Interpreting and Disentangling Feature Components of Various Complexity from DNNs

Jie Ren [*1] **Mingjie Li** [*1] **Zexu Liu** [1] **Quanshi Zhang** [1 2]

Abstract

This paper aims to define, visualize, and analyze the feature complexity that is learned by a DNN. We propose a generic definition for the feature complexity. Given the feature of a certain layer in the DNN, our method decomposes and visualizes feature components of different complexity orders from the feature. The feature decomposition enables us to evaluate the reliability, the effectiveness, and the significance of over-fitting of these feature components. Furthermore, such analysis helps to improve the performance of DNNs. As a generic method, the feature complexity also provides new insights into existing deep-learning techniques, such as network compression and knowledge distillation.

1. Introduction

Deep neural networks (DNNs) have demonstrated significant success in various tasks. Besides the superior performance of DNNs, some attempts have been made to investigate the interpretability of DNNs in recent years. Previous studies of interpreting DNNs can be roughly summarized into two types, *i.e.* the explanation of DNNs in a post-hoc manner (Lundberg & Lee, 2017; Ribeiro et al., 2016), and the analysis of the representation capacity of a DNN (Higgins et al., 2017; Achille & Soatto, 2018a;b; Fort et al., 2019; Liang et al., 2019).

This study focuses on a new perspective of analyzing the representation capacity of DNNs. *I.e.* we define, visualize, and analyze the complexity of features in DNNs. Previous research usually analyzed the theoretic maximum complexity of a DNN according to network architectures (Arora et al., 2016; Zhang et al., 2016; Raghu et al., 2017; Manurangsi & Reichman, 2018). In comparison, we propose to measure the complexity of features by analyzing the complexity of nonlinear transformations. The actual complexity of features caused by nonlinear transformations is usually different from the maximum complexity computed based on the network architecture.

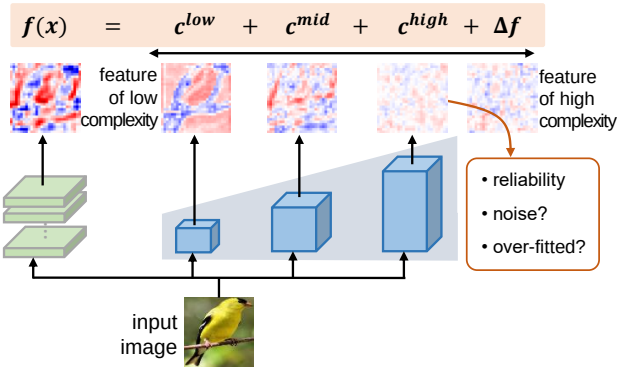

Figure 1. We decompose the raw feature into feature components of different complexity orders. We further visualize and analyze the feature components using some generic metrics.

In this paper, given the feature of a specific intermediate layer, we define the complexity of this feature as the minimum number of nonlinear transformations required to compute this feature under certain constraints. However, the quantification of nonlinear transformations presents significant challenges to state-of-the-art algorithms. Thus, we use the number of nonlinear layers to approximate the feature complexity. *I.e.* if a feature component can be computed using k nonlinear layers with a fixed width, but cannot be computed with $k-1$ nonlinear layers, we consider its complexity to be of the k-th order.

Analyzing DNNs using feature complexity. Based on the above definition, we decompose an intermediate-layer feature into feature components of different complexity orders, as Figure 1 shows. The clear feature decomposition enables both qualitative and quantitative analysis of a DNN as follows.

• We first visualize feature components of different complexity orders. Then, we explore the relationship between the feature complexity and the task difficulty. The distribution

[*]Equal contribution [1]Shanghai Jiao Tong University. [2]Quanshi Zhang is the corresponding author. He is with the John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, China. Correspondence to: Quanshi Zhang <zqs1022@sjtu.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

of feature components' strength over different complexity orders potentially reflects the difficulty of the task. A simple task usually makes the DNN mainly learn simple features.

• We further analyze the reliability, the effectiveness, and the significance of over-fitting for the decomposed feature components: (1) In this paper, *reliable feature components* refer to features that can be stably learned for the same task by DNNs with different architectures and parameters. (2) *The effectiveness of a feature component* is referred to as whether the feature component corresponds to neural activations relevant to the task. Usually, irrelevant neural activations can be considered as noises. (3) *The significance of over-fitting of a feature component* is quantified as the difference between a feature component's numerical contribution to the decrease of the training loss and its contribution to the decrease of the testing loss.

From the above perspectives, we discover:

(1) The number of training samples has small influence on the distribution of feature components' strength, but significant impacts on the feature reliability and the significance of over-fitting of feature components.

(2) Feature components of the complexity order, which is about the half of the depth of DNNs, are usually more effective in inference than other feature components.

• Improving performance. Above conclusions can be further used to improve performance of DNNs. We use feature components of low complexity orders, especially those with high effectiveness and reliability, to improve DNNs.

Method. More specifically, the feature decomposition into different complexity orders is inspired by knowledge distillation (Hinton et al., 2015). We consider the target DNN as the teacher network. Then, we design several neural networks (namely decomposer nets) with different depths to mimic the feature in an intermediate layer of the teacher network. In this way, we assume that feature components mimicked by shallow decomposer nets usually have low complexity. A deeper decomposer net can incrementally learn an additional feature component of a bit higher complexity order, besides components of low complexity.

In addition, different types of decomposer nets usually provide consistent feature decomposition. We find that the moderate change of the width or the architecture of decomposer nets does not significantly affect the distribution of feature components' strength over different complexity orders, which ensures the trustworthiness of our method. Besides, the feature decomposition also provides new insights into network compression and knowledge distillation.

Contributions of this work can be summarized as follows: (1) We propose a method to decompose, visualize, and analyze the complexity of intermediate-layer features in a DNN. We measure the minimum number of nonlinear transformations actually used to compute the feature, which is usually different from the maximum complexity of a DNN computed based on its architecture. (2) We visualize the feature components of different complexity orders. (3) We propose new metrics to analyze these feature components in terms of the reliability, the effectiveness, and the over-fitting level. Such metrics provide insightful analysis of advantages and disadvantages of the network compression and the knowledge distillation. (4) The feature decomposition improve the performance of DNNs.

2. Related Work

In this section, we discuss related studies in the scope of interpreting DNNs.

