Exploring Adversarial Examples and Adversarial Robustness of Convolutional Neural Networks by Mutual Information

Jiebao Zhang, Wenhua Qian, Rencan Nie, Jinde Cao, Fellow, IEEE, and Dan Xu

Abstract—A counter-intuitive property of convolutional neural networks (CNNs) is their inherent susceptibility to adversarial examples, which severely hinders the application of CNNs in security-critical fields. Adversarial examples are similar to original examples but contain malicious perturbations. Adversarial training is a simple and effective defense method to improve the robustness of CNNs to adversarial examples. The mechanisms behind adversarial examples and adversarial training are worth exploring. Therefore, this work investigates similarities and differences between normally trained CNNs (NT-CNNs) and adversarially trained CNNs (AT-CNNs) in information extraction from the mutual information perspective. We show that 1) whether NT-CNNs or AT-CNNs, for original and adversarial examples, the trends towards mutual information are almost similar throughout training; 2) compared with normal training, adversarial training is more difficult and the amount of information that AT-CNNs extract from the input is less; 3) the CNNs trained with different methods have different preferences for certain types of information; NT-CNNs tend to extract texture-based information from the input, while AT-CNNs prefer to shape-based information. The reason why adversarial examples mislead CNNs may be that they contain more texture-based information about other classes. Furthermore, we also analyze the mutual information estimators used in this work and find that they outline the geometric properties of the middle layer’s output.

Index Terms—Adversarial attacks, adversarial examples, deep neural networks, mutual information, information bottleneck.

I. INTRODUCTION

CONVOLUTIONAL Neural Networks (CNNs) have achieved surpassing performance on many tasks [1], e.g., image classification [2], object detection [3], style transfer [4], image captioning [5], and other fields. However, researchers have found that adding small malicious perturbations crafted by adversaries to the original examples can drastically degrade the performance of CNNs [6], [7]. This makes CNNs unreliable for security-sensitive tasks. There are many methods to improve the resistance of CNNs against adversarial examples. Adversarial training (AT), a training paradigm, is one of the most effective defense methods [8]–[10]. Compared to the objective of previous normal training that minimizes the empirical risk of the original dataset, adversarial training pursues the empirical risk minimization over the modified dataset containing malicious perturbations.

Where the vulnerability of CNNs to adversarial examples stems from and why adversarial training can improve the robustness of CNNs are worth exploring. At first, Szegedy et al. argues that the vulnerability of CNNs to adversarial examples is caused by the local linearity of the neural networks [8]. Recent works claim that the vulnerability could be attributed to the presence of non-robust features [11]. The robustness of adversarially trained CNNs is a consequence of the fact that models learn the representations aligned better with human perception, namely the shape-based representation (e.g., shapes and contours in images) [12]. Many experimental results explicitly or implicitly support this view. The BagNet [13] is a simple variant of the ResNet [14] with limiting the receptive field size of the topmost convolutional layer. Its performance and uncomplicated architecture suggest that current network architectures base their decisions on relatively weak and local statistical regularities of inputs [13]. For example, the number of wheels in a image determines whether it is a bicycle or a car. Furthermore, Geirhos et al. and Hermann et al. demonstrate that ImageNet-trained CNNs are inclined to make decisions by the texture-based representation rather than shape-based representation [15], [16]. Echoing the literature mentioned above, Zhang et al. also show that adversarial training can change the texture-based representation bias of CNNs and force CNNs to appreciate more the shape-based information [17].

Moreover, some researchers have tried to explain the working mechanism of deep neural networks (DNNs), e.g., information bottleneck (IB) principle [18]. In IB principle, given two random variables $X$ (e.g., images) and $Y$ (e.g., labels), $Y$ implicitly determining the relevant and irrelevant features in $X$, an optimal representation $\hat{X}$ of $X$ can capture the relevant features about predicting $Y$ [18]. The IB principle provides a new information-theoretic optimization criteria for optimal DNN representations [19]. The trends towards the mutual information in DNNs estimated by the binning method appear to report the plausibility of the IB principle [20]. Mean-
while, inspired by the IB method, deep variational information bottleneck (VIB) [21] and non-linear information bottleneck (NIB) [22] objectives are proposed. Compared with deterministic models with various forms of regularization, the models trained with the VIB or NIB objective have better performance concerning generalization and resistance to adversarial attacks [21]. However, there are some arguments about the applicability and conclusions derived from the IB principle [23], [24]. Saxe et al. demonstrate that the information plane trajectory is predominantly a function of the nonlinear activation and the double-sided saturating nonlinear function (e.g., tanh) yields the compression phase [25]. Goldfeld et al. claim that the compression phenomenon throughout training is driven by progressive geometric clustering of the middle representation [26].

