QUANTIFYING LAYERWISE INFORMATION DISCARDING OF NEURAL NETWORKS AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents a method to explain how input information is discarded through intermediate layers of a neural network during the forward propagation. The layerwise analysis of information discarding is used to explain and diagnose various deep-learning techniques. We define two types of entropy-based metrics, *i.e.* the strict information discarding and the reconstruction uncertainty, which measure input information of a specific layer from two perspectives. We develop a method to compute entropy-based metrics, which ensures the fairness of comparisons between different layers of different networks. Preliminary experiments have shown the effectiveness of our metrics in analyzing benchmark networks and explaining existing deep-learning techniques. *The code will be released when the paper is accepted.*

1 INTRODUCTION

Deep neural networks (DNNs) have shown a significant discrimination capacity in many tasks. However, black-box feature representations of a DNN have increased difficulties of analyzing the correctness of the DNN, which has presented continuous challenges for decades.

Therefore, in this study, we aim to propose generic metrics to help diagnose intermediate-layer features in DNNs and explain the success of existing deep-learning techniques. Specifically, given a pre-trained DNN, we analyze features in intermediate layers from the new perspective, *i.e.* information propagation. Information propagation through cascaded layers of a DNN can be considered as a process of feature selection. Without loss of generality, we take convolutional neural networks for image classification as examples to help simplify the introduction. The signal-processing logic of a DNN can be roughly understood as follows: non-linear layers are learned to pass neural activations of task-related visual concepts to the next layer and remove neural activations of unrelated visual concepts (Wolchover, 2017).

In this way, we propose to quantify information of the input after propagating through intermediate layers of the DNN. The quantification of the information of each layer can be used as a new metric to examine a DNN, besides the traditional metric of the testing accuracy. In general, this research includes the following two tasks.

(1) *Quantification of the input information*: In this paper, knowledge representations of a certain layer are referred to as the amount of the input information that has been passed to this layer. Note that a DNN keeps discarding information of the input during the forward propagation of neural activations (Goldfeld et al., 2019; Schwartz-Ziv & Tishby, 2017). Information discarding of intermediate layers is measured and shown in Fig. 1.

(2) *Evaluation of the utility of DNNs and existing deep-learning methods*: We use information discarding as generic tools to evaluate representation capacity of DNNs, analyze the effectiveness of network compression and knowledge distillation, and diagnose architectural flaws in DNNs.

Above tasks of quantitative network diagnosis have extremely high requirements of generality and coherency for the evaluation metric (the definition of the generality and coherency will be introduced later). This makes our research essentially different from traditional visualization of neural networks. Previous research usually visualizes image appearance corresponding to a feature map or the network output (Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox,

2016). Other studies extract pixels/regions in the input image that are highly correlated to the network output (Ribeiro et al., 2016; Lundberg & Lee, 2017; Kindermans et al., 2018). However, traditional pixel-level saliency is usually estimated using heuristic assumptions, which lead to issues in generality and coherency (see Fig. 1(top-left)).

Therefore, to overcome the lack of generality and coherency, we design the following two types of metrics to quantify the information discarding in intermediate layers. We track the layerwise information discarding during the forward propagation to evaluate DNNs.

(1) **Strict information discarding (SID) & concentration → how much input information is used to compute the feature:** Considering the redundancy of the input information, a DNN usually selectively discards information of certain input units (pixels) to compute the intermediate-layer feature. In general, there are two reasons for information discarding. First, some pixels are not related to the task, *e.g.* those on the background. Second, input information is redundant. For example, neighboring pixels in an image usually have similar colors to represent the same superpixel. The measurement of the layerwise information discarding was originally used in (Guan et al., 2019) to estimate word attributions in tasks of natural language processing. We reformulate the information discarding as SID to boost the fairness of layerwise comparisons.

Furthermore, the concentration metric measures the relative magnitude of information discarding on the foreground *w.r.t.* that on the background to evaluate the effectiveness of information processing.

(2) **Reconstruction uncertainty (RU) → how much input information can be recovered by the feature:** As mentioned above, certain pixels may be discarded during the computation of intermediate-layer features, but their information can still be well recovered by other pixels due to the information redundancy. Thus, we propose another metric, *i.e.* RU, to quantify the information discarding from the perspective of inverting features back to the input.

Theoretically, both SID and RU can be modeled as specific measurements of the entropy of input information, which is contained in the feature of an intermediate layer. More specifically, the SID is related to the compactness of knowledge representation in the model and robustness to adversarial attacking (see Appendix A). In comparison, the RU measures the information discarding with prior knowledge of the low-dimensional manifold of input data. *I.e.* the RU reconstructs the input image without considering samples outside the manifold of real data.

Then, we will discuss issues of generality and coherency in the diagnosis of DNNs.

- *Generality* refers to the problem that existing methods of explaining DNNs are usually based on heuristic assumptions, specific network architectures, or specific tasks. To this end, both SID and RU are formulated in the form of the entropy. As a generic mathematical tool, the entropy is a standard metric with strong connections to existing information theories (Wolchover, 2017; Schwartz-Ziv & Tishby, 2017). In comparison, previous pixel-level attribution based on heuristic assumptions (*e.g.* gradient-based methods (Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015), perturbation-based methods (Fong & Vedaldi, 2017; Kindermans et al., 2018), and inversion-based methods (Dosovitskiy & Brox, 2016)) are not defined using generic mathematical concepts.

- *Coherency:* As generic metrics, the SID and RU are agnostic to both the network architecture and the task. Thus, theoretically, they ensure the fairness of comparisons of knowledge representations (1) between neural networks learned for different tasks, (2) between different network architectures for the same task, and (3) between different layers of the same neural network. Please see Section 4 for experimental proof.

Such comprehensive comparisons ensure the broad applicability of this study. We use metrics of SID and RU to evaluate the representation capacity of various DNNs, *e.g.* diagnosing DNNs learned via network compression and knowledge distillations.

Contributions of this study can be summarized as follows. (1) In this study, we propose to measure the discarding of input information during the forward propagation, in order to diagnose layerwise knowledge representations of pre-trained DNNs. (2) We define two generic types of information discarding and develop a method to quantify the information discarding, which enables fair comparisons of knowledge representations over different layers of different DNNs. (3) The proposed metrics help diagnose DNNs and analyze existing deep-learning techniques. Preliminary experiments have demonstrated the effectiveness of our method.

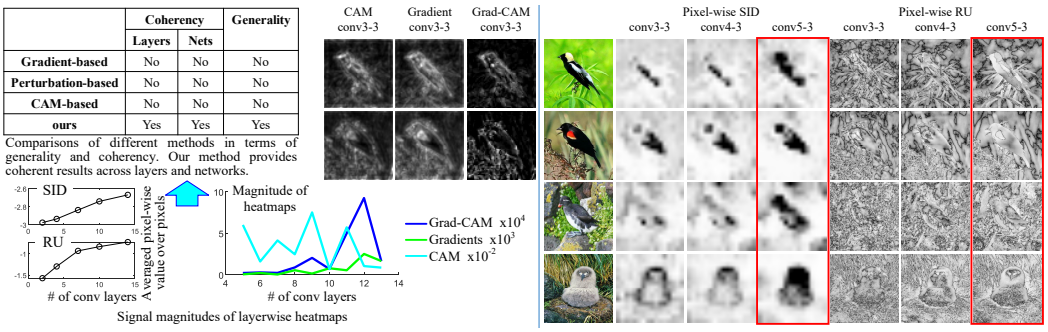


Figure 1: Coherency of our metrics. We show saliency maps of intermediate-layer features yielded by different baselines. (bottom left) We also compare layerwise magnitudes of different results. Our metrics measure how input information is gradually discarded through layers. (top left) Our metrics provide consistent measures of information discarding that enabled faithful comparisons over layers and networks. (right) We visualize pixel-level SID and RU using $H_i(\sigma_i)$ and $\hat{H}_i(\boldsymbol{\sigma})$, respectively.

2 Related work

In this section, we limit our discussion to the literature of interpreting feature representations of DNNs. In general, previous studies can be roughly classified into following three types.

Explaining DNNs visually or semantically: The visualization of DNNs is the most direct way of explaining knowledge hidden inside a DNN, which include gradient-based visualization (Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015) and inversion-based visualization (Dosovitskiy & Brox, 2016). (Zhou et al., 2015) computed the actual image-resolution receptive field of neural activations in a feature map of a convolutional neural network (CNN). Based on (Zhou et al., 2015), six types of semantics were defined to explain intermediate-layer features of CNNs (Bau et al., 2017; Zhou et al., 2018).

Beyond visualization, some methods diagnose a pre-trained CNN to obtain insight understanding of CNN representations. Fong and Vedaldi (Fong & Vedaldi, 2018) analyzed how multiple filters jointly represented a specific semantic concept. (Selvaraju et al., 2017), (Fong & Vedaldi, 2017), and (Kindermans et al., 2018) estimated image regions that directly contribute the network output. The LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) assumed a linear relationship between the input and output of a DNN to extract important input units.

However, previous studies were usually developed based on heuristic assumptions, which hurt their generality and coherency. For example, many visualization methods assumed gradients on features reflected the importance of the feature, which had been disputed by (Lundberg & Lee, 2017). In comparison, our metrics are based on the generic concept of the entropy of the input and enable fair comparisons of the representation capacity through different layers of different DNNs.

Learning explainable deep models: Some studies directly learn DNNs with meaningful representations. In the capsule net (Sabour et al., 2017), each output dimension of a capsule may encode a specific meaning. (Zhang et al., 2018) proposed to learn CNNs with disentangled intermediate-layer representations. The infoGAN (Chen et al., 2016) and β-VAE (Higgins et al., 2017) learned interpretable input codes for generative models.

Mathematical evaluation of the representation capacity: Formulating and evaluating the representation capacity of DNNs is another emerging direction. The analysis of representation similarity between DNNs based on canonical correlation analysis is widely used to diagnose DNN representations (Kornblith et al., 2019; Raghu et al., 2017; Morcos et al., 2018). (Novak et al., 2018) measured the sensitivity of network outputs with respect to parameters of neural networks. (Zhang et al., 2017) discussed the relationship between the parameter number and the generalization capacity of DNNs. (Arpit et al., 2017) analyzed the representation capacity of DNNs considering real training data and noises. (Yosinski et al., 2014) evaluated the transferability of filters in intermediate layers. Network-attack methods (Koh & Liang, 2017; Szegedy et al., 2014; Koh & Liang, 2017) can also be used to evaluate representation robustness by computing adversarial samples for a CNN. (Deutsch,

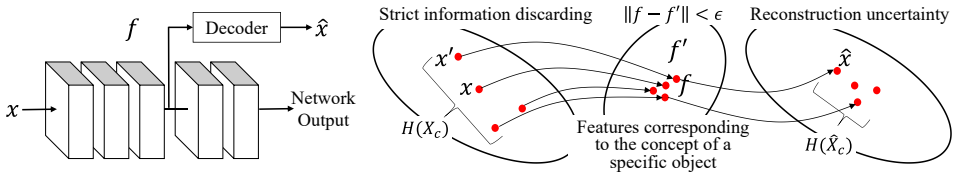


Figure 2: Overview of the algorithm. Given a trained DNN, we compute the maximal entropy of the input $H(X_c)$ and the maximal entropy of image reconstruction $H(\hat{X}_c)$, when we constraint the intermediate-layer feature f within the small range of the concept of a specific object.

2018) learned the manifold of network parameters to diagnose neural networks. Recently, the stiffness (Fort et al., 2019) was proposed to evaluate the generalization of DNNs. (Gotmare et al., 2019) explained knowledge distillation and learning rate heuristics of restarts and warmup.

In particular, the information-bottleneck theory (Wolchover, 2017; Schwartz-Ziv & Tishby, 2017) provides a generic metric to quantify the information contained in DNNs. The information-bottleneck theory can be extended to evaluate the representation capacity of DNNs (Goldfeld et al., 2019; Xu & Raginsky, 2017; Cheng et al., 2018). (Achille & Soatto, 2018) further used the information-bottleneck theory to revise the dropout layer in a DNN. Our study is also inspired by the information-bottleneck theory. Unlike exclusively analyzing the final output of a DNN in (Cheng et al., 2018), we pursue new model-agnostic and task-agnostic metrics of input information to enable comparisons over different layers and networks.

3 METRICS TO DIAGNOSE FEATURE REPRESENTATIONS OF DNNS

In order to conduct comparative studies to diagnose DNNs learned by various deep-learning techniques, in this section, we introduce three generic metrics (*i.e.* SID, RU, and concentration), as well as their pixel-wise versions for visualization. Theoretically, these metrics can be applied to various tasks, but to simplify the story, we limit our discussions to the task of object classification.

Both the SID and the RU are derived from the entropy of the input information, given the feature of a specific intermediate layer. The concentration is defined based on the SID. Let $x \in \mathbb{R}^n$ and $f = h(x)$ denote the input and an intermediate-layer feature of the DNN, respectively. We assume that the DNN represents the concept of a specific object instance using a very limited range of features F_c. All features in F_c are assumed to represent the same object-instance concept c. For example, the concept of an object x can be represented by a small range of features with the center of f, *i.e.* F_c is defined to satisfy $\mathbb{E}_{f' \in F_c}[\|f' - f\|^2] = \epsilon$, where ϵ is a small constant.