Visual explanations for DNNs. The most direct way to interpret DNNs includes the visualization of the knowledge encoded in intermediate layers of DNNs (Zeiler & Fergus, 2014; Simonyan et al., 2017; Yosinski et al., 2015; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016), and the estimation of the pixel-wise attribution/importance/saliency on input images (Ribeiro et al., 2016; Lundberg & Lee, 2017; Kindermans et al., 2017; Fong & Vedaldi, 2017; Zhou et al., 2016; Selvaraju et al., 2017; Chattopadhay et al., 2018; Zhou et al., 2015). They visualized salient regions in input images or salient feature units. In comparison, we propose to decompose and visualize feature components of different complexity orders, which provides a new perspective to understand DNNs.

Explanations for the representation capacity of DNNs. The evaluation of the representation capacity of DNNs provides a new perspective for explanations. The information-bottleneck theory (Wolchover, 2017; Shwartz-Ziv & Tishby, 2017) used the mutual information to evaluate the representation capacity of DNNs (Goldfeld et al., 2019; Xu & Raginsky, 2017). Achille & Soatto (2018b) further used the information-bottleneck to constrain the feature representation. Chen et al. (2018) proposed instance-wise feature selection for model interpretation. The CLEVER score (Weng et al., 2018) was used to estimate the robustness of DNNs. The stiffiness (Fort et al., 2019), the Fourier analysis (Xu, 2018), and the sensitivity metrics (Novak et al., 2018) were proposed to analyze the generalization capacity of DNNs. The canonical correlation analysis (CCA) (Kornblith et al., 2019) was used to measure the similarity between feature representations of DNNs. Liang et al. (2019) investigated the knowledge consistency between different DNNs.

Unlike previous methods, our research aims to explain a DNN from the perspective of feature complexity. In comparison, previous methods mainly analyzed the difficulty of optimizing a DNN (Arora et al., 2016; Blum & Rivest, 1989;

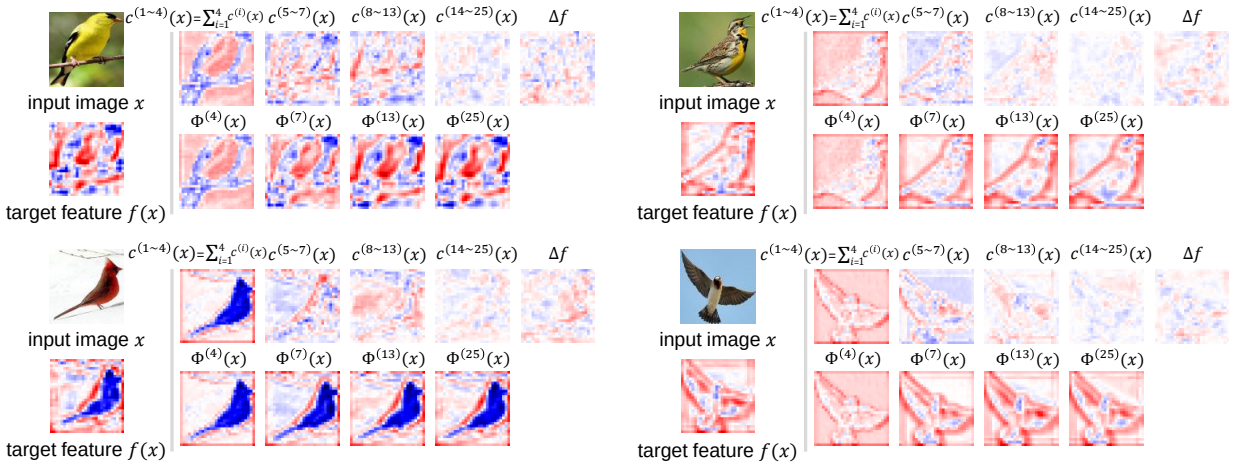

Figure 2. Visualization of the feature components. Feature components of low complexity orders were usually more significant than those of high complexity orders. Feature components of high complexity orders usually looked like noises.

Boob et al., 2018), the architectural complexity (Zhang et al., 2016), and the representation complexity (Liang et al., 2017; Cortes et al., 2017; Raghu et al., 2017), which are introduced as follows.

• Difficulty or computational complexity of optimizing a DNN: Some studies focus on the amount of computation, which is required to ensure a certain accuracy of tasks. Blum & Rivest (1989); Livni et al. (2014); Boob et al. (2018); Manurangsi & Reichman (2018) proved that learning a neural network with one or two hidden layers was NP-hard in the realizable case. Arora et al. (2016) showed that a ReLU network with a single hidden layer could be trained in polynomial time when the dimension of input was constant. Based on topological concepts, Bianchini & Scarselli (2014) proposed to evaluate the complexity of functions implemented by neural networks. Rolnick & Tegmark (2017) focused on the number of neurons required to compute a given function for a network with a fixed depth.

• Complexity measures of the feature representation in DNNs: Pascanu et al. (2013); Zhang et al. (2016) proposed three architectural complexity measures for RNNs. Raghu et al. (2017) proved the maximal complexity of features grew exponentially with depth. Liang et al. (2017); Cortes et al. (2017) measured the maximal complexity of DNNs with Rademacher complexity. Kalimeris et al. (2019) investigated the change of the mutual information between features in a DNN and features in a linear classifier. They found that during the learning process, the SGD optimizer learned functions of increasing complexity.

Unlike investigating the maximal complexity of DNNs based on the network architecture, we measure the feature complexity by exploring the complexity of nonlinear transformations, and visualize feature components of different complexity orders. Moreover, we analyze the quality of feature components and successfully boost the performance of DNNs with these feature components.

Knowledge distillation. Knowledge distillation is a popular and successful technique in knowledge transferring. Hinton et al. (2015) thought "soft targets" led to the superior performance of knowledge distillation. From another perspective, Lopez-Paz et al. (2016) unified knowledge distillation and privileged information. Phuong & Lampert (2019) explained the knowledge distillation from the view of data distribution, optimization bias, and the size of the training set. Cheng et al. (2020) explained the knowledge distillation by quantifying the knowledge. This paper uses knowledge distillation to decompose feature components of different complexities, and further explains the superior performance of knowledge distillation as the fact that knowledge distillation removes noisy components.