Here we put these debates about the IB principle aside. In this work, we get inspiration from the way the mutual information is used to analyze the relevant information between layers in the IB principle. We study the similarities and differences between normally training CNNs (NT-CNNs) and adversarially training CNNs (AT-CNNs) from the mutual information perspective. The relevant information is quantified during the training process between the middle layer’s output and the example, and between the middle layer’s output and the label. Specifically, for a label variable $Y$, an input variable $X$, and a corresponding $i$-th layer’s output $T_i$ of a model, we can estimate the mutual information $I(T_i; X)$ and $I(T_i; Y)$ on each training epoch, as shown in Fig. [1] Whether NT-CNNs or AT-CNNs, we feed original and adversarial examples into models and explore the trends towards $I(T_i; X)$ and $I(T_i; Y)$ throughout training. We have some observations after analyzing the experimental results. First, whether NT-CNNs or AT-CNNs, for original and adversarial examples, the trends towards mutual information are almost similar throughout training; second, compared with normal training, adversarial training is more difficult and the amount of information that AT-CNNs extract from the input is less; and third, the CNNs trained with different methods have different preferences for certain types of information; NT-CNNs tend to extract texture-based information from the input, while AT-CNNs prefer to shape-based information.

The vulnerability of NT-CNNs to adversarial examples may derive from the extraction preferences about some types of information. The reason why adversarial examples easily mislead NT-CNNs may be that they contain more texture-based information about other categories, while NT-CNNs prefer to extract texture-based one. Conversely, AT-CNNs’ predictions predominately depend on shape-based information. The fact that small perturbations in adversarial examples do not easily change the shape-based information may bring about robustness in AT-CNNs. Furthermore, we find that the mutual information estimators outline some properties of the middle layer’s output from the geometric perspective. The tighter the middle layer’s output cluster, the smaller the mutual information $I(T_i; Y)$. When the middle layer’s output distribution is determined, the tighter the middle layer’s output belongs to one class cluster, the larger the mutual information $I(T_i; Y)$.

In summary, this article has made the following contributions.

- We explore the trends towards $I(T_i; X)$ and $I(T_i; Y)$ on NT-CNNs and AT-CNNs during the training phase. The empirical results show whether the input is the original or adversarial example, the mutual information trend of AT-CNN nearly coincides with the one of NT-CNN.
- We analyze the information curves about $I(T_i; X)$ and $I(T_i; Y)$ on NT-CNNs and AT-CNNs when the original and adversarial input suffers information distortion. The empirical results demonstrate that CNNs trained with different training paradigms indeed arise the information extraction bias.
- We uncover that the mutual information estimators used in this work practically outline some geometric properties of the middle layer’s output after analyzing key parts of algorithms.

The rest of this paper is organized as follows. Section II reviews some literature on adversarial attacks and adversarial training methods. Section III elaborates on the mutual information estimators. Section IV illustrates extensive experimental results based on various models and datasets. Section V analyzes the mutual information estimators used in this work and concludes some viewpoints from the geometric perspective. Finally, conclusions are given in Section VI.

II. RELATED WORK

A. Adversarial Attacks

In the inference phase, when examples contain human-imperceptible and malicious perturbations, the classification performance of DNNs shows a sharp decrease [6], [7]. This inherent weakness of DNNs arouses the interest of researchers. Many adversarial attack methods have been proposed to seek perturbations according to the exclusive attributes of DNNs and optimization techniques. Adversarial attacks can be divided into two categories: white-box and black-box attacks. White-box attacks can be further divided into optimization-based attacks [7], [27], single-step attacks [8], [10], and iterative attacks [9], [28]–[30].