Fig. 2 illustrates the basic idea of the algorithm. We compute the entropy of the input when the input represents the same object-instance concept (*i.e.* the SID). We also use features of the object concept to reconstruct the input $\hat{x} = g(f)$ and measure the entropy of the reconstructed input (*i.e.* the RU). In this way, two types of information discarding (SID and RU) of a specific layer can be represented using the same prototype formulation as follows.

$$H(X_c) = -\sum_{x' \in X_c} p(x') \log p(x'), \qquad \text{s.t.} \quad \mathbb{E}_{f' \in F_c}\left[\|f' - f\|^2\right] = \epsilon, \tag{1}$$

If $x' \in X_c$ represents the raw input x s.t. $f = h(x)$, then $H(X_c)$ indicates the SID; if x' denotes the reconstructed input $\hat{x} = g(f)$ using the feature in F_c, then $H(X_c)$ measures the RU.

3.1 METRICS OF STRICT INFORMATION DISCARDING & CONCENTRATION

The metric of SID is derived from the entropy in Equation (1) from the perspective of feature extraction $f = h(x)$. The SID quantifies the discarding of input pixels (units) during the computation of intermediate-layer features.

The core challenge is that the explicit low-dimensional manifold F_c of features *w.r.t.* the input x is unknown. Therefore, we approximate F_c by adding noises to the original input x to generate new inputs x' around x, *i.e.* $x' = x + \boldsymbol{\delta}$, which satisfy $\mathbb{E}_{f'=h(x')}[\|f' - f\|^2] = \epsilon$. We assume that $\boldsymbol{\delta}$ is a Gaussian noise, so the distribution of x' follows the Gaussian distribution $x' \sim \mathcal{N}(\mu, \Sigma)$.

Considering the local linearity within a small feature range of ϵ and $f = h(x)$, the mean value μ can be approximated as $\mu = x$. We further simplify the covariance matrix as a diagonal matrix $\Sigma = diag[\sigma_1^2, \ldots, \sigma_n^2]$ to ease the computation. In this way, the entropy of the SID can be decomposed to pixel-wise entropy.

$$H(X_c) = \sum_{i=1}^{n} H_i(\sigma_i), \qquad H_i(\sigma_i) = \log \sigma_i + C, \qquad \text{s.t. } \mathbb{E}_{f'=h(x'):\, x'=x+\boldsymbol{\delta}\in X_c}\left[\|f' - f\|^2\right] = \epsilon \tag{2}$$

Here, the overall SID value $H(X_c)$ can be decomposed to the pixel-level entropy (**pixel-level SID**) $\{H_i(\sigma_i)\}$, and $i = 1, \ldots, n$ denotes the index of each pixel. $C = \frac{1}{2}\log(2\pi e)$. We can use the map of pixel-wise SID $H_i(\sigma_i)$ to visualize the discarding of the information of each input pixel (see Fig. 1).

Our method follows the **maximum-entropy principle**, which maximize $H(X_c)$ subject to constraining features f' within the scope of a specific concept $\mathbb{E}_{f'=h(x'):x'\in X_c}[\|f' - f\|^2] = \epsilon$. ***I.e.*** we enumerate all input images x' in all perturbation directions within a small variance of features f', in order to approximate the local manifold of intermediate-layer features. Considering Lagrange multipliers, we can further relax the constraint on the range of ϵ and design the following loss.

$$\begin{aligned}
Loss(\boldsymbol{\sigma}) &= \frac{1}{\delta_f^2}\mathbb{E}_{f'}\left[\|f' - f\|^2\right] - \lambda\sum_{i=1}^{n} H_i(\sigma_i) = \frac{1}{\delta_f^2}\mathbb{E}_{f'}\left[\|f' - f\|^2\right] - \lambda\sum_{i=1}^{n}\left(\log\sigma_i + C\right) && (3)\\
&= \frac{1}{\delta_f^2}\mathbb{E}_{x'\sim\mathcal{N}(\mu=x,\Sigma)}\left[\|h(x') - f\|^2\right] - \lambda\sum_{i=1}^{n}\left(\log\sigma_i + C\right) && (4)\\
&= \frac{1}{\delta_f^2}\mathbb{E}_{\substack{x'=x+\boldsymbol{\sigma}\circ\boldsymbol{\delta}:\\ \boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\Sigma)}}\left[\|h(x') - f\|^2\right] - \lambda\sum_{i=1}^{n}\left(\log\sigma_i + C\right), && (5)
\end{aligned}$$

where $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_n]^\top$. The first term constricts the range of the feature. $\delta_f^2 = \lim_{\tau\to 0^+}\mathbb{E}_{x'\sim\mathcal{N}(x,\tau^2\mathbf{I})}\left[\|h(x') - f\|^2\right]/\tau^2$ is computed by averaging over all images to normalize the activation magnitude. The second term boosts the entropy $H(X_c)$. λ is a positive scalar. Note that the original loss in Equation (4) is intractable. We use $x' = x + \boldsymbol{\sigma}\circ\boldsymbol{\delta}$ to simplify the computation of the gradient *w.r.t.* $\boldsymbol{\sigma}$, where $\circ$ denotes the element-wise multiplication.

For fair layerwise comparisons: In order to ensure coherent layerwise comparisons, we need to control the value range of the first term in Equation (5). Features of different layers need to represent similar ranges of variations, ***i.e.*** features of each specific layer need to be perturbed at a comparable level. To this end, we use δ_f^2 to normalize the first term in Equation (5). δ_f^2 denotes the inherent variance of intermediate-layer features subject to a small input noise. In addition, for each specific layer, we measure and compare pixel-level SID $H_i(\sigma_i)$ when $\mathbb{E}_{f'}[\|f' - f\|^2] = \epsilon = \alpha\delta_f^2$, where α is a positive scalar. The value of λ is slightly adjusted (manually or automatically) to make the learned $\boldsymbol{\sigma}$ satisfy $\mathbb{E}_{f'}[\|f' - f\|^2] \approx \alpha\delta_f^2$.

Concentration of information discarding (termed "concentration" for short): Based on the SID, we design the metric of the concentration to evaluate the efficiency of feature extraction of DNNs that are learned for object classification. Given an input image x containing both the target object and some background area, let Λ denote the ground-truth segment (or the bounding box) of the target object in x. $\forall i \in \Lambda, x_i$ is given to represent pixels within Λ of the target object. Thus, the concentration is formulated as

$$\mathbb{E}_{i\notin\Lambda}\left[H_i(\sigma_i)\right] - \mathbb{E}_{i\in\Lambda}\left[H_i(\sigma_i)\right] \tag{6}$$

Ideally, a DNN for object classification is supposed to discard background information, rather than foreground information. Thus, the concentration measures the relative background information discarding *w.r.t.* foreground information discarding, which reflects the efficiency of feature extraction.

3.2 Metrics of reconstruction uncertainty

The metric of RU is also derived from the entropy in Equation (1). The SID focuses on the input information used to compute an intermediate-layer feature, while the RU describes the discarding of input information that can be recovered from the feature. Due to the redundancy of the input information, an input pixel may be well recovered by the feature, even when the pixel is not used for feature extraction.

We use a decoder net $\hat{x}' = g(f')$ to reconstruct the input. We consider the reconstructed result $\hat{x}'$ as the information represented by f'. Although the architecture of g affects the measurement of

RU, RU values are still comparable through different layers and between DNNs when we fix g's architecture in all comparisons (see Fig. 1). Given a target DNN, g is pre-trained using the MSE loss $Loss^{\text{decoder}} = \|x' - \hat{x}'\|^2$. In this way, the RU is formulated as the entropy of the reconstructed result $\hat{x}' = g(f')$.

$$H(\hat{X}_c) = -\sum_{\hat{x}'=g(f'):f'\in F_c} p(\hat{x}')\log p(\hat{x}'), \qquad \text{s.t.} \quad \mathbb{E}_{f'\in F_c}\left[\|f'-f\|^2\right] = \epsilon \tag{7}$$

where $\hat{X}_c$ denotes a set of features that are reconstructed using intermediate-layer features.

The above entropy $H(\hat{X}_c)$ can be computed in the same manner as the quantification of the SID. First, we synthesize the feature distribution F_c by assuming that inputs follow a Gaussian distribution $x' \sim \mathcal{N}(\mu = x, \Sigma)$. $\hat{x}' = g(h(x'))$ denotes the reconstructed result using x'. Second, we can also assume $\hat{x}'$ follows a Gaussian distribution with i.i.d. random variables $\mathcal{N}(\mu^{\text{rec}} = x, \Sigma^{\text{rec}})$. As a result, the entropy of RU $H(\hat{X}_c)$ can be decomposed into each pixel.

$$H(\hat{X}_c) = \sum_{i=1}^{n} \hat{H}_i(\boldsymbol{\sigma}), \qquad \hat{H}_i(\boldsymbol{\sigma}) = \log\hat{\sigma}_i + C = \frac{1}{2}\log\left(\mathbb{E}_{x'\sim\mathcal{N}(\mu=x,\Sigma=diag[\sigma_1,\sigma_2,\ldots])}\left[\|\mu_i^{\text{rec}} - \hat{x}_i'\|^2\right]\right) + C \tag{8}$$

$\hat{H}_i(\boldsymbol{\sigma})$ is referred to as the **pixel-level RU** for the i-th pixel (unit) in the input (see Fig 1). Just like the SID, $H(\hat{X}_c)$ is also estimated via the maximum-entropy principle, as follows.

$$Loss(\boldsymbol{\sigma}) = \frac{1}{\delta_f^2}\mathbb{E}_{f'}\left[\|f'-f\|^2\right] - \lambda\sum_{i=1}^{n}\hat{H}_i(\boldsymbol{\sigma}) \tag{9}$$

$$= \frac{1}{\delta_f^2}\mathbb{E}_{f'}\left[\|f'-f\|^2\right] - \frac{\lambda}{2}\sum_{i=1}^{n}\left\{\log\left(\mathbb{E}_{x'\sim\mathcal{N}(\mu=x,\Sigma)}\left[\|\mu_i^{\text{rec}} - \hat{x}_i'\|^2\right]\right) + C\right\} \tag{10}$$

$$= \frac{1}{\delta_f^2}\mathbb{E}_{\substack{x'=x+\boldsymbol{\sigma}\circ\boldsymbol{\delta}:\\ \boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\Sigma)}}\left[\|h(x')-f\|^2\right] - \frac{\lambda}{2}\sum_{i=1}^{n}\left\{\log\left(\mathbb{E}_{\substack{x'=x+\boldsymbol{\sigma}\circ\boldsymbol{\delta}:\\ \boldsymbol{\delta}\sim\mathcal{N}(\mathbf{0},\Sigma)}}\left[\|\mu_i^{\text{rec}} - \hat{x}_i'\|^2\right]\right) + C\right\} \tag{11}$$

We use the learned $\boldsymbol{\sigma}$ to compute $\hat{H}_i(\boldsymbol{\sigma})$ as the pixel-level RU. The loss in Equation (10) is intractable, so the loss is revised to that in Equation (11) instead. Like the computation of SID, λ is also determined to ensure $\mathbb{E}_{f'}[\|f'-f\|^2] \approx \alpha\delta_f^2$.

3.3 Discussions

Relationship with the information-bottleneck principle: The information-bottleneck theory (Wolchover, 2017; Schwartz-Ziv & Tishby, 2017) proposes $I(F;Y) - \beta I(F;X)$ as a standard metric to analyze DNNs, where X, F, Y denote the input, the feature, and the output of a DNN. This metric has considerable challenges in computation (Alemi et al., 2017; Kolchinsky et al., 2017; Goldfeld et al., 2019). Our metrics are related to the mutual information $I(F;X)$, when we only consider a local range of features. In addition, the SID, RU, and concentration are much easier to compute and enable the pixel-level quantification of information discarding.

	Objective	Feature range for computation	Difficulties of computation
$I(F;X)$ in information bottleneck	Representations of all samples	Considering inputs and features of all samples	Difficult
Strict information discarding	Pixel ignorance in feature extraction	Considering a small range of features *w.r.t.* a single sample	Easier

Relationships between two metrics: The SID and RU are highly related to each other. As mentioned before, redundant pixels ignored by the DNN increase the SID value, but may still be well recovered via the input reconstruction. On the other hand, pixels used for feature extraction may not be restructured. A toy example is that if the feature is computed as $f = h(x) = \sum_i x_i$, then all pixels contribute to the feature extraction, but none of them can be well reconstructed.

Relationship between the SID and the metric in (Guan et al., 2019): (Guan et al., 2019) also measured the entropy of the input information, but there was no quantitative definition for the range of the target concept. In other words, for each intermediate layer, the entropy may be measured within a different range of features, which significantly hurt the coherency in layerwise comparisons. In comparison, we clearly define the feature range $\epsilon = \alpha\delta_f^2$ to enable fair layerwise comparisons.

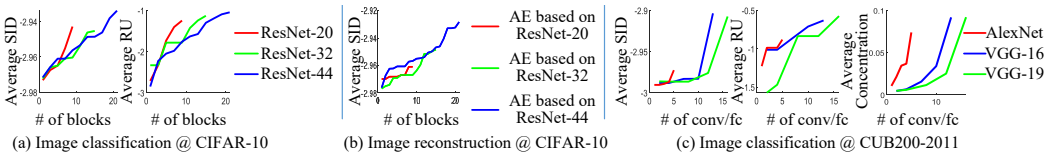


Figure 3: Layerwise strict information discarding, reconstruction uncertainty, and concentration. The values were normalized by the pixel number per image and averaged over all input images.