3. Definition, quantification and analysis of feature complexity

3.1. Complexity of feature components

Given an input x, let $f(x) \in \mathbb{R}^n$ denote the feature of a specific intermediate layer of the DNN. $y = \mathcal{D}(f(x)) \in \mathbb{R}^C$ is the output of the DNN, where $\mathcal{D}$ denotes the network module above $f(x)$. The basic requirement for feature complexity is to ensure that the entire feature $f(x)$ can be decomposed into feature components of different complexity orders.

$$f(x) = c^{(1)}(x) + c^{(2)}(x) + \ldots + c^{(L)}(x) + \Delta f \qquad (1)$$

where $c^{(l)}(x)$ denotes the feature component of the l-th complexity order (or, the *l-order complexity* for short). Δf is the feature component of a higher-order complexity.

Definition. Considering the above requirement for linear feature decomposition, the feature component $c^{(l)}(x)=c$ of the l-order complexity is defined as the feature component

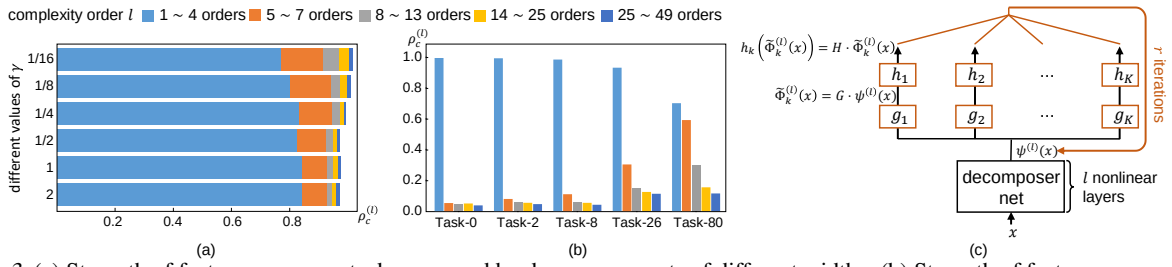

Figure 3. (a) Strength of feature components decomposed by decomposer nets of different widths. (b) Strength of feature components of different orders learned for different tasks. (c) The network for the decomposition of reliable feature components.

that can be computed using l nonlinear layers, but cannot be computed with $l-1$ nonlinear layers, when we constrain the network Φ to have a fixed width. *I.e.* the feature complexity of c is defined as $l = \text{argmin}_{l'}\{\Phi^{(l')}(x) = c\}$, where $\Phi^{(l')}(\cdot)$ denotes a neural network with l' nonlinear transformation layers, and $\sum_{l'=1}^{L} c^{(l')}(x)$ is a reconstruction of $f(x)$.

This definition poses a significant challenge, *i.e.* measuring the exact number of non-linear layers required to compute $f(x)$. It usually involves two problems: (1) people cannot exhaust all network architectures of Φ to guarantee a feature component is not able to be computed with less nonlinear layers; (2) even if given a good enough network architecture, people still cannot guarantee network parameters can be fully optimized, in order to calculate the component accurately. Therefore, we used the following method to approximately decompose features into different complexity orders.

Approximate-yet-efficient solution. Instead of directly decomposing the feature component $c^{(l)}$, we propose to use knowledge distillation to extract all feature components with the complexity of no higher than the l-th order, *i.e.* $\Phi^{(l)}(x) = \sum_{i=1}^{l} c^{(i)}(x)$. Given a trained DNN as the teacher (target) network, we select an intermediate layer f of the DNN as the target layer. $\Phi^{(l)}(x) = \sum_{i=1}^{l} c^{(i)}(x)$ is decomposed using another DNN (termed the *decomposer net*) with l nonlinear layers. The loss $\|f(x) - \Phi^{(l)}(x)\|^2$ is used to force $\Phi^{(l)}(x)$ to mimic $f(x)$, where $f(x)$ denotes the feature of the target network. We use decomposer nets with different depths $\Phi^{(1)}, \ldots, \Phi^{(L)}$ to extract feature components of different complexity orders. Thus, the feature component of the l-order complexity is given as follows.

$$\begin{aligned} &c^{(l)}(x) = \Phi^{(l)}(x) - \Phi^{(l-1)}(x), \\ &\text{where } \min_{\Phi^{(l)}} Loss = \|f(x) - \Phi^{(l)}(x)\|^2. \end{aligned} \tag{2}$$

In particular, $c^{(1)}(x) = \Phi^{(1)}(x)$. Thus, $f(x)$ is decomposed into two parts: $f(x) = \Phi^{(L)}(x) + \Delta f$ where Δf denotes the feature component with a higher complexity order than L.

Actually, it is not necessary to analyze feature components of every order, which leads to a large computational load. Instead, we compute feature components in intervals. Specifically, we calculate $c^{(1\sim l_1)}(x) = c^{(1)}(x) + \cdots + c^{(l_1)}(x) = \Phi^{(l_1)}(x), c^{(l_1+1\sim l_2)}(x) = c^{(l_1+1)}(x) + \cdots + c^{(l_2)}(x) = \Phi^{(l_2)}(x) - \Phi^{(l_1)}(x), c^{(l_2+1\sim l_3)}(x) = c^{(l_2+1)}(x) + \cdots + c^{(l_3)}(x) = \Phi^{(l_3)}(x) - \Phi^{(l_2)}(x)$, etc., instead of computing the component of each single order. This approximation does not significantly affect the objectiveness of the quantified distribution of feature components' strength over different complexity orders.

Strength of feature components ($\rho_c^{(l)}$). Furthermore, we quantify the strength of feature components of different complexity orders as the variance of feature components. The metric is designed as $\rho_c^{(l)} = Var[c^{(l)}(x)]/Var[f(x)]$, where $Var[c^{(l)}(x)] = \mathbb{E}_x[\|c^{(l)}(x) - \mathbb{E}_{x'}[c^{(l)}(x')]\|^2]$. For fair comparisons between different DNNs, we use the variance of the entire feature $f(x)$ to normalize $Var[c^{(l)}(x)]$. The variance indicates the numerical impact of the feature component $c^{(l)}(x)$ to $f(x)$. $\rho_c^{(l)}$ represents the relative strength of the l-th order complex feature component *w.r.t.* the entire feature.

Limitations: accurate estimation vs. fair comparison. Theoretically, if the target DNN has D nonlinear layers, the complexity of its features must be no higher than the D-th order, *i.e.* $\Phi^{(D')}(x) = f(x)$, $D' \le D$. However, the optimization capacity for the learning of decomposer nets is limited. A decomposer net with D nonlinear layers cannot learn all features in $f(x)$. Thus, when $\Phi^{(D')}(x) \approx f(x)$ in real implementations, we have $D' \ge D$.

