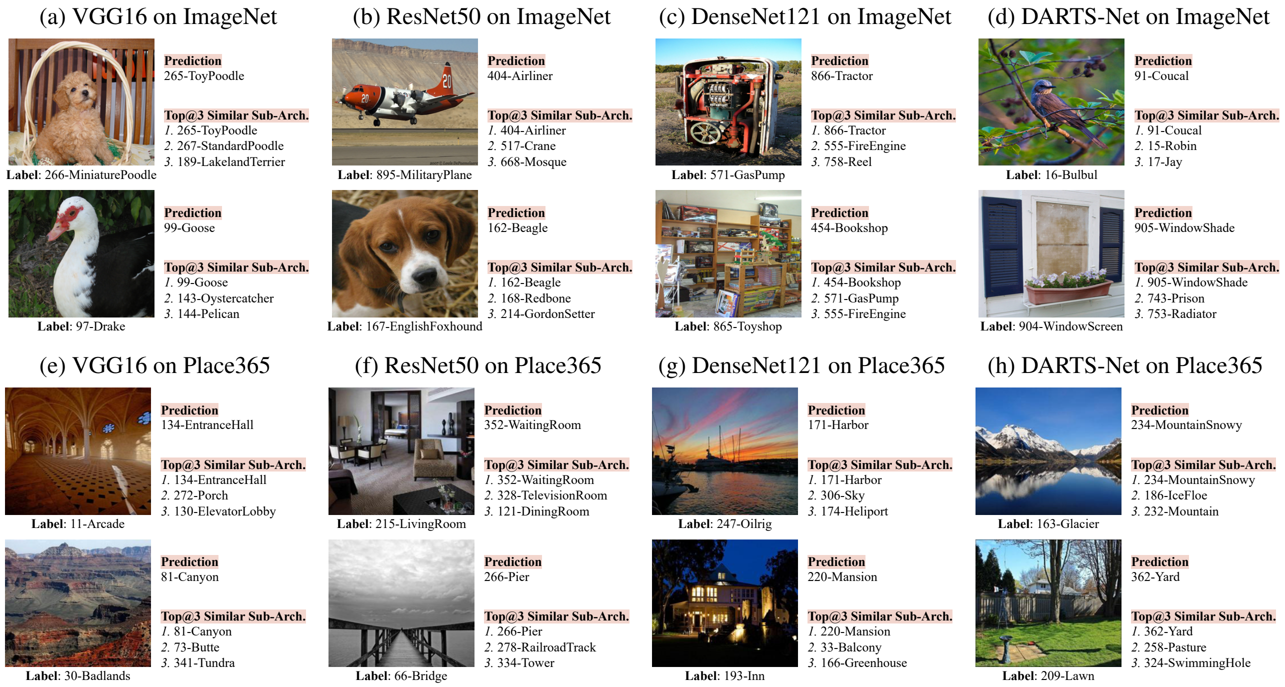


Figure 5: Examples of misclassified classes with their Top@1 classification predictions and the 'label-class' of the Top@3 similar sub-architectures with the correct labels.

ence is that the disentanglement skips over specific layers, such as the 15th, and ends in the 16th instead of the last layer. Early disentanglement stopping can also be found in DARTS-Net, tested on the ImageNet dataset, where the semantic concepts have already been selected in the 12th and 15th layers. By observing the visualization of the disentangled sub-architectures and the original architecture in DenseNet121, we find that some layers activate less valuable patterns in the input images for classification, *e.g.*, the 3rd layer and the 4th layer. However, the high-level information is still extracted in the last layer. This indicates that the information is severely mixed up in DenseNet121, where useful information for classification is amortized into each layer by dense skip-connections.

4.3. Why DNNs Misjudge

We try to explain why DNNs sometimes give wrong predictions. To this end, we show the misclassified labels of the input images, and compute the average similarity between the sub-architectures of the misclassified labels and the correct ones. For comparison, we also calculate the average similarity between the sub-architectures of the correct labels and all other labels except the misclassified ones. Given one class, the above experiment compares the sub-architecture similarity with the misclassified classes and other classes. The results, shown in Table 1, indicate that the misclassified input images tend to be assigned to classes with similar sub-architectures to the correct ones. Fig. 5 shows some examples of misclassified input images with their classification predictions and similar concepts of the sub-architectures. The sub-architectures tend to have similar semantic meanings to the target ones, *e.g.*, for the label "bulbul", the Top@3 similar sub-architectures are "Coucal", "Robin" and "Jay", which are all birds.

5. Conclusion

In this paper, we introduce neural architecture disentanglement (NAD) for better understanding DNNs. Starting from the current line of research, which aligns concepts to DNNs' units or layers, we try to link concepts to DNNs' sub-architectures instead. NAD learns to disentangle a pre-trained network in terms of tasks, forming information flows that describe the inference processes throughout the network. We investigate the properties of NAD on object-based and scene-based datasets with DNNs ranging from handcrafted to automatically-searched. Experimental results yield three new findings to explain the inner workings of DNNs, and an explanation of the classification performance is further discussed from the perspective of NAD. We hope NAD will shed light on the inference processes in DNNs for understanding how DNNs work.

Acknowledgements. This work is supported by the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No.U1705262, No. 62072386, No. 62072387, No. 62072389, No. 62002305, No.61772443, No.61802324 and No.61702136), Guangdong Basic and Applied Basic Research Foundation (No.2019B1515120049) and the Fundamental Research Funds for the central universities (No. 20720200077, No. 20720200090 and No. 20720200091).

References

[1] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 2018.

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[3] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *International Conference on Learning Representations*, 2019.

[4] Santiago A Cadena, Marissa A Weis, Leon A Gatys, Matthias Bethge, and Alexander S Ecker. Diverse feature visualizations reveal invariances in early layers of deep neural networks. In *Proceedings of the European Conference on Computer Vision*, 2018.

[5] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2020.

[6] Bin Dai, Chen Zhu, Baining Guo, and David Wipf. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, 2018.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[9] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[10] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, 2018.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[13] Jie Hu, Rongrong Ji, ShengChuan Zhang, Xiaoshuai Sun, Qixiang Ye, Chia-Wen Lin, and Qi Tian. Information competing process for learning diversified representations. In *Advances in Neural Information Processing Systems*, 2019.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[15] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, 2018.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv*, 2013.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Jie Li, Rongrong Ji, Peixian Chen, Baochang Zhang, Xiaopeng Hong, Ruixin Zhang, Shaoxin Li, Jilin Li, Feiyue Huang, and Yongjian Wu. Aha! adaptive history-driven attack for decision-based black-box models. In *Proceedings of the IEEE international conference on computer vision*, 2021.

[19] Jie Li, Rongrong Ji, Hong Liu, Jianzhuang Liu, Bineng Zhong, Cheng Deng, and Qi Tian. Projection & probability-driven black-box attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[21] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *IEEE International Conference on Data Mining Workshops*, 2018.

[22] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[23] Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018.

[24] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv preprint arXiv:1605.09304*, 2016.

[25] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[26] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[27] Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 2019.

[28] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

[29] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image

classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2014.

[33] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the AAAI/ACM Conference on AI*, 2018.

[34] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the Annual Allerton Conference on Communication, Control and Computing*, 1999.

[35] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, 2015.

[36] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 2014.

[38] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[39] Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[40] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[41] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[42] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.