About information discarding in invertible networks[1]**:** Strictly speaking, there is no strict way to quantify the discarding of the input information during the computation of an intermediate-layer feature. The RU metric is related to image inversion based on invertible nets (Behrmann et al., 2019; andArnold W. M. Smeulders & Oyallon; Kingma & Dhariwal), which both focus on whether the feature can recover the input (theoretically, the decoder g can be implemented as the inversion operations in invertible nets). In comparison, the SID metric is defined from another perspective, *i.e.* whether the input information can contribute significant numerical values to the intermediate-layer feature or the final output. Please see Appendix B for details.

Relationship with perturbation-based methods: Our method is related to (Du et al., 2018; Fong & Vedaldi, 2017). These studies extract input pixels responsible for the intermediate-layer feature by deleting as many input pixels as possible while keeping the feature unchanged. They remove inputs by replacing inputs with human-designed meaningless values. This is heuristic because the designed values are not always meaningless. More crucially, pixel-level saliency computed by (Du et al., 2018; Fong & Vedaldi, 2017) does not correspond to a generic evaluation of representation capacity of DNNs. In comparison, our entropy-based metrics can provide fair comparisons without specific requirements for model parameters, model architecture, and the task.

High SID $\rightarrow$ robustness: Note that a high SID does **NOT** mean lousy feature representations. In contrast, we can regard the forward propagation as a process of discarding irrelevant input information *w.r.t.* the task and maintaining relevant information. Theoretically, features with a high SID may be robust to noises in the input.

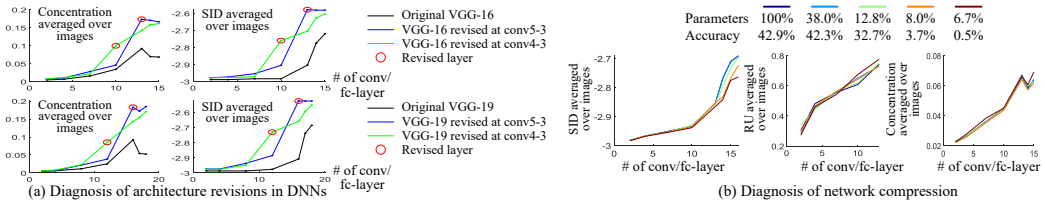
4 Comparative Studies

We designed various experiments, in order to demonstrate the utility of the proposed metrics in comparing feature representations of various DNNs, analyzing flaws of network architectures, and diagnosing inner mechanisms of knowledge distillation and network compression. In experiments, we used our metrics to diagnose nine DNNs, including the AlexNet (Krizhevsky et al., 2012), VGG-16, and VGG-19 (Simonyan & Zisserman, 2015), ResNet-20, ResNet-32, ResNet-44 (He et al., 2016), and auto-encoders[2] based on architectures of ResNet-20, ResNet-32 and ResNet-44 (He et al., 2016). These DNNs were learned using the CIFAR-10 dataset (Krizhevsky, 2009) and the CUB200-2011 dataset (Wah et al., 2011).

In all experiments for image classification, we used object images cropped by object bounding boxes for both training and testing, except for experiments of computing concentration in Fig. 1 where images were cropped by the box of $1.5\,width \times 1.5\,height$ of the object. For the computation of RU, all experiments used a decoder with six residual blocks. To invert low-resolution features back to high-resolution images, we added two transposed conv-layers to two parallel tracks in the residual block to enlarge the feature map. Considering the size of input feature of the decoder, we added transposed conv-layers to the first 2–4 residual blocks. We set $\alpha = 1.5 \times 10^{-4}$ to determine λ in all experiments.

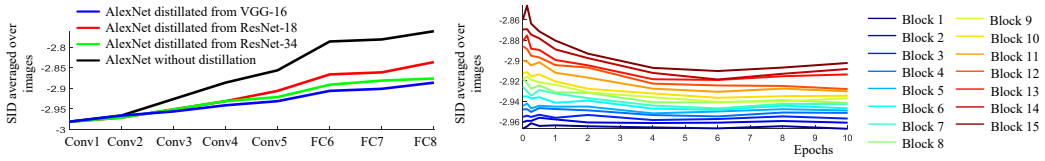
Coherency[1] **for fair comparisons:** Metrics of SID, RU, and concentration reflect the entropy of input information and are defined without strong assumptions of feature representations, network architectures, and the task. Thus, these metrics yield a coherent evaluation of intermediate-layer features and enable fair comparisons over different layers of DNNs. To this end, we compared the proposed metrics with existing visualization techniques, such as the CAM (Zhou et al., 2016), grad-

[1]Please see Appendix B for limitations of the SID considering invertible nets.



(a) Diagnosis of architecture revisions in DNNs (b) Diagnosis of network compression

Figure 4: Diagnosis of architectural revision (a) and network compression (b). Values were normalized by the pixel number per image and averaged over images.



(a) SID (normalized by the pixel number per image) of DNNs learned with and without distillation (b) Change of SID (normalized by the pixel number per image) after different epochs

Figure 5: Effects of knowledge distillation and learning epochs. (a) We compared layerwise information discarding between DNNs learned with and without distillations. (b) Each curve shows information discarding of the output of a specific block during the learning process.

CAM (Selvaraju et al., 2017) and heatmaps of gradients $map = \sqrt{\sum_{i,j}(\frac{\partial y}{\partial f_{ijc}})^2}$, where y denotes the output score of a certain class, and $f \in \mathbb{R}^{M\times M\times C}$ denotes an intermediate-layer feature. A VGG-16 was learned to classify birds based on the CUB200-2011 dataset (Wah et al., 2011). Given a pre-trained DNN, we slightly revised the magnitude of parameters in every pair of neighboring convolutional layers $y = x \otimes w + b$ to examine the coherency of our metrics[1]. For the L-th and $L+1$-th layers, parameters were revised as $w^{(L)} \leftarrow w^{(L)}/4$, $w^{(L+1)} \leftarrow 4w^{(L+1)}$, $b^{(L)} \leftarrow b^{(L)}/4$, $b^{(L+1)} \leftarrow b^{(L+1)}$. Such revisions did not change knowledge representations or the network output[1].

In Fig. 1(bottom-left), our metrics provided consistent and faithful measures, which demonstrated their coherency. Result magnitudes of baseline methods were sensitive to the magnitude of parameters, whereas our metrics produced consistent results, because "knowledge representations" of the layer did not change. Therefore, the coherency of our visualization results enabled layerwise comparisons within a DNN. We also found that edges were usually better reconstructed than textures and colors.

Comparisons between different DNNs for various tasks: We compared layerwise measures of SID and RU of different DNNs. We trained various DNNs for image classification using different datasets and trained auto-encoders (AEs) for image reconstruction (by revising architectures of ResNet-20, ResNet-32, and ResNet-44 (He et al., 2016)[2]). Fig. 3(left,middle) compares input information discarding of intermediate layers of both DNNs for classification and DNNs for reconstruction. We found that image classification and image reconstruction had similar SID values. A deep DNN usually had higher SID, RU, and concentration values than a shallow DNN.

Concentration of information discarding: Fig. 3(right) illustrates the layerwise concentration of various DNNs, which were learned to classify birds in the CUB200-2011 (Wah et al., 2011) dataset. We found that compared to the AlexNet, VGG nets distracted attention to the background to learn diverse features in low layers, but more concentrated on the foreground object in high layers.

Diagnosis of architectural revisions (damage): In this experiment, we aimed to analyze whether the proposed metrics reflected architectural revisions of DNNs. To this end, we revised the architecture of the VGG-16/VGG-19 network by changing a specific convolutional layer to contain 4 $7 \times 7 \times 512$ filters with $padding = 3$, which hurt the representation capacity of the DNN. We trained both the original VGG-16/VGG-19 and the revised VGG-16/VGG-19 for binary classification between birds images cropped from the CUB200-2011 dataset (Wah et al., 2011) and random images in the ImageNet (Deng et al., 2009). Fig. 4(a) compares the original and revised DNNs. We

[2]To construct the auto-encoder, the encoder was set as all layers of the residual network before the FC layer. The decoder was the same as that for the computation of RU.

found that compared to the original DNN, the architectural revision significantly boosted the information discarding at the revised layer. Meanwhile, the architectural revision (damage) also slightly increased of the concentration of DNNs. The increase of concentration seemed to conflict with the architectural damage, but this can be explained as follows. 1. Compared to the increase of the information discarding of the revised net, the increase of concentration was significantly lower. Thus, in general, the architectural revision hurt the representation capacity of the DNN. 2. The DNN with the reduced feature dimension could only encode much fewer concepts of object parts. Thus, the revised DNN usually encoded fewer, simpler, but more discriminative features than original DNNs. 3. Original DNNs usually ignored background information and extract discriminative foreground features at high FC layers (see Fig. 4(a)), whereas the dimension reduction at the revised layer made the DNN ignored background information at much lower layers.

Diagnosis of network compression: We used our metrics to analyze the compressed DNN. We trained another VGG-16 using the CUB200-2011 dataset (Wah et al., 2011) for fine-grained classification. Then, the VGG-16 was compressed using the method of (Han et al., 2016) with different pruning thresholds. Fig. 4(b) compares layerwise information discarding of the original VGG-16 and the compressed VGG-16 nets with different numbers of parameters. We found that network compression decreased the SID of features, which may indicate that the compressed DNN were more sensitive to noises or adversarial attacks (see Appendix A for details). Meanwhile, network compression did not significantly affect the reconstruction capacity and the concentration of intermediate-layer features. It may be also because that parameter reduction in network compression was mainly conducted in the last two fully-connected layers, and most layers (*i.e.* convolutional layers) were affected much.

Analysis of knowledge distillation: We used our metrics to analyze the inner mechanism of knowledge distillation. We trained the VGG-16, ResNet-18, and ResNet-34 using the CUB200-2011 dataset (Wah et al., 2011) as three teacher nets for fine-grained classification. Each teacher net was used to guide the learning of an AlexNet (Krizhevsky et al., 2012). Fig. 5(a) compares layerwise information discarding between AlexNets learned with and without knowledge distillation. We found that AlexNets learned using knowledge distillation had lower information discarding than the ordinarily learned AlexNet. Knowledge distillation helped AlexNets to preserve more information. Meanwhile, knowledge distillation may make intermediate-layer features more sensitive to noises (or adversarial attacking, see Appendix A for details), because AlexNets was mainly learned from distillation and used less noisy information from real training data during the distillation process.

Analysis of information discarding after different epochs during the learning process: We trained the ResNet-32 using the CIFAR-10 dataset (Krizhevsky, 2009). Fig. 5(b) shows the change of information discarding *w.r.t.* outputs of different blocks during the learning process. Information discarding in high layers satisfied the information-bottleneck theory.

5 Conclusion

In this paper, we have defined two metrics to quantify information discarding during the forward propagation. A model-agnostic method is developed to measure the proposed metrics for each specific layer of a DNN. Comparing existing methods of network visualization and extraction of important pixels, our metrics provide consistent and faithful results across different layers. Therefore, our metrics enable fair comparisons over different layers of various DNNs. In preliminary experiments, we have used our metrics to diagnose and understand inner mechanisms of existing deep-learning techniques, which demonstrates the effectiveness of our method.

References

Alessandro Achille and Stefano Soatto. Information dropout: learning optimal representations through noise. *In Transactions on PAMI*, 40(12):2897–2905, 2018.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *In ICLR*, 2017.

Jörn-Henrik Jacobsen andArnold W. M. Smeulders and Edouard Oyallon. i-revnet: Deep invertible networks. *In ICLR*.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. *In ICLR*, 2017.

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

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *In ICML*, 2019.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. *In AAAI*.

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

Hao Cheng, Dongze Lian, Shenghua Gao, and Yanlin Geng. Evaluating capability of deep neural networks for image classification via information plane. *In ECCV*, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *In CVPR*, 2009.

Lior Deutsch. Generating neural networkswith neural networks. *In arXiv:1801.01952*, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. *In ICLR*.

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

Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. Towards explanation of dnn-based prediction with guided feature inversion. *In arXiv:1804.00506*, 2018.

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

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

Stanislav Fort, Pawel Krzysztof Nowak, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *In arXiv:1901.09491*, 2019.

Ziv Goldfeld, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. Estimating information flow in deep neural networks. *In ICML*, 2019.

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. Analyzing inverse problems with invertible neural networks. *In ICLR*.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A close look at deep learning heuristics: Learning rate restarts, warmup and distillation. *In ICLR*, 2019.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. Towards a deep and unified understanding of deep neural models in nlp. *In ICML*, 2019.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *In ICLR*, 2016.

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

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

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *In ICLR*, 2018.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *In NIPS*.

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

Artemy Kolchinsky, Brendan D. Artemy, and David H. Wolpert. Nonlinear information bottleneck. *In arXiv:1705.02436*, 2017.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *In arXiv:1905.00414*, 2019.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. *In Techical report*, 2009.

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

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

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *In NIPS*, 2018.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: An empirical study. *In ICLR*, 2018.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *In NIPS*, 2017.

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

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

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

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

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *In arXiv:1312.6199v4*, 2014.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, In California Institute of Technology, 2011.

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

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *In NIPS*, 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *In NIPS*, 2014.

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

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Undersantding deep learning requires rethinking generalization. *In ICLR*, 2017.

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

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

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

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. *In ECCV*, 2018.

A RELATIONSHIP BETWEEN THE SID VALUE AND THE ADVERSARIAL ROBUSTNESS

In this section, we conducted an experiment to test the relationship between the SID value and the adversarial robustness of the DNN. We followed the settings of the network-compression experiment. We trained another VGG-16 using the CUB200-2011 dataset (Wah et al., 2011) for fine-grained classification. Then, the VGG-16 was compressed using the method of (Han et al., 2016) with different pruning thresholds. The following table compares the SID value of the last FC layer and the adversarial robustness of the DNN, when the DNN was compressed at different ratios.