In this way, $\rho_c^{(l)}$ measures the relative distribution of feature components' strength over different complexity orders, instead of an accurate strength of feature components. Nevertheless, as Figure 3(a) shows, even if we use decomposer nets with different architectures (different widths), we still get similar distributions of feature components' strength. This proves the trustworthiness of our method, and enables the fair comparison between different DNNs.

Decomposer nets. We design decomposer nets $\Phi^{(1)}(x), \ldots, \Phi^{(L)}(x)$ with residual architectures. The decomposer net consists of three types of residual blocks, each type having m blocks. Each block of the three types consists of a ReLU layer and a convolutional layer with $128\gamma, 256\gamma, 512\gamma$ channels, respectively. In most experiments, we set $\gamma = 1$, but in Figure 3(a), we try different values of γ to test decomposer nets of different

widths. We use two additional convolutional layers before and after all $3m$ blocks, respectively, to match the input and output dimensions. Therefore, a decomposer net contains $3m+2$ convolutional layers and $l=3m+1$ ReLU layers. For fair comparisons between DNNs, we use the same set of decomposer nets to measure the complexity of each DNN.

Various decomposer nets generate similar distributions of feature components' strength, which demonstrates the trustworthiness of our methods. To verify this, we learn a target DNN for Task-26 (which will be introduced later) on the CIFAR-10 dataset and decompose the output feature of the target DNN. We use decomposer nets with different widths (different values of γ) for analysis. We analyze the complexity of the output feature of the last convolutional layer. We set $m=1,2,4,8,16,32$, so that the nonlinear layer numbers of decomposer nets are set to $l_1=4, l_2=7, l_3=13, l_4=25, l_5=49, l_6=97$ according to settings in the *approximate-yet-efficient solution* paragraph. Thus, we calculate $c^{(1\sim4)}(x), c^{(5\sim7)}(x)$, etc. To boost the learning efficiency, we used parameters of the learned $\Phi^{(l_i)}$ to initialize first l_i layers in $\Phi^{(l_{i+1})}$. Figure 3(a) compares distributions of feature components' strength computed using different decomposer nets. We find that decomposer nets with different widths generate similar distributions of feature components' strength, thereby verifying the trustworthiness of our method.

The relationship between the task complexity and the feature complexity. We define the complexity of tasks first. Let *Task-n* denote a task of the n-order complexity as follows: we construct another network (namely the task DNN) with n ReLU layers and randomly initialized parameters (without further training), whose output is an $8\times8\times64$ tensor. We learn the target DNN[1] to reconstruct this output tensor via an MSE loss. Since the task DNN contains n ReLU layers, we use Task-n to indicate the complexity of mimicking the task DNN. Due to the randomness of the initialized parameters in the task DNN, when n is large, the task complexity is high. Figure 3(b) compares distributions of feature components' strength computed on target DNNs learned for Task-0, Task-2, Task-8, Task-26, and Task-80. We find that DNNs learned for more complex tasks usually encode more high-complexity feature components.[2] Let us take target DNNs learned for Task-0 and Task-80 for example. For the target DNN learned for Task-0, the significance of the 4-order feature component is much higher than the significance of higher-order feature components. However, in the target DNN learned for Task-80, 7-order and the 13-order feature components are strengthened significantly.

[1]For simplicity, we design the target DNN to have the same architecture as the decomposer net with $l=19$.

[2]Note that if the target DNN has D nonlinear layers, the actual complexity of its features can be higher than the D-th order, *i.e.* $\rho_l^{(D')}>0$ for $D'>D$, due to the limited optimization capacity of learning decomposer nets. Please see the paragraph *limitations: accurate estimation vs. fair comparison* for details.

3.2. Reliability of feature components

Understanding the reliability of a set of feature components $\Phi^{(l)}(x)=\sum_{i=1}^{l}c^{(i)}(x)$. We aim to decompose reliable feature components $\Phi^{(l),\text{reli}}(x)$ and unreliable feature components $\Phi^{(l),\text{unreli}}(x)$. The goal can be represented as

$$\Phi^{(l)}(x)=\Phi^{(l),\text{reli}}(x)+\Phi^{(l),\text{unreli}}(x) \tag{3}$$

As discussed in (Liang et al., 2019), DNNs with different initializations of parameters usually learn similar feature representations for the same task, and these similar features are proved to be reliable for the task. Thus, we consider the reliable feature components as features that can be stably learned by different DNNs trained for the same task. Suppose that we have K different DNNs learned for the same task. For each DNN, we select the feature of a specific intermediate layer as the target feature. Let $f_1(x), f_2(x), \ldots, f_K(x)$ denote target features of K DNNs. We aim to extract features shared by $f_1(x), f_2(x), \ldots, f_K(x)$. Let $\Phi_1^{(l),\text{reli}}(x), \Phi_2^{(l),\text{reli}}(x), \ldots, \Phi_K^{(l),\text{reli}}(x)$ denote reliable feature components extracted from the K DNNs, which are shared by $f_1(x), f_2(x), \ldots, f_K(x)$, respectively. Then, we can consider the goal is to learn $\Phi_1^{(l),\text{reli}}(x), \Phi_2^{(l),\text{reli}}(x), \ldots, \Phi_K^{(l),\text{reli}}(x)$, which satisfies each pair of $\Phi_i^{(l),\text{reli}}(x)$ and $\Phi_j^{(l),\text{reli}}(x)$ are able to reconstruct each other by a linear transformation:

$$\Phi_i^{(l),\text{reli}}(x)=r_{j\to i}(\Phi_j^{(l),\text{reli}}(x)),\ \Phi_j^{(l),\text{reli}}(x)=r_{i\to j}(\Phi_i^{(l),\text{reli}}(x)) \tag{4}$$

where $r_{i\to j}$ and $r_{j\to i}$ denote two linear transformations.

Computation of reliable feature components that satisfy requirements in Eq. (3) and Eq. (4). We design a direct and simple algorithm to decompose reliable and unreliable feature components. The decomposition can provide new insights into the feature representation capacity of DNNs, as experiments in Section 4 show.