Optimization-based attacks formulate finding an optimal perturbation as a box constraint optimization problem. Experiments show that optimization-based attacks can achieve excellent attack performance. The L-BFGS attack uses the second-order Newton method to solve this problem [7]. Compared with the L-BFGS attack, the C&W attack uses variable substitution to bypass the box constraint, replace the objective with a more powerful one, and uses the Adam optimizer [31] to solve the optimization problem [27], the expression as follows:

$$\begin{align} \min_{\delta} \|\delta\|_p + c \cdot f(x + \delta) \\
{\text{subject to}} \quad x + \delta &\in [0, 1]^n \\
f(\cdot) &= \max(\max\{Z(\cdot)_{1,i\neq t} - Z(\cdot)_t, -\kappa\}) \\
\delta &= \frac{1}{2}(\tanh(w) + 1) - x, \end{align}$$

where $\delta$ denotes the perturbation vector, $x$ is original input, $Z(\cdot)$ represent the target DNN, $t$ is the target label index and $\kappa \geq 0$ is a tuning parameter for attack transferability.
Single-step attacks are straightforward and effective to avoid the high computation costs caused by optimization attacks. Since models are assumed to be locally linear in single-step attacks, the perturbation is directly added along the gradient direction of the model to the original examples.

Iterative attacks achieve the trade-off between computation and attack performance. Iterative attacks add perturbations several times. After each gradient is calculated, perturbations are added. The MI-FGSM incorporates the momentum term to stabilize the perturbation direction, which promotes the attack performance and the transferability of adversarial examples [30]. The PGD attack does not directly add perturbations to the original example but selects a substitute example in the example’s vicinity for subsequent operations [9], formulated as

\[ x^{k+1} = \Pi_{x+S} \left( x^k + \alpha \text{sgn} (\nabla_x L(\theta, x, y)) \right), \]

where \( y \) is the label, \( \theta \) denotes the model parameters, \( L \) is the loss function of DNNs, \( \alpha \) is the step size in attacks, \( S \) denotes the set of allowed perturbations, \( \Pi \) is the projection operation, and \( x^k \) represents the example endures \( k \) perturbations. The DeepFool approximates general non-linear classifiers to affine multiclass classifiers to construct the misclassification boundary and find the minimum perturbations to move examples to the nearest misclassification boundary [29]. The ability of white-box attacks paves the way for adversarial training.

Transfer-based attacks [32, 33] take various forms. Since researchers have found that the attack ability of adversarial examples is transferable [7, 8, 32–34]. Namely, adversarial examples crafted for a certain model may also have the attack effect on other models. Using this transferability, adversaries can use substitute models to generate adversarial examples without the knowledge of the specific structure and parameters of the target model to be attacked. So, the adversarial examples generated by the above white box attack are all transferable.

Black-box attacks mean that the adversaries barely have the information of the structures and parameters of the target model, but can query the output of models to examples that attackers feed into models. The OPA attack [35] is a black-box attack based on the differential evolution algorithm. When only the confidence score of classifiers to the images is known, this attack method only needs to change a few pixels in images to achieve a strong attack effect. The ZOO-attack [36] is also a black-box attack, based on zeroth-order optimization. It approximates the gradient by querying the model’s confidence score to the input and imitates the optimization-based attack. The Boundary attack [37], which solely relies on the final model decision, resort to the random walk and rejection sampling techniques. The performance is competitive with that of gradient-based attacks.

B. Adversarial Training

Goodfellow et al. initially proposed adversarial training [8], and PGD-AT [9] has become the mainstream method. The PGD-AT formulates how to find the parameters of a robust model as a saddle point problem:

\[ \min_{\theta} \left( \mathbb{E}_{(x,y) \sim D} \left[ \max_{\delta \in S} L(\theta, x + \delta, y) \right] \right), \]

where \((x, y)\) denotes the image-label pair that is sampled from dataset \(D\). The solution to the inner maximization problem is to find the adversarial example with a high loss. The outer minimization is to update the model parameters to remove the effect of adversarial attacks.

The FreeAT [38] optimizes the training speed. At each step of generating adversarial examples, it reuses the gradient of the model on immature adversarial examples to update the model parameters, shorten the epochs of adversarial training and reduce the computation. The YOPO [39] regards a DNN as a dynamic system, considers that the layers are uncoupled, and accelerates the training speed by limiting the number of forward and backward propagations. Compared with the improvement of training speed by the FreeAT and YOPO,
the FreeLB \cite{40} pays more attention to the optimization effect of each min-max process. It is similar to PGD-AT with slight differences. The FreeLB first obtains the average of the gradients of the model to immature adversarial examples in the $k$ iterations, and then the parameters of the model are updated by the average of the gradients.

III. ESTIMATE THE MUTUAL INFORMATION IN DETERMINISTIC CNNS

A. Preliminaries

A DNN can be regarded as a complicated function nested by some simple functions (e.g., fully-connected layers, convolutional layers) $f_1, f_2, \cdots, f_L$, where $f_i$ accept the output of $f_{i-1}$, then the output of $f_i$ as the input