DNN	SID value	adversarial robustness $\|\epsilon\|_2$
Original DNN (with 100% parameters)	-2.690	0.00276
DNN with 38.0% parameters	-2.693	0.00281
DNN with 12.8% parameters	-2.695	0.00254
DNN with 8.0% parameters	-2.723	0.00109

where for each input image, we computed adversarial samples to its top-20 incorrect fine-grained categories with the highest probabilities. For each adversarial perturbation, we measured its L-2 norm value, and the adversarial robustness was reported as the average value over all images and all adversarial perturbations. Note that we only measured the SID value of the last FC layer (*i.e.* the *fc-8* layer in the VGG-16 network), instead of using SID values of intermediate layers, because the SID value of the last layer most fit the logic of the final prediction of the DNN. We found that the DNN with 8% parameters had much lower SID value than the other three DNNs, and the adversarial robustness of the DNN with 8% parameters was much weaker than other DNNs. This indicated the close relationship between the SID value and the adversarial robustness.

B ABOUT HOW TO UNDERSTAND THE LIMITATION OF SID FROM THE PERSPECTIVE OF INVERTIBLE NETS

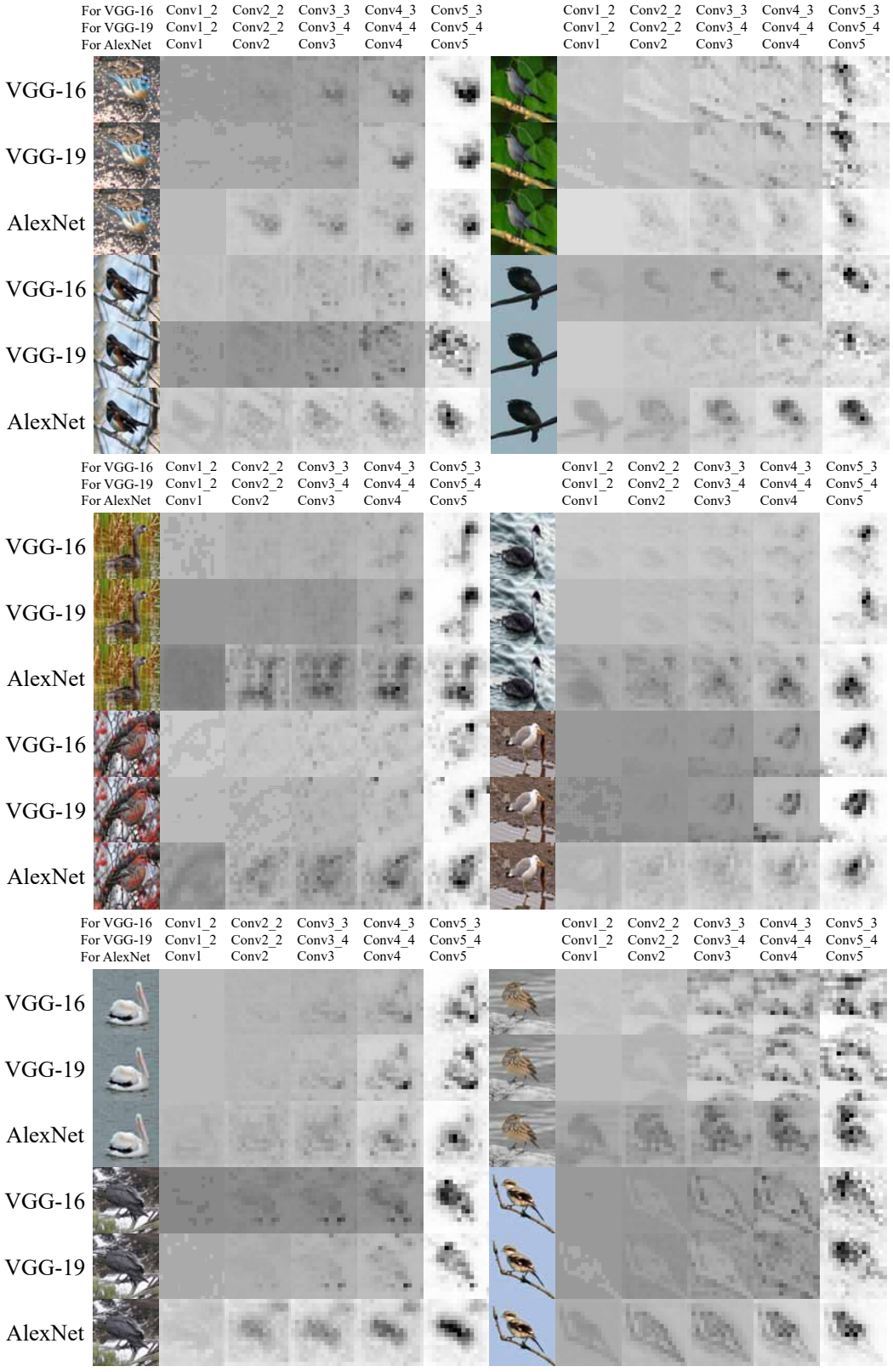
Strictly speaking, there is no strict way to quantify the discarding of the input information during the computation of an intermediate-layer feature. Our method is based on the assumption that the concept of a specific object instance is within the range of $\mathbb{E}_{f' \in F_c}[\|f' - f\|^2] = \epsilon$, which makes the algorithm sensitive to the activation magnitude of each feature dimension. For example, a typical failure case for this assumption is invertible neural networks (Behrmann et al., 2019; Chang et al.; Gomez et al.; andArnold W. M. Smeulders & Oyallon; Kingma & Dhariwal; Dinh et al.). Theoretically, invertible neural networks do not discard any input information; otherwise, the input cannot be inverted from intermediate-layer features. Instead, invertible neural networks usually significantly decrease the magnitude of neural activations caused by unimportant pixels *w.r.t.* the task, and boost the magnitude of neural activations triggered by important pixels *w.r.t.* the task. Similarly, given a pre-trained DNN, if we revise a DNN by selectively halving magnitudes of parameters of 50% filters $w \leftarrow 0.5w$, theoretically, this revision does not discard any input information.

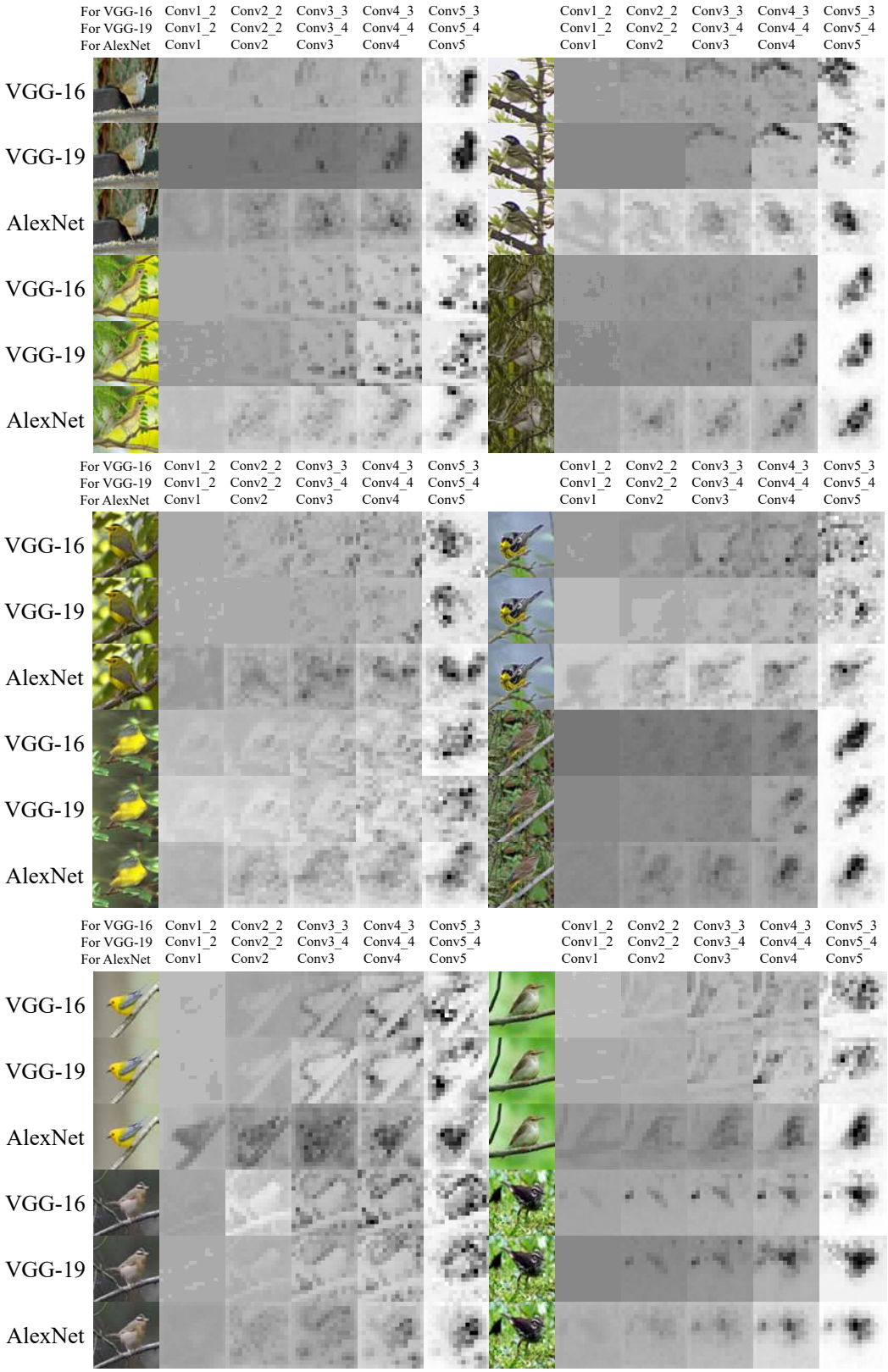
However, information discarding in this paper is defined from another perspective, *i.e.* whether the input information can significantly contribute to the final output of the neural network. For both invertible neural networks and the above revision of halving magnitudes of parameters, these techniques all decrease activation magnitudes caused by certain pixels, thereby letting these pixels contribute less numerical values to the network output.

Therefore, our definition of information discarding does not conflict with the information processing in invertible neural networks. Based on our definition of information discarding, a high information discarding of a pixel indicates that this pixel will contribute a low numerical component to the intermediate-layer feature or the final output.