Inspired by the CycleGAN (Zhu et al., 2017), we apply the idea of cycle consistency on knowledge distillation to extract reliable feature components. To extract reliable feature components, we construct the following neural network for knowledge distillation. As Figure 3(c) shows, the network has a total of l ReLU layers. We add K parallel additional convolutional layers $g_1, g_2, \ldots, g_K$ to generate K outputs $\widetilde{\Phi}_1^{(l)}(x), \widetilde{\Phi}_2^{(l)}(x), \ldots, \widetilde{\Phi}_K^{(l)}(x)$, to mimic $f_1(x), f_2(x), \ldots, f_K(x)$, respectively. More specifically, $\widetilde{\Phi}_k^{(l)}(x)=g_k(\psi^{(l)}(x))=G_k\cdot\psi^{(l)}(x)$, where G_K is a matrix, and $\psi^{(l)}(x)$ denotes the output of the decomposer net with l ReLU layers. Then, the distillation loss is given as $\mathcal{L}^{\text{distill}}=\sum_{k=1}^{K}\|f_k(x)-\widetilde{\Phi}_k^{(l)}(x)\|^2$.

To enable $\widetilde{\Phi}_k^{(l)}(x)$ and $\widetilde{\Phi}_{k'}^{(l)}(x)$ to reconstruct each other linearly, we apply the idea of cycle consistency. We first use $\widetilde{\Phi}_k^{(l)}(x)$ to reconstruct $\psi^{(l)}(x)$ by using another linear transformation $h_k(\widetilde{\Phi}_k^{(l)}(x)) = H_k \cdot G_k \cdot \psi^{(l)}(x)$ to fit $\psi^{(l)}(x)$. Similarly, we can also use $\widetilde{\Phi}_{k'}^{(l)}(x)$ to reconstruct $\psi^{(l)}(x)$ by fitting $h_{k'}(\widetilde{\Phi}_{k'}^{(l)}(x)) = H_{k'} \cdot G_{k'} \cdot \psi^{(l)}(x)$ to $\psi^{(l)}(x)$, as shown in Figure 3 (c). In this way, we consider $\widetilde{\Phi}_k^{(l)}(x)$ and $\widetilde{\Phi}_{k'}^{(l)}(x)$ construct each other through $\psi^{(l)}(x)$. We conduct cycle reconstructions between $\psi^{(l)}(x)$ and $\widetilde{\Phi}_k^{(l)}(x)$ for R iterations ($R=10$ in experiments) to ensure a certain reconstruction accuracy. Let $\psi_0^{(l)}(x) = \psi^{(l)}(x)$, $\psi_r^{(l)}(x) = \mathbb{E}_k[h_k \circ g_k \circ \psi_{r-1}^{(l)}(x)]$ denote the reconstruction output in the r-th iteration, where $h_k \circ g_k$ denotes the cascaded layerwise operations. The cycle construction loss is as follows.

$$\mathcal{L}^{\text{cycle}} = \sum\nolimits_{r=1}^{R} \sum\nolimits_{k=1}^{K} \|h_k \circ g_k \circ \psi_{r-1}^{(l)}(x) - \psi_{r-1}^{(l)}(x)\|^2 \quad (5)$$

This loss makes the feature $\widetilde{\Phi}_k^{(l)}(x)$ approximately shared by K DNNs. In this way, $\Phi_k^{(l),\text{reli}}(x) = \widetilde{\Phi}_k^{(l)}(x)$ can be considered as the reliable feature component. Compared with the traditional cycle consistency (Zhu et al., 2017), the above loss is much simpler and requires less computational cost. In this way, we can decompose the unreliable feature component of the k-th DNN as $\Phi_k^{(l),\text{unreli}}(x) = \Phi_k^{(l)}(x) - \Phi_k^{(l),\text{reli}}(x)$, with the other $K-1$ DNNs as assistant DNNs. In experiments, we set $K=3$, including a target DNN and DNNs A and B. The reliable feature components are shared by the three DNNs. DNNs A and B have been well-trained as two additional assistant DNNs, namely *exemplary DNNs*, in order to decompose reliable and unreliable feature components from the target DNN. The exemplary DNNs A and B are selected as those with state-of-the-art performance in the target task, in order to obtain convincing results. To enable fair comparisons, the same pair of DNNs A and B are uniformly used to analyze various DNNs.

Reliability of feature components in $\Phi_k^{(l)}(x)$ can be quantified as the ratio of reliable feature components in $\Phi_k^{(l)}(x)$ as $\rho^{(l),\text{reli}} = Var[\Phi_k^{(l),\text{reli}}(x)]/Var[\Phi_k^{(l)}(x)]$.

3.3. Other metrics to evaluate feature components

Effectiveness of feature components ($\alpha_{\text{effective}}^{(l)}$) measures whether the feature components $c^{(l)}(x)$ extracted from the training sample x directly contributes to the task. We compute the Shapley value of each l-order feature component to quantify the numerical contribution of this feature component, *i.e.* its contribution to the decrease of the task loss. The Shapley value is widely considered as a standard metric for feature importance in literature (Chen et al., 2019; Ghorbani & Zou, 2019; Williamson & Feng, 2020). Given a set of feature components $\mathcal{C} = \{c^{(l)}(x),\ l = 1, \ldots, L\}$, let $S \subseteq \mathcal{C}$ denote a subset of $\mathcal{C}$ and $f'_S = \sum_{c^{(l)}(x) \in S} c^{(l)}(x)$ is the sum of feature components in the subset S. Then, $v(S) = -\mathbb{E}_{x \in X^{\text{train}}}[\mathcal{L}^{\text{task}}(y^{\text{truth}}, y = D(f'_S + \Delta f)_{\text{given } x})]$ denotes the negative value of the task loss when we only use feature components in the subset S for inference. Δf is the high-order component in Eq. (1). Then, the Shapley value of the l-order feature components *w.r.t.* the decrease in the task loss is computed as follows (Shapley, 1953).

$$\varphi_l^{\text{train}} = \sum_{S \subseteq \mathcal{C} \setminus \{c^{(l)}(x)\}} p(S) \Big[v(S \cup \{c^{(l)}(x)\}) - v(S) \Big] \quad (6)$$

where $p(S) = \frac{(L-|S|-1)!|S|!}{L!}$. In this way, numerical contributions of all the L feature components can be fairly allocated and given as $\varphi_1^{\text{train}} + \varphi_2^{\text{train}} + \cdots + \varphi_L^{\text{train}} = \mathbb{E}_{x \in X_{\text{train}}}[\mathcal{L}^{\text{task}}(y^{\text{truth}}, y = D(\Delta f)_{\text{given } x}) - \mathcal{L}^{\text{task}}(y^{\text{truth}}, y = D(f)_{\text{given } x})]$. Thus, the metric $\alpha_{\text{effective}}^{(l)} = \varphi_l^{\text{train}} / \sqrt{Var[c^{(l)}(x)]}$ measures the normalized effectiveness of the feature component $c^{(l)}$ to the decrease of the training loss. $\sqrt{Var[c^{(l)}(x)]}$ is used for normalization.