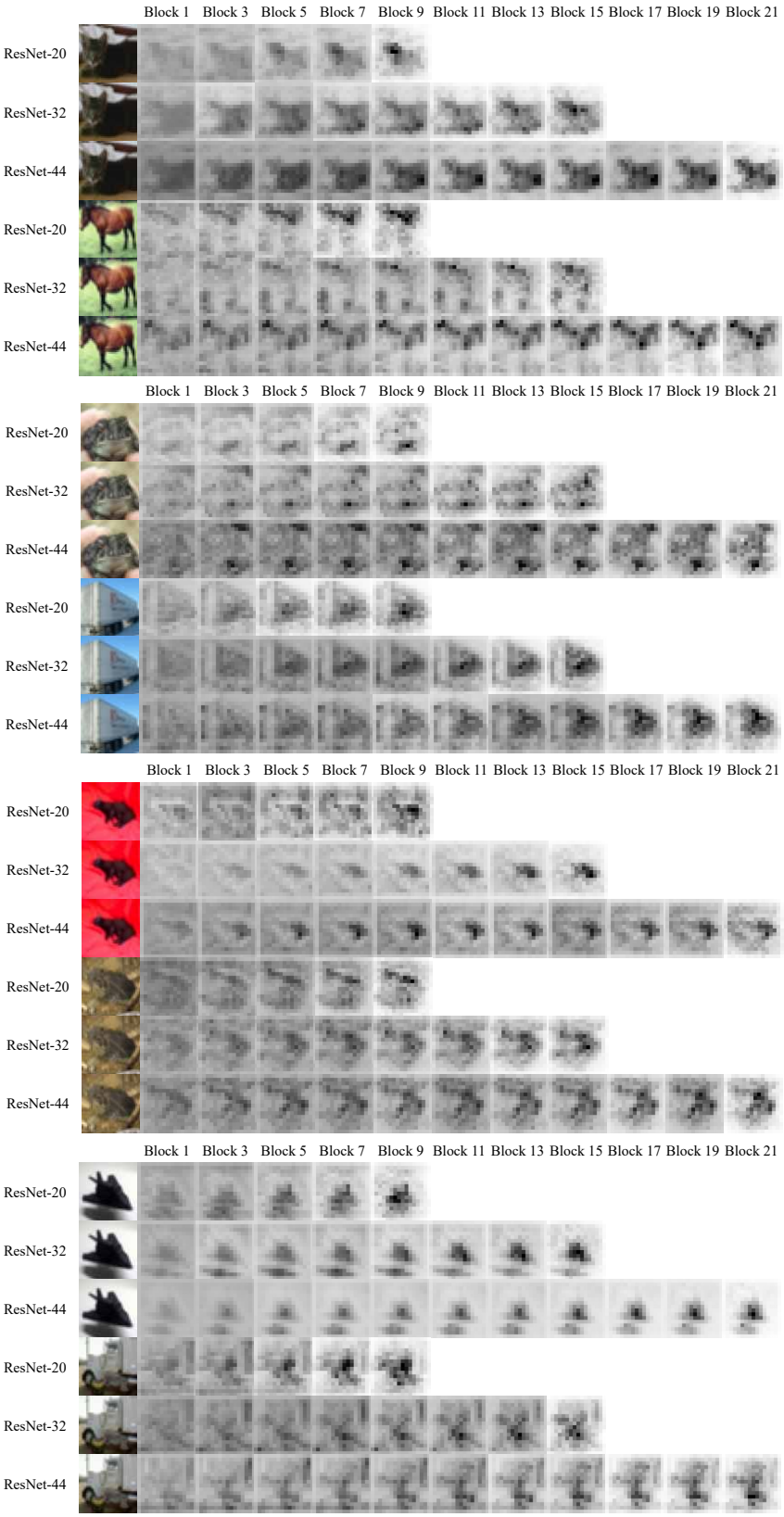
C VISUALIZATION OF PIXEL-WISE SID

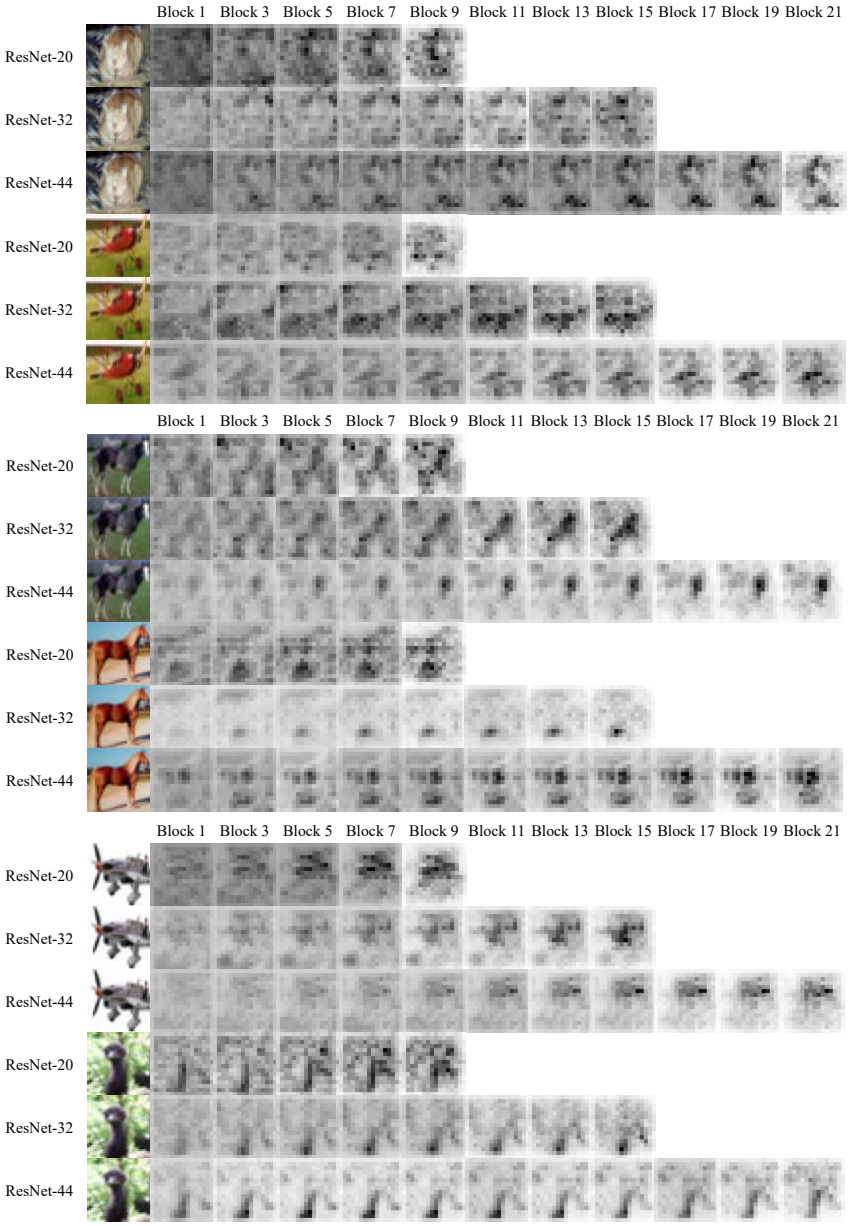
C.1 FOR THE VGG-16 LEARNED USING THE CUB200-2011 DATASET




For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5
Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet
For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5
Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet
For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5
Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet

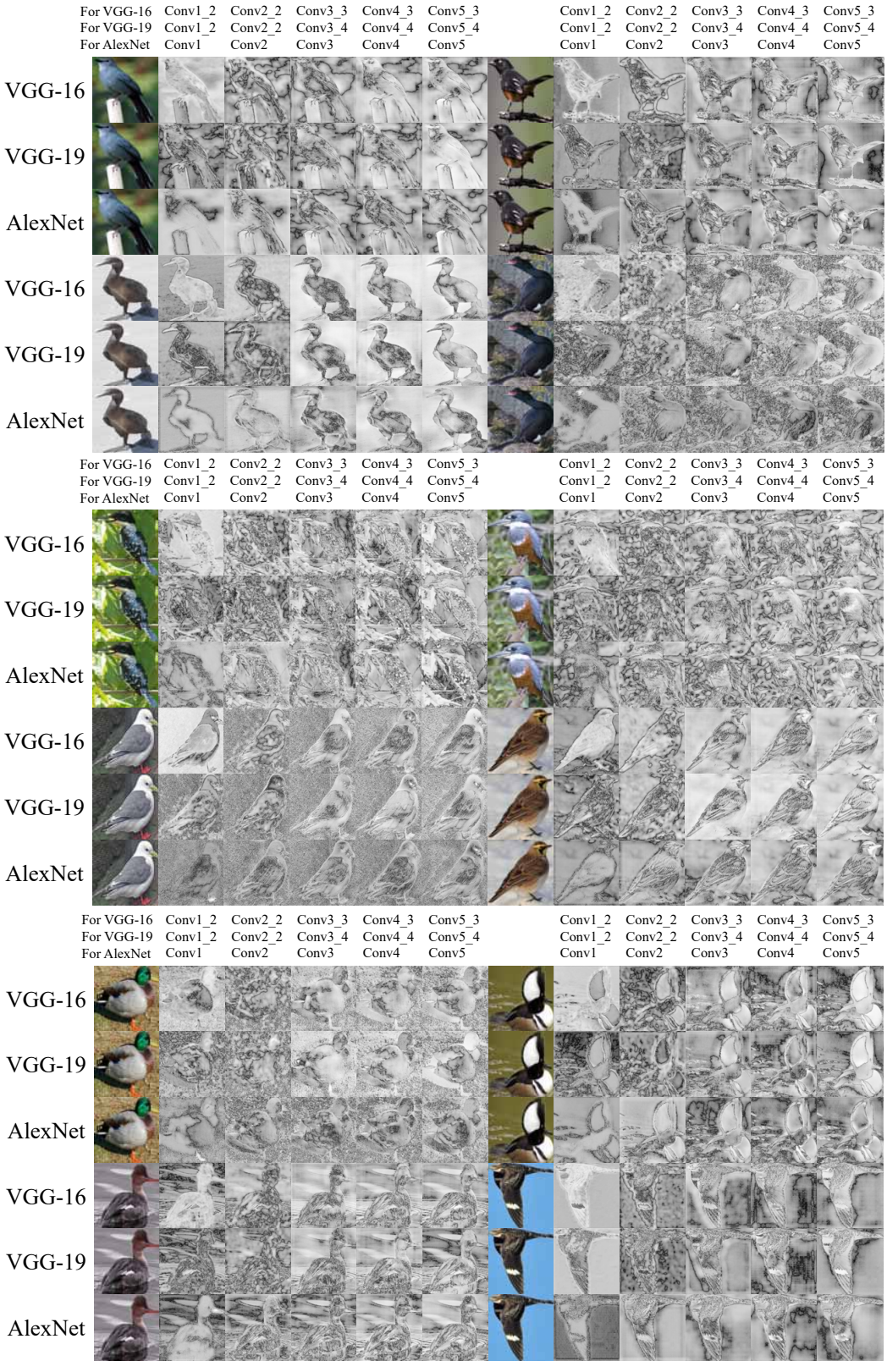
C.2 FOR THE RESNET-20/32/44 LEARNED USING THE CIFAR-10 DATASET

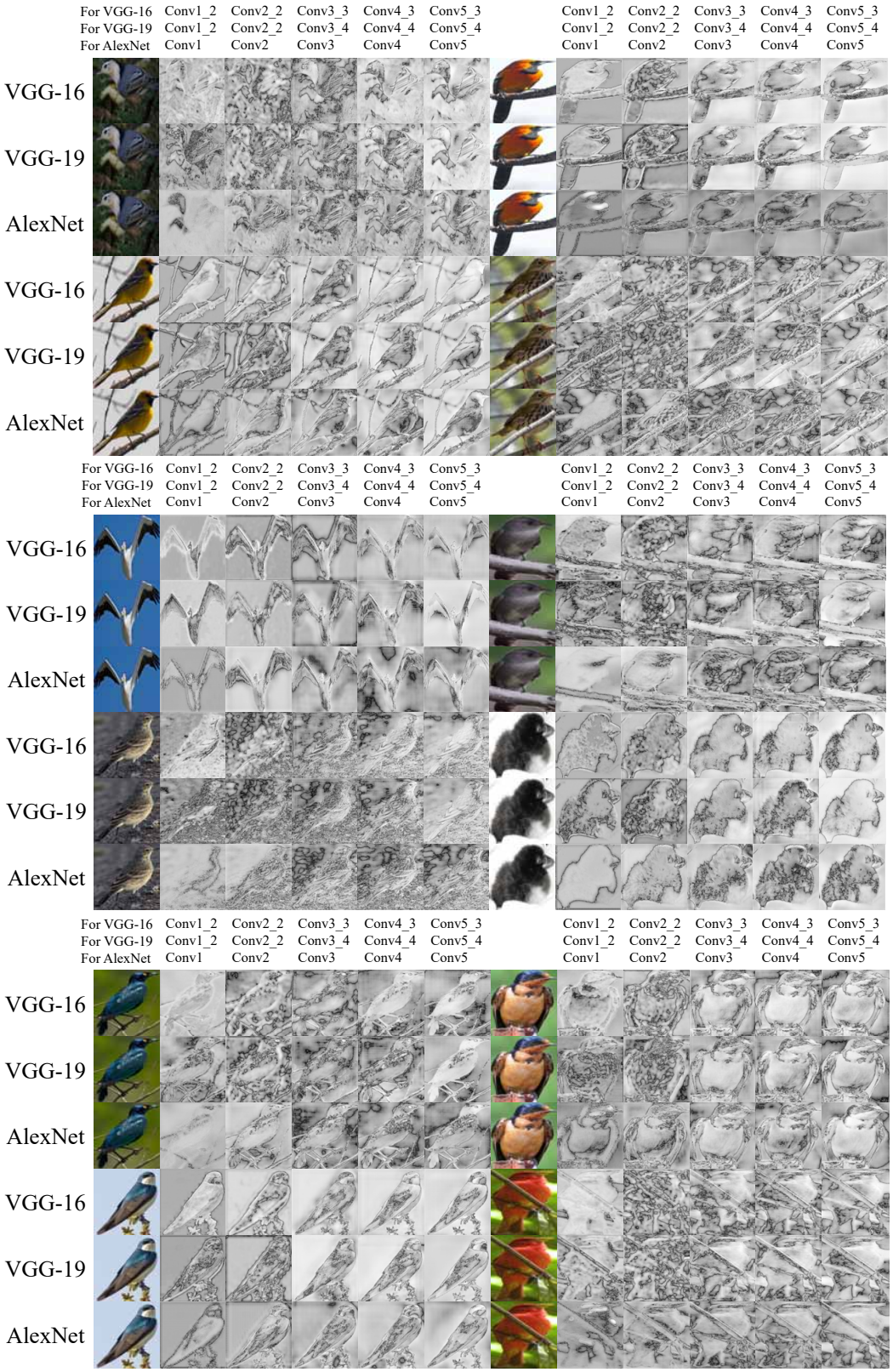



Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44
Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44
Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44

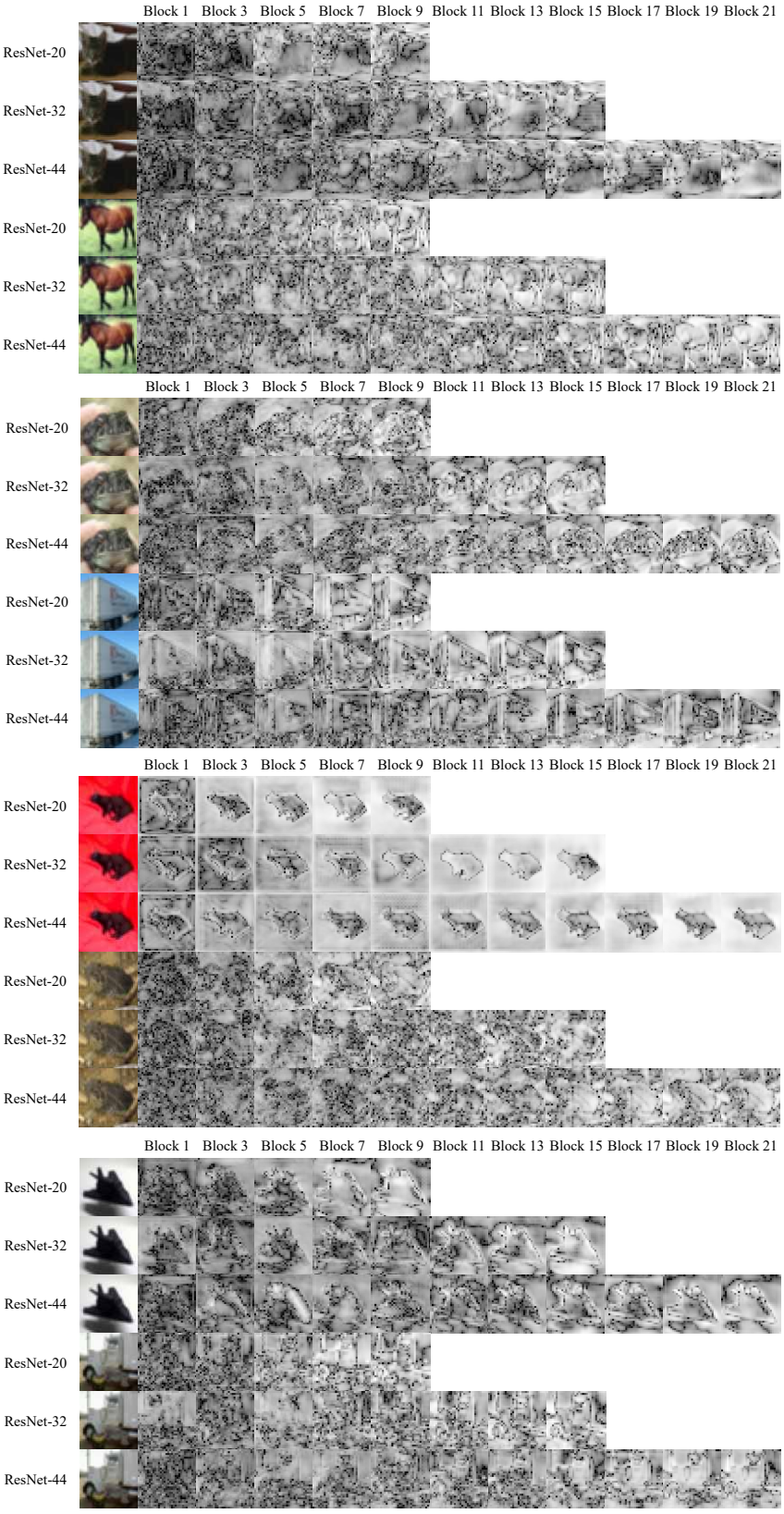
D VISUALIZATION OF PIXEL-WISE RU

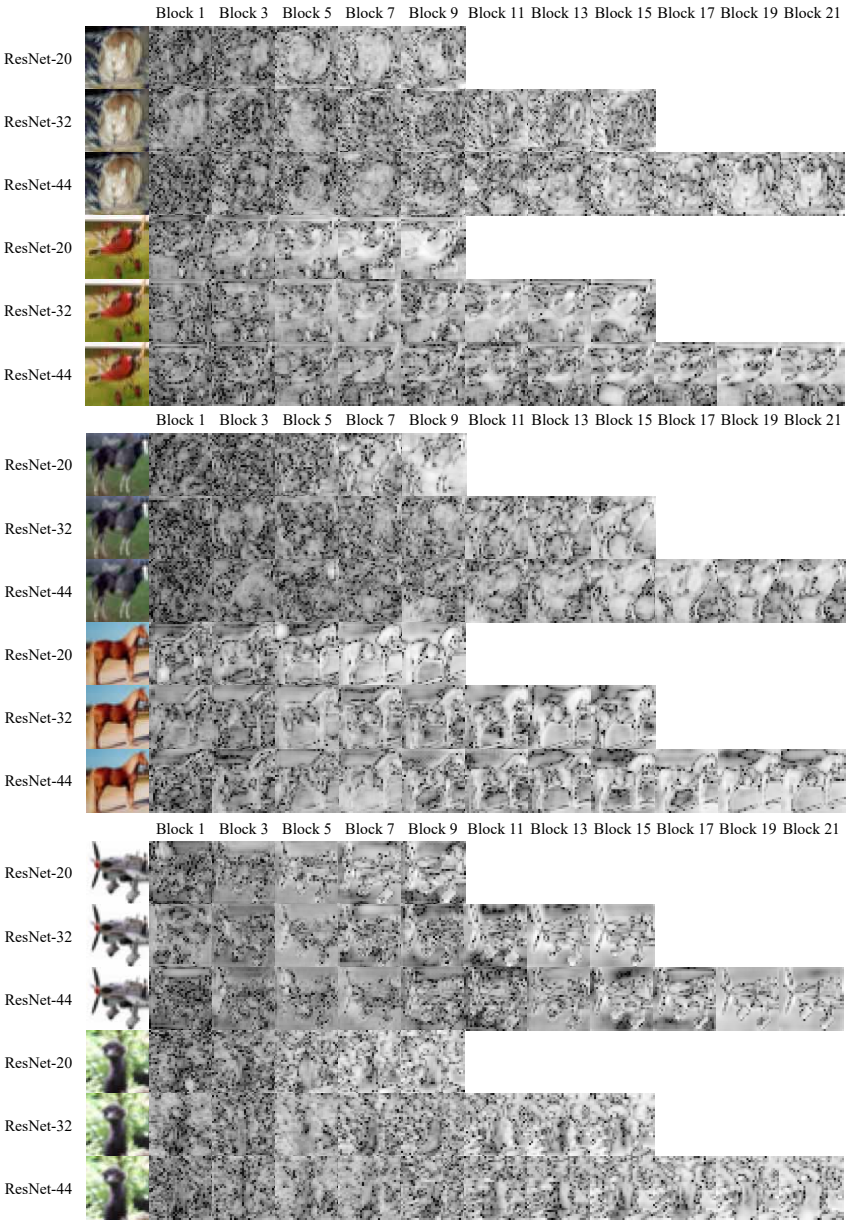
D.1 FOR THE VGG-16 LEARNED USING THE CUB200-2011 DATASET




For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5 Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet
For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5 Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet
For VGG-16 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3 Conv1_2 Conv2_2 Conv3_3 Conv4_3 Conv5_3
For VGG-19 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4 Conv1_2 Conv2_2 Conv3_4 Conv4_4 Conv5_4
For AlexNet Conv1 Conv2 Conv3 Conv4 Conv5 Conv1 Conv2 Conv3 Conv4 Conv5
VGG-16
VGG-19
AlexNet
VGG-16
VGG-19
AlexNet

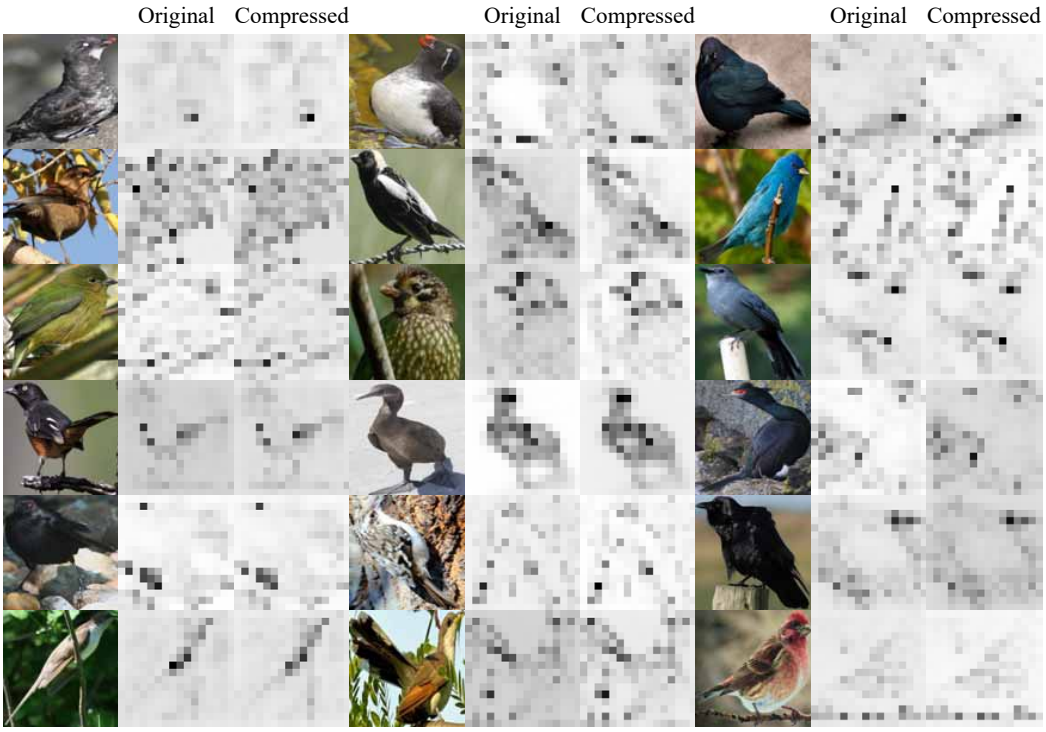
D.2 FOR THE RESNET-20/32/44 LEARNED USING THE CIFAR-10 DATASET

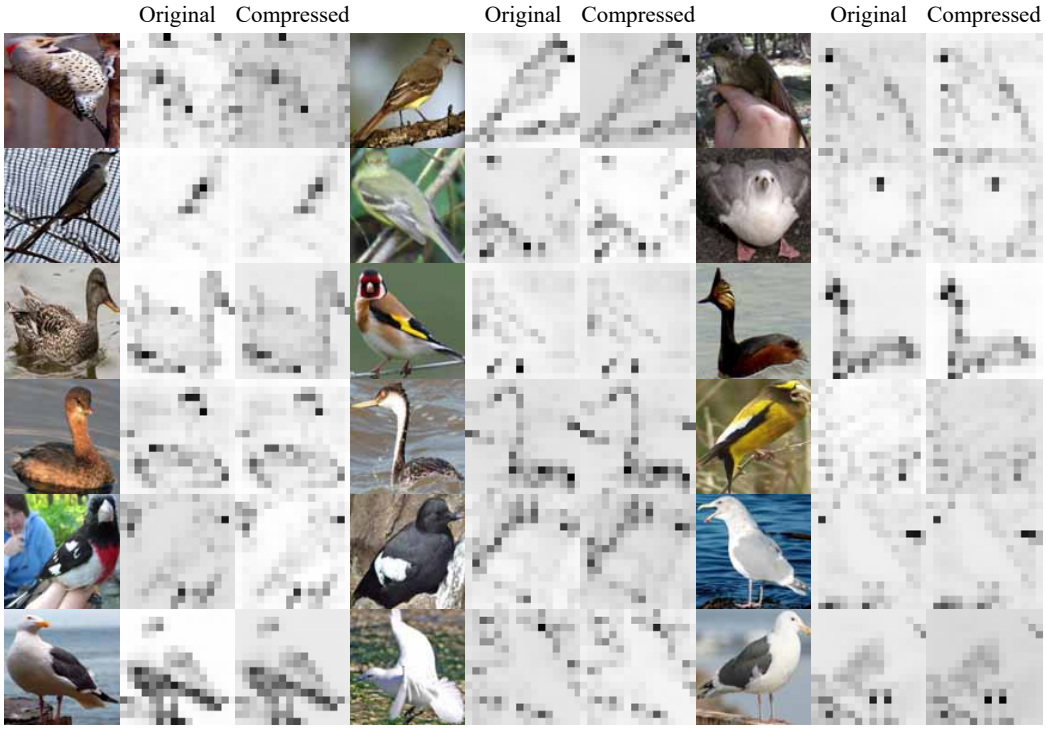


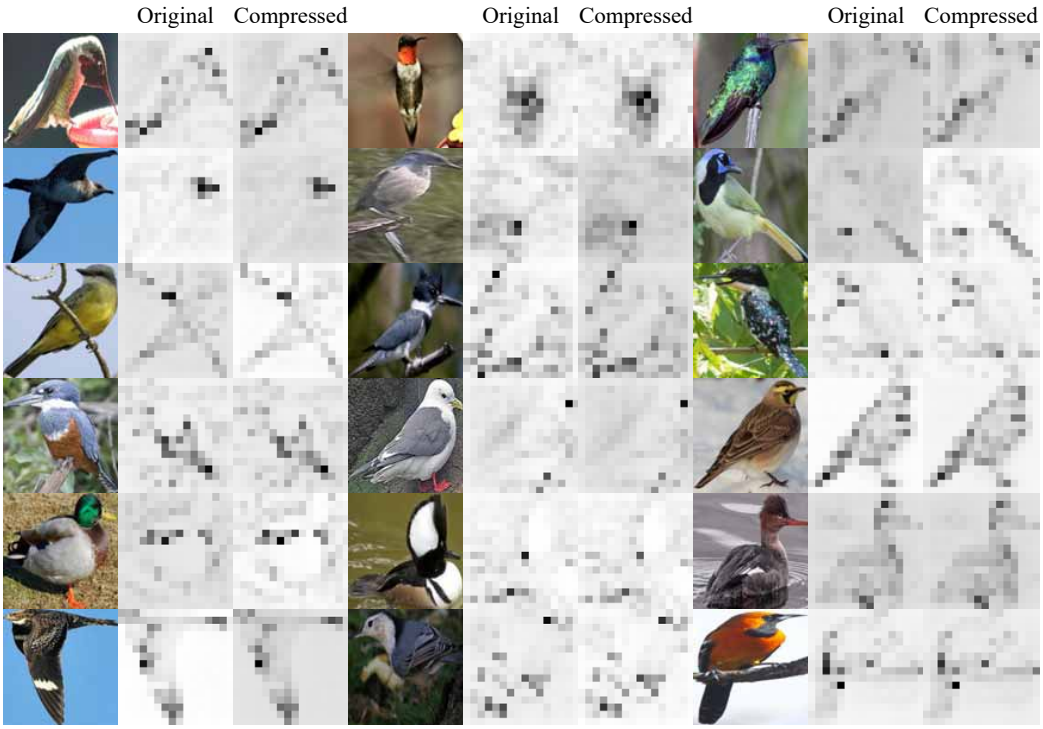

Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44
Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44
Block 1 Block 3 Block 5 Block 7 Block 9 Block 11 Block 13 Block 15 Block 17 Block 19 Block 21
ResNet-20
ResNet-32
ResNet-44
ResNet-20
ResNet-32
ResNet-44