Significance of over-fitting of feature components ($\alpha_{\text{overfit}}^{(l)}$) measures whether $c^{(l)}(x)$ is over-fitted to specific training samples. Just like the effectiveness, we use the Shapley value to measure the numerical contribution $\varphi_l^{\text{overfit}}$ of each l-order feature component to the over-fitting level. We use $\mathcal{L}_{\text{overfit}}(f) = \mathbb{E}_{x \in X_{\text{test}}}[\mathcal{L}^{\text{task}}(y^{\text{truth}}, y = D(f)_{\text{given } x})] - \mathbb{E}_{x \in X_{\text{train}}}[\mathcal{L}^{\text{task}}(y^{\text{truth}}, y = D(f)_{\text{given } x})]$ to represent the significance of over-fitting when we use the feature f for inference. In this case, $v(S) = \mathcal{L}_{\text{overfit}}(f'_S + \Delta f)$ quantifies the over-fitting level caused by both feature components in Δf and feature components in S. We plug such definition of $v(S)$ to the formulation of Shapley values in Eq. (6) to compute the Shapley value $\varphi_l^{\text{overfit}}$. In this way, the computed Shapley value $\varphi_l^{\text{overfit}}$ represents the numerical contribution of the l-order feature components to the over-fitting level. Based on Shapley values, we have $\varphi_1^{\text{overfit}} + \varphi_2^{\text{overfit}} + \cdots + \varphi_L^{\text{overfit}} = \mathcal{L}_{\text{overfit}}(f) - \mathcal{L}_{\text{overfit}}(\Delta f)$. Then, the metric of the significance of over-fitting for $c^{(l)}$ is given as $\alpha_{\text{overfit}}^{(l)} = \varphi_l^{\text{overfit}} / \varphi_l^{\text{train}}$.

4. Experiments

Datasets, DNNs & Implementation details. We used our method to analyze VGG-16 (Simonyan et al., 2017) and ResNet-8/14/18/20/32/34/44 (He et al., 2016).[3] For simplicity, we limited our attention to coarse-grained and fine-grained classification. We trained these DNNs based on the CIFAR-10 dataset (Krizhevsky et al., 2009) and the CUB200-2011 dataset (Wah et al., 2011). For the CUB200-2011 dataset, we used object images cropped by object bounding boxes for both training and testing.

Visualization of feature components. Given a trained

[3] Compared with the original VGG-16, we added a BatchNorm layer before the output feature of each convolutional layer, before we use its feature to guide the distillation process. ResNet-8 and ResNet-14 had the similar structure as ResNet-20, ResNet-32 and ResNet-44 in (He et al., 2016), except that they had 1 and 2 blocks in each stage, respectively.

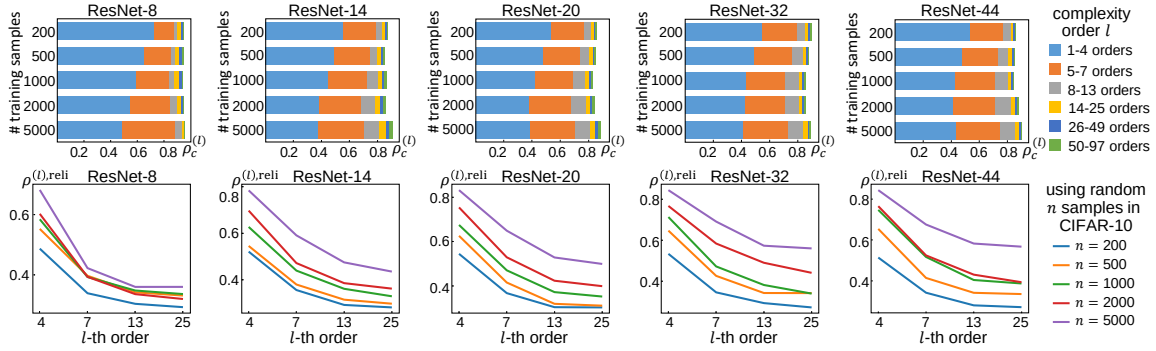

Figure 4. Strength ($\rho_c^{(l)}$) and reliability ($\rho^{(l),\text{reli}}$) of the decomposed feature components.

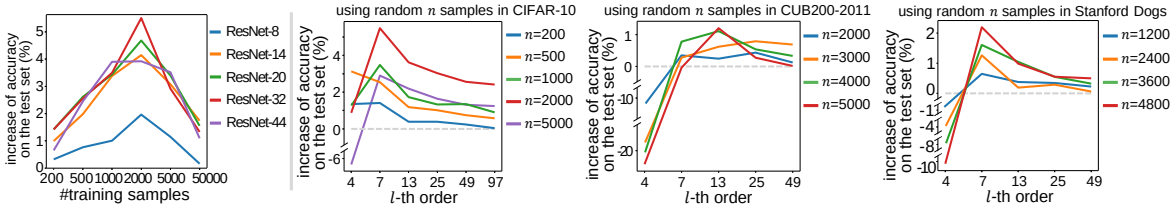

Figure 5. Improvements of the classification accuracy based on $\Phi^{(l)}(x)$. (left) The accuracy improvement of different DNNs learned on the CIFAR-10 dataset with $l = 7$ using different number of training samples. (right) The accuracy improvement with different values of l. Here ResNet-32 was learned on the CIFAR-10 dataset, and ResNet-34 was learned on the CUB200-2011 dataset and the Stanford Dogs dataset, respectively.

VGG-16 and images in the CUB200-2011 dataset, we decomposed and visualized feature components of different orders. We took the feature in the `conv_4-3` layer (with the size of $28 \times 28 \times 512$) as the target feature $f(x)$. Figure 2 shows the feature map of a random channel in $f(x)$, and the corresponding channel in $c^{(l)}(x)$ and $\Phi^{(l)}(x)$. Low-complexity components usually represented the general shape of objects, while high-complexity components corresponded to detailed shape and noises.[2]

Exp. 1, the number of training samples had small influence on the distribution of feature components' strength, but had significant impacts on the feature reliability. We learned ResNet-8/14/20/32/44 using different numbers of training samples, which were randomly sampled from the the CIFAR-10 dataset. Then, we decomposed feature components and reliable feature components of different complexity orders from the output feature of the last residual block. More specifically, two exemplary DNNs A and B were used to help us extract the reliable feature components in the target feature. They were implemented as ResNet-44 learned on the entire CIFAR-10 dataset with different initial parameters.