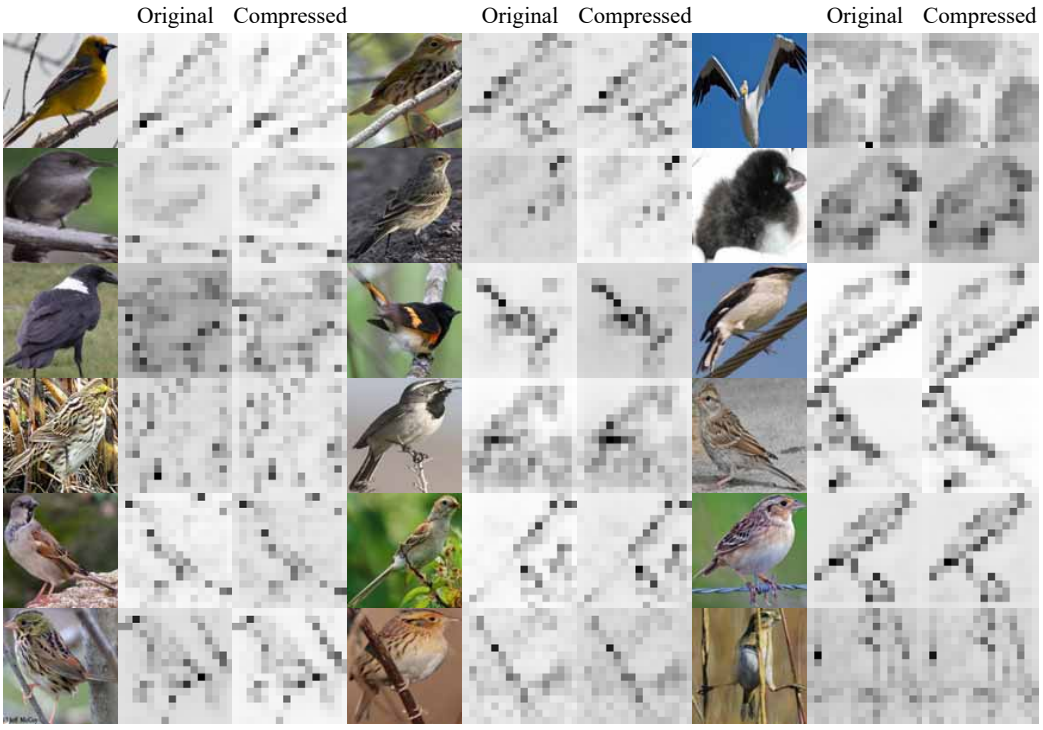
E COMPARISONS OF PIXEL-WISE SID BETWEEN THE ORIGINAL DNN AND THE COMPRESSED DNN

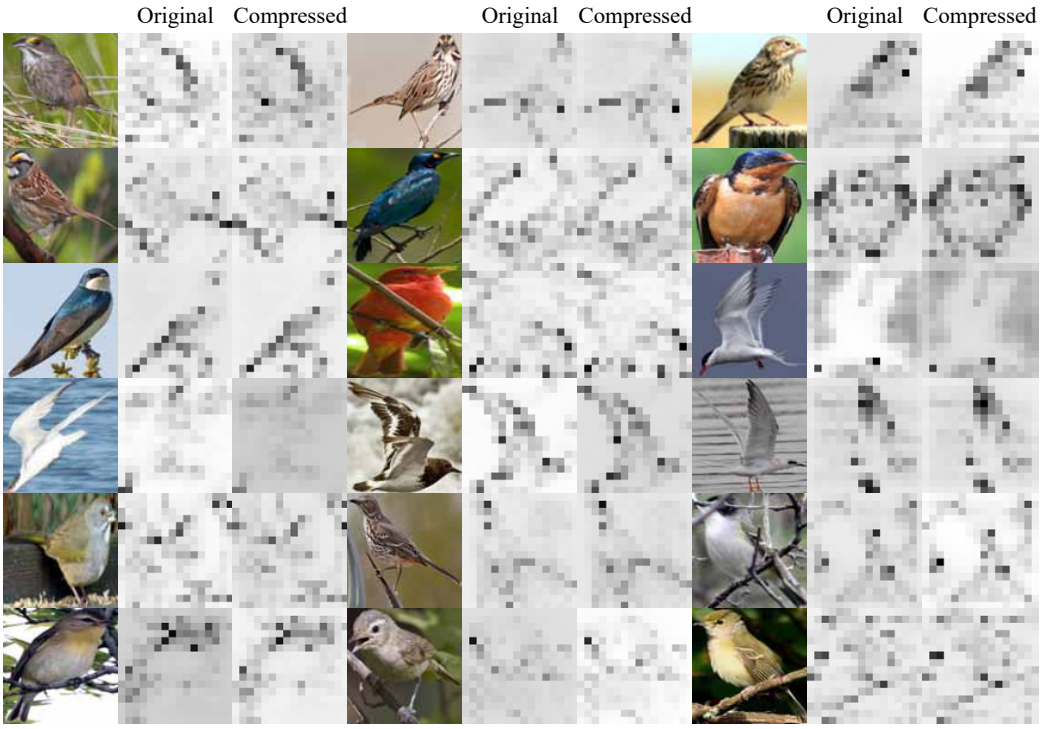
When we removed 93.3% parameters from the VGG-16 network, the network compression did not significantly change the pixel-wise SID of intermediate-layer features.





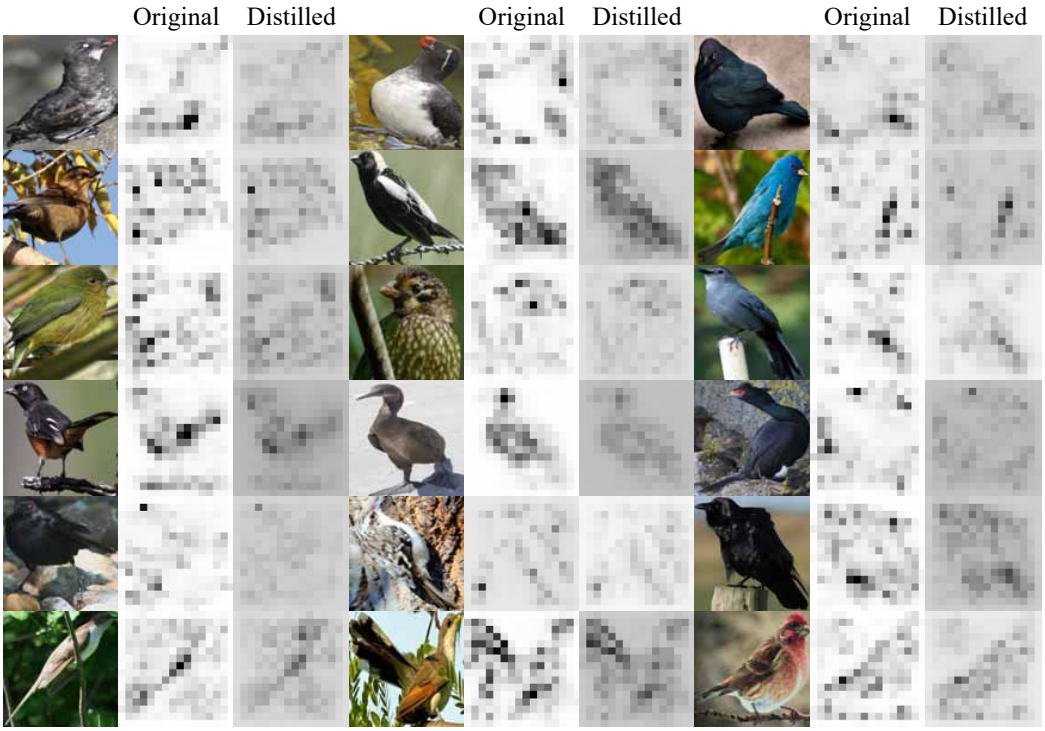

Original
Compressed
Original
Compressed
Original
Compressed

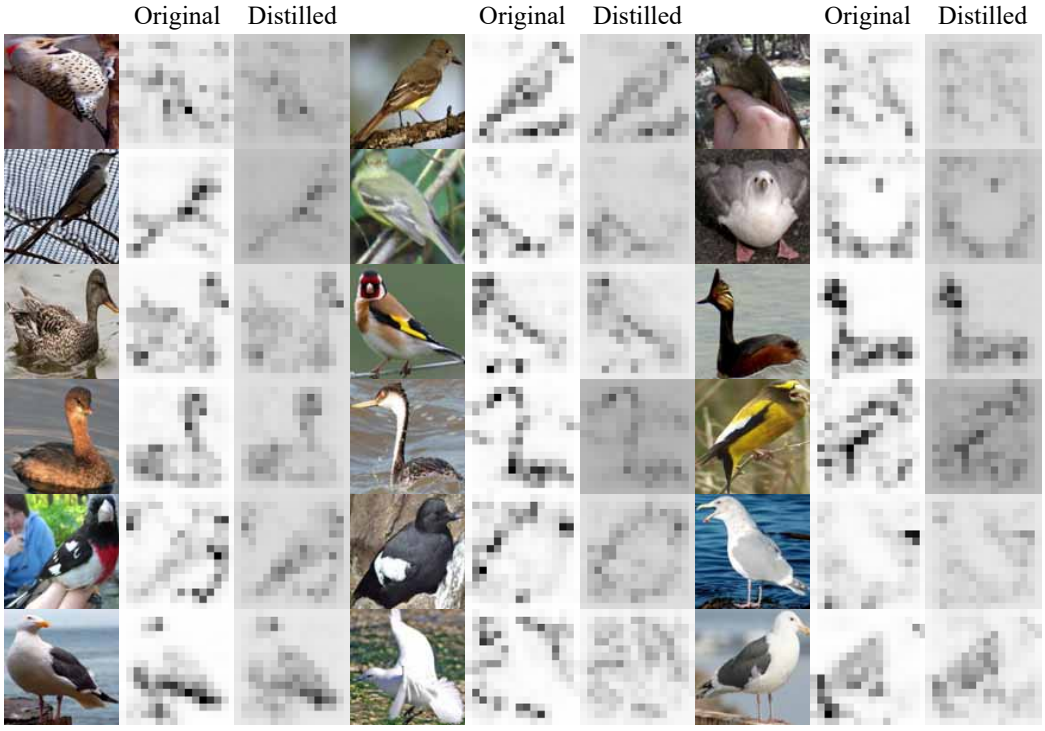

Original
Compressed
Original
Compressed
Original
Compressed

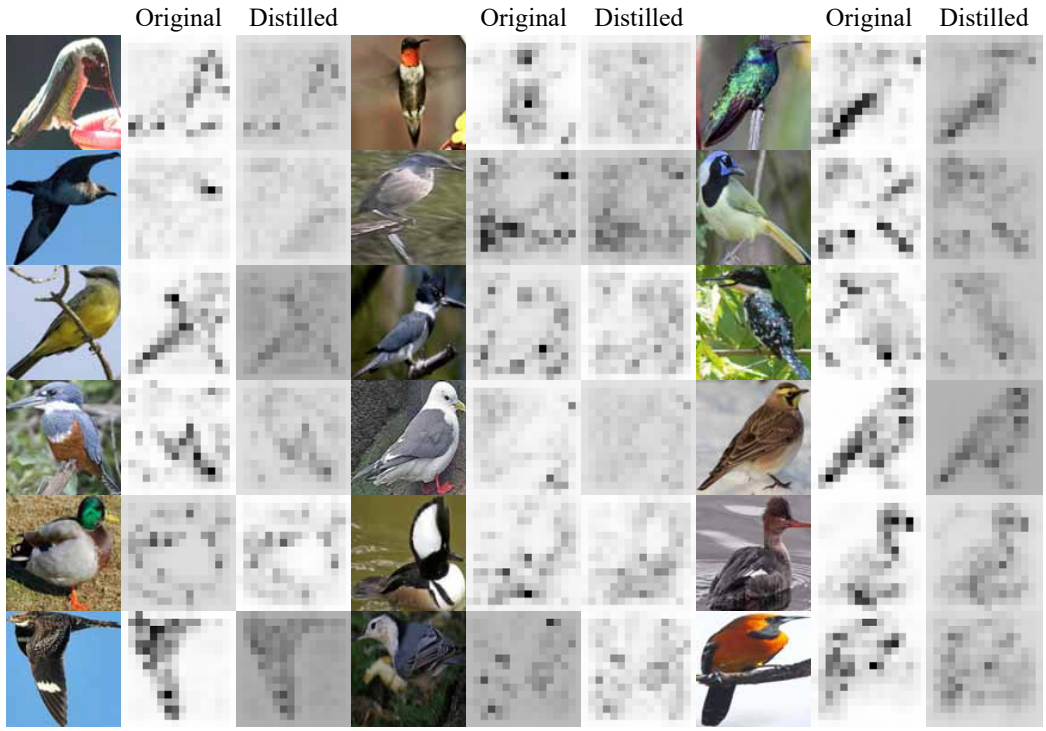

Original
Compressed
Original
Compressed
Original
Compressed

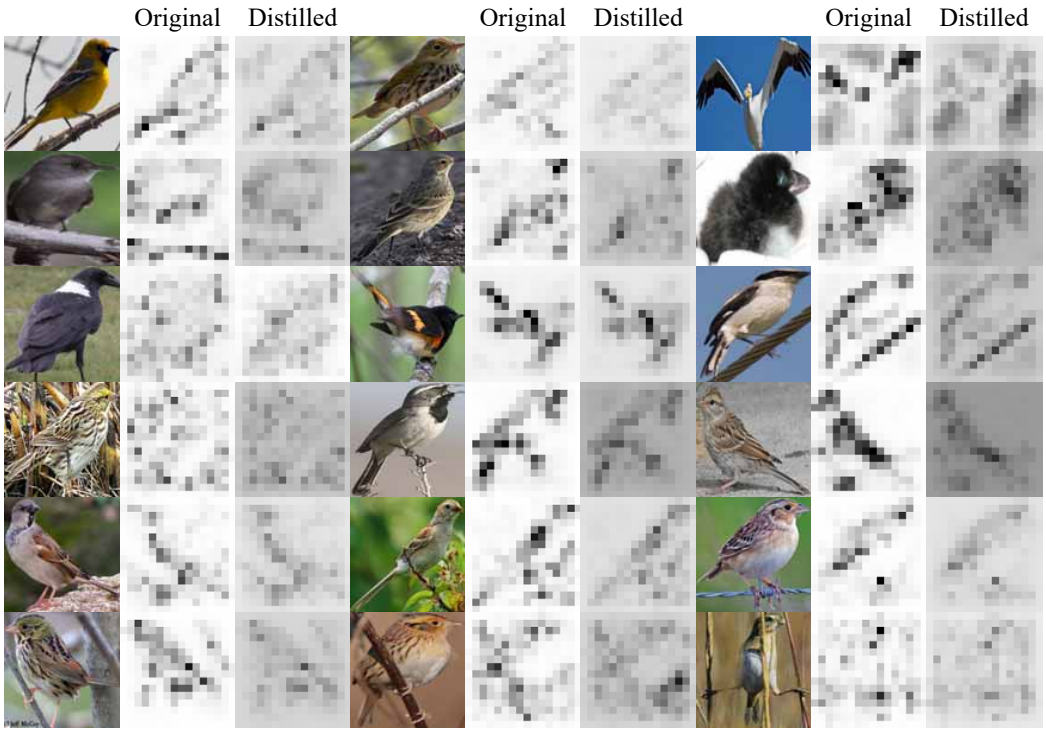
F COMPARISONS OF PIXEL-WISE SID BETWEEN THE ORIGINAL DNN (THE TEACHER) AND THE DNN LEARNED VIA KNOWLEDGE DISTILLATION (THE STUDENT)

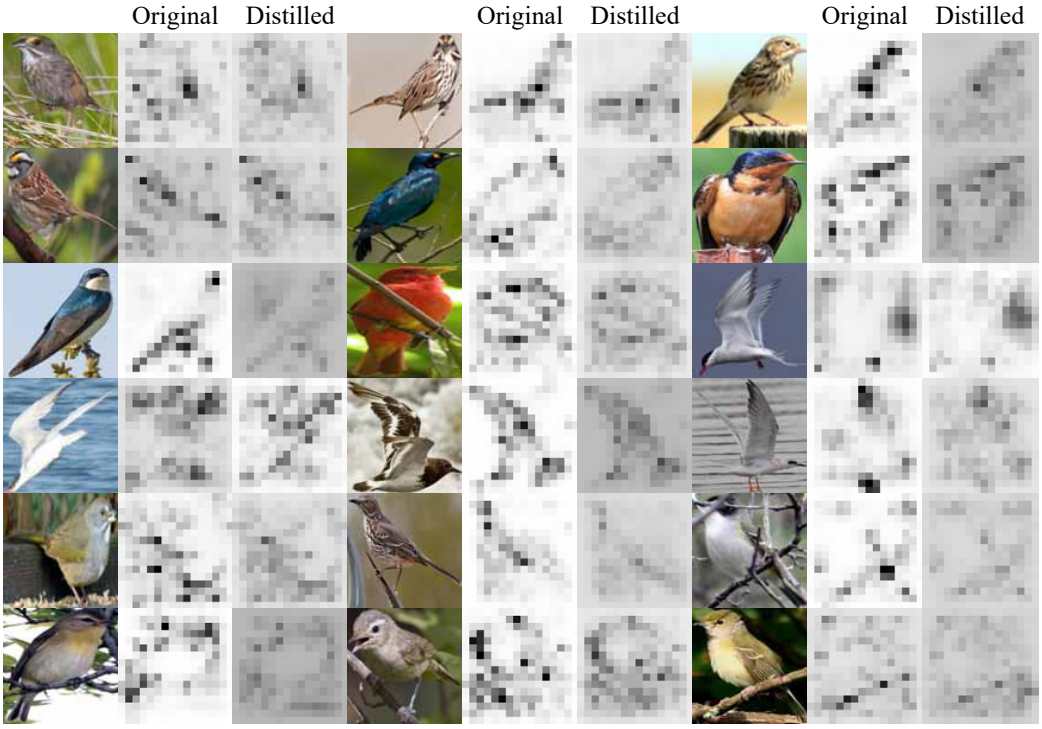
We visualized the pixel-wise SID of VGG-16 networks that were learned using the CUB200-2011 dataset (Wah et al., 2011).





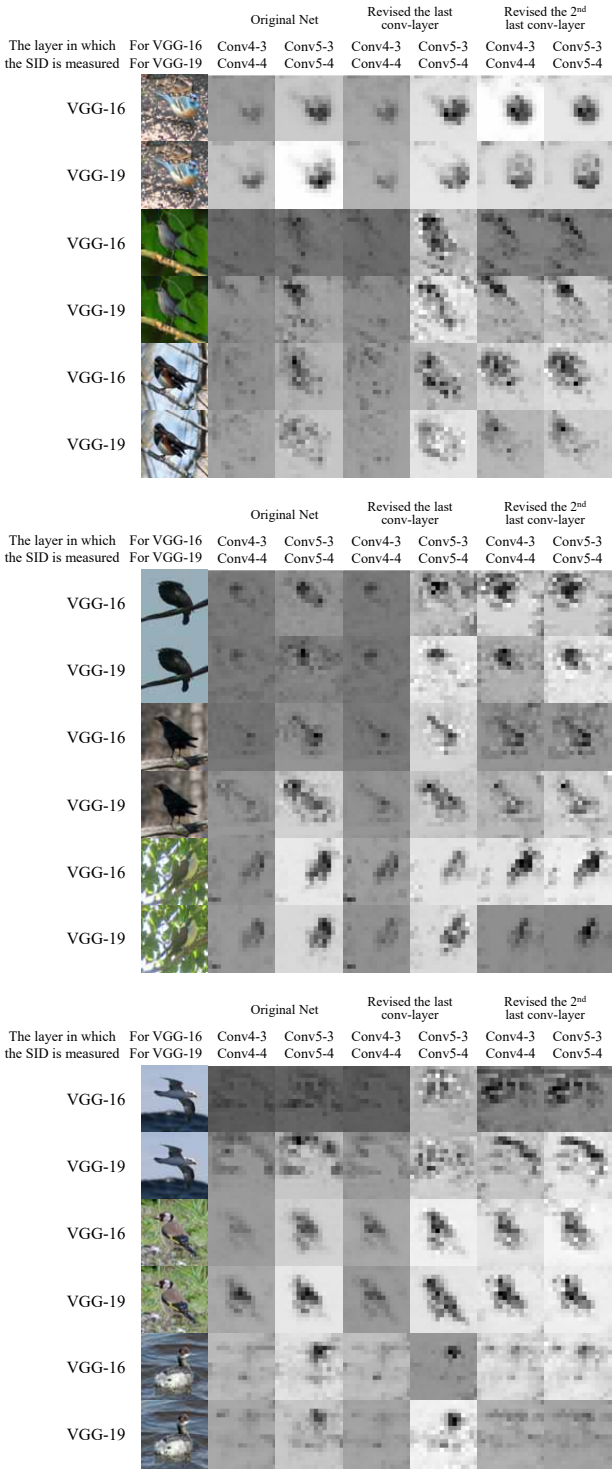

Original
Distilled
Original
Distilled
Original
Distilled

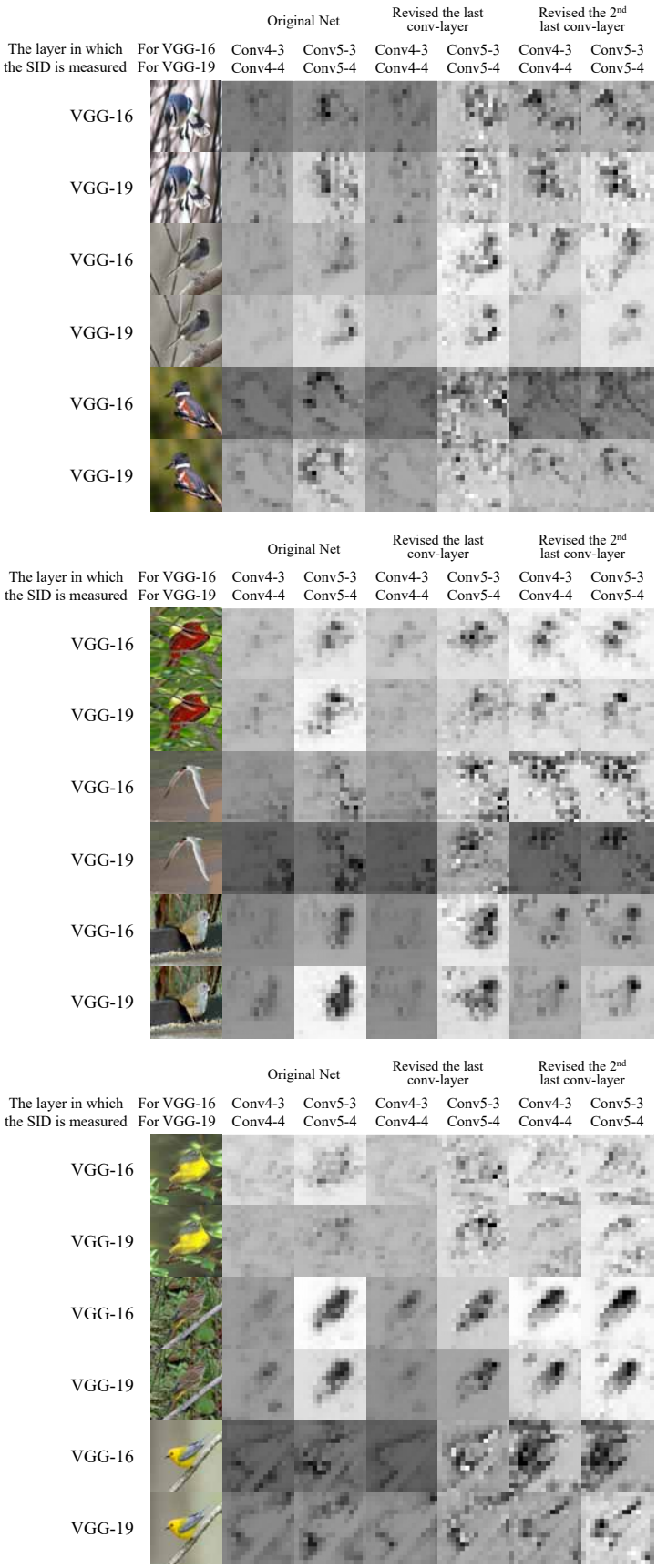

Original
Distilled
Original
Distilled
Original
Distilled


Original
Distilled
Original
Distilled
Original
Distilled

G COMPARISONS OF PIXEL-WISE SID BETWEEN THE ORIGINAL AND THE REVISED DNNS

We visualized the pixel-wise SID of the original and damaged networks that were learned using the CUB200-2011 dataset (Wah et al., 2011). We focused on the VGG-16 and VGG-19 networks. For each neural network, we revised either the last convolutional layer or the second last convolutional layer to generated the revised networks.




Original Net
Revised the last conv-layer
Revised the 2nd last conv-layer
The layer in which the SID is measured
For VGG-16
For VGG-19
Conv4-3
Conv5-3
Conv4-4
Conv5-4
VGG-16
VGG-19
VGG-16
VGG-19
VGG-16
VGG-19
Original Net
Revised the last conv-layer
Revised the 2nd last conv-layer
The layer in which the SID is measured
For VGG-16
For VGG-19
Conv4-3
Conv5-3
Conv4-4
Conv5-4
VGG-16
VGG-19
VGG-16
VGG-19
VGG-16
VGG-19
Original Net
Revised the last conv-layer
Revised the 2nd last conv-layer
The layer in which the SID is measured
For VGG-16
For VGG-19
Conv4-3
Conv5-3
Conv4-4
Conv5-4
VGG-16
VGG-19
VGG-16
VGG-19
VGG-16
VGG-19