Figure 4 compares the strength of feature components $\rho_c^{(l)}$ and the reliability of feature components $\rho^{(l),\text{reli}}$ in different DNNs.[2] The DNN learned from the larger training set usually encoded more complex features, but its overall distribution of feature components' strength was very close to that of the DNN learned from the smaller training set. This indicated that the number of training samples had small impacts on the strength of feature components of different complexity orders. However, in Figure 4 (bottom), DNNs learned from many training samples always exhibited higher reliability than DNNs learned form a few training samples, which meant that the increase of the number of training samples would help DNN learn more reliable features.

Exp. 2, improvement of the classification accuracy based on $\Phi^{(l)}(x)$. We compared the effectiveness $\alpha_{\text{effective}}^{(l)}$ and the over-fitting level $\alpha_{\text{overfit}}^{(l)}$ of feature components[2] of different DNNs in Figure 6. We found that **(1)** when the complexity order of feature components is about the half of the depth of the DNN, these feature components exhibited the highest effectiveness. For ResNet-8, the feature component with the highest effectiveness was of the 4-th order. For ResNet-14, the 7-order feature component was the most effective.

(2) Low-complexity feature components learned from a small number of samples were usually more over-fitted than low-complexity components learned from many samples. However, we could not summarize clear conclusions from the significance of over-fitting for high-complexity components. However, for high-complexity feature components, we can not summarize clear relationships between the sample number and the over-fitting level. This might be due to the low effectiveness of these high-complexity feature components (noise-like features).

Based on above observations, we further tested the classifi-

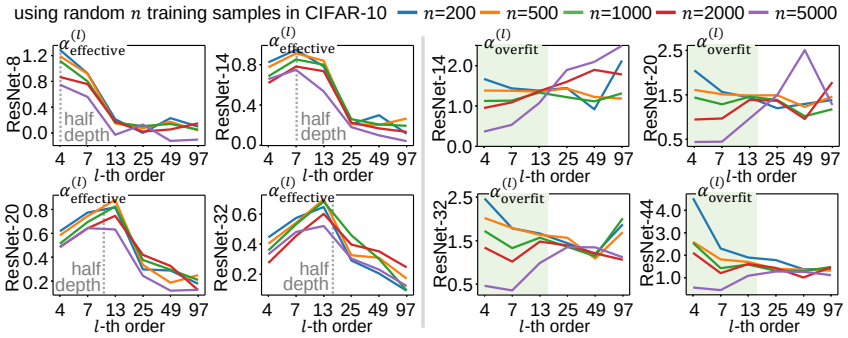

Figure 6. Comparisons of (left) the effectiveness of feature components $\alpha^{(l)}_{\text{effective}}$; (right) the significance of feature components being over-fitted $\alpha^{(l)}_{\text{overfit}}$, between DNNs learned using various training samples.

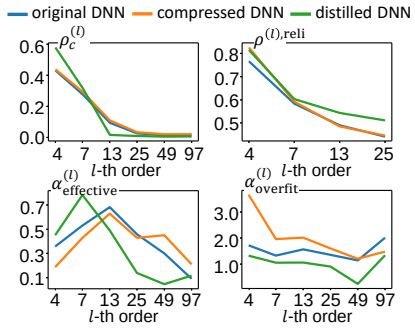

Figure 7. Comparisons between the original DNN, the compressed DNN, and the distilled DNN.

cation accuracy of DNNs by directly replacing the original target feature $f(x)$ with $\Phi^{(l)}(x)$. Figure 5 (left) shows the accuracy improvement using $\Phi^{(l)}(x)$ when $l = 7$. When we used the feature component with $l = 7$, the classification accuracy of ResNet-14 learned on the CIFAR-10 dataset was improved by over 5%. Beyond the existing finding that knowledge distillation can improve DNNs (Liang et al., 2019), results in Figure 6 and Figure 5 (right) explicitly specified the middle-complexity feature components were more responsible for the accuracy increase than high-complexity features. Figure 5 (right) shows that for different complexity orders, the accuracy improvement was different. When the complexity order l increased, the accuracy improvement first increased, and then decreased. This was because very few feature components of the lowest complexity did not contain enough information for the classification, while high-complexity feature components were significantly over-fitted, which hurt the generalization power of DNNs.

Exp. 3, analysis of network compression and knowledge distillation. We learned the ResNet-32 on the CIFAR-10 dataset with 1000 training samples as the original DNN. We used the compression algorithm (Han et al., 2015) to learn another DNN (termed the *compressed DNN*) by pruning and quantizing the original DNN. For the knowledge distillation, we used another network (termed the *distilled DNN*)[4], to distill (Hinton et al., 2015) the output feature of the last residual block in the original DNN. The supplementary material summarizes technique details of network compression and knowledge distillation provided in (Han et al., 2015; Hinton et al., 2015). We compared the compressed DNN and the distilled DNN with the original DNN. We decomposed feature components from the output feature of the last residual block in the original DNN and the compressed DNN, and the output feature of the distilled DNN.

[4]The distilled DNN had the same architecture with the decomposer net with 7 ReLU layers.

Figure 7 shows $\rho^{(l)}_c$, $\rho^{(l),\text{reli}}$, $\alpha^{(l)}_{\text{effective}}$, and $\alpha^{(l)}_{\text{overfit}}$ in three DNNs. For the compressed DNN, **(1)** the network compression did not affect feature components' strength distribution and reliability. **(2)** Low-complexity feature components in the compressed DNN exhibited lower effectiveness and higher significance of over-fitting than low-complexity feature components in the original DNN.

For the knowledge distillation, **(1)** the distilled DNN had more low-complexity feature components than the original DNN. The low-complexity feature components in the distilled DNN were more effective than those in the original DNN. **(2)** High-complexity feature components in the distilled DNN were more reliable and less over-fitted than high-complexity feature components in the original DNN. These results demonstrated that the knowledge distillation would help DNNs learn more reliable features, which prevented over-fitting.

Exp. 4, the close relationship between the feature complexity and the performance of DNNs. Inspired by Figure 7, we thought there was a close relationship between the feature complexity and the performance of DNNs.

To this end, we learned a regression model, which used the distribution of feature components' strength over different complexity orders to predict the performance of DNNs. For each DNN, we used decomposer nets with $l_1 = 4, l_2 = 7, l_3 = 13, l_4 = 25$ to decompose $\Phi^{(l),\text{reli}}(x)$ and $\Phi^{(l),\text{unreli}}(x)$. Then, we calculated $Var[\Phi^{(l_i),\text{reli}}(x) - \Phi^{(l_{i-1}),\text{reli}}(x)]/Var[f(x)]$ and $Var[\Phi^{(l_i),\text{unreli}}(x) - \Phi^{(l_{i-1}),\text{unreli}}(x)]/Var[f(x)]$, thereby obtaining an 8-dimensional feature to represent the distribution of different feature components. In this way, we learned a linear regressor to use the 8-dimensional feature to predict the testing loss or the classification accuracy. For the CIFAR-10 dataset, we applied cross validation: we randomly selected 20 DNNs from 25 pre-trained ResNet-8/14/20/32/44 models on different training sets in Exp. 1 to learn the re-

	Predicting the accuracy		Predicting the loss	
	Prediction error	Range of value	Prediction error	Range of value
CIFAR-10	**2.73%**	28.73%-72.83%	**0.49**	1.59-6.42
CUB200-2011	**5.66%**	28.18%-56.18%	**0.47**	2.94-5.76
Stanford Dogs	**3.26%**	9.37%-37.95%	**0.34**	4.34-7.97

Table 1. Verifying the close relationship between the feature complexity and the performance of DNNs. The mean error of using the feature complexity to predict the classification accuracy and the classification loss was reported. The prediction error was much less than the range of the testing accuracy and the range of the loss.

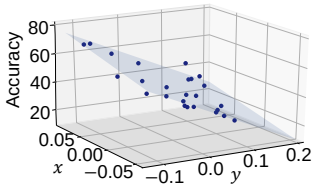

Figure 8. Relationship between the feature complexity and the accuracy on the CIFAR-10 dataset.

gressor and used the other 5 DNNs for testing.[5] These 25 DNNs were learned using 200-5000 samples, which were randomly sampled from the CIFAR-10 dataset to boost the model diversity. We repeated such experiments for 1000 times for cross validation.

Table 1 reports the mean prediction error for the classification accuracy and the task loss over 1000 repeated experiments. The prediction error was much less than the range of the testing accuracy and the range of the classification loss, which indicated the strong relationship between the distribution of feature complexity and the performance of DNNs.

Figure 8 further visualizes the plane of the linear regressor learned on the CIFAR-10 dataset. The visualization was conducted by using PCA (Wold et al., 1987) to reduce the 8-dimensional feature into a 2-dimensional space, *i.e.* (x, y) in Figure 8. There was a close relationship between the distribution of feature complexity and the performance of a DNN.

5. Conclusion

In this paper, we have proposed a generic definition of the feature complexity of DNNs. We design a method to decompose and visualize feature components of different complexity orders, and analyze feature components from three perspectives. Then, a close relationship between the feature complexity and the performance of DNNs is discovered. Furthermore, the feature components can improve the classification accuracy of DNNs. As a generic tool, the feature complexity provides a new perspective to explain existing deep-learning techniques, which has been validated by experiments.

Acknowledgments

This work is partially supported by the National Nature Science Foundation of China (No. 61906120, U19B2043), and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

[5]For the CUB200-2011 dataset and the Stanford Dogs dataset, we randomly selected 11 models from 12 pre-trained ResNet-18/34 and VGG-16 models to learn the regressor. One model was used for testing.

References

Achille, A. and Soatto, S. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018a.

Achille, A. and Soatto, S. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2897–2905, 2018b.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Bianchini, M. and Scarselli, F. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

Blum, A. and Rivest, R. L. Training a 3-node neural network is np-complete. In *Advances in neural information processing systems*, pp. 494–501, 1989.

Boob, D., Dey, S. S., and Lan, G. Complexity of training relu neural network. *arXiv preprint arXiv:1809.10787*, 2018.

Chattopadhay, A., Sarkar, A., Howlader, P., and Balasubramanian, V. N. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847. IEEE, 2018.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 882–891, 2018.

Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. L-shapley and c-shapley: Efficient model interpretation for structured data. *In ICLR*, 2019.

Cheng, X., Rao, Z., Chen, Y., and Zhang, Q. Explaining knowledge distillation by quantifying the knowledge. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12925–12935, 2020.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 874–883. JMLR. org, 2017.

Dosovitskiy, A. and Brox, T. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4829–4837, 2016.

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

Fort, S., Nowak, P. K., and Narayanan, S. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Ghorbani, A. and Zou, J. Data shapley: Equitable valuation of data for machine learning. *In ICML*, 2019.

Goldfeld, Z., Van Den Berg, E., Greenewald, K., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. Estimating information flow in deep neural networks. In *International Conference on Machine Learning*, pp. 2299–2308, 2019.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *In ICLR*, 2(5):6, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., and Zhang, H. Sgd on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems 32*, pp. 3496–3506. Curran Associates, Inc., 2019.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Liang, R., Li, T., Li, L., and Zhang, Q. Knowledge consistency between neural networks and beyond. In *International Conference on Learning Representations*, 2019.

Liang, T., Poggio, T., Rakhlin, A., and Stokes, J. Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Livni, R., Shalev-Shwartz, S., and Shamir, O. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pp. 855–863, 2014.

Lopez-Paz, D., Bottou, L., Schölkopf, B., and Vapnik, V. Unifying distillation and privileged information. *In ICLR*, 2016.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Manurangsi, P. and Reichman, D. The computational complexity of training relu (s). *arXiv preprint arXiv:1810.04207*, 2018.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Phuong, M. and Lampert, C. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pp. 5142–5151, 2019.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Dickstein, J. S. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2847–2854. JMLR. org, 2017.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Rolnick, D. and Tegmark, M. The power of deeper networks for expressing natural functions. *arXiv preprint arXiv:1705.05502*, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2017.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. 2011.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

Williamson, B. D. and Feng, J. Efficient nonparametric statistical inference on population feature importance using shapley values. *In ICML*, 2020.

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

Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

Xu, A. and Raginsky, M. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2524–2533, 2017.

Xu, Z. J. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhang, S., Wu, Y., Che, T., Lin, Z., Memisevic, R., Salakhutdinov, R. R., and Bengio, Y. Architectural complexity measures of recurrent neural networks. In *Advances in neural information processing systems*, pp. 1822–1830, 2016.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Object detectors emerge in deep scene cnns. *In ICLR*, 2015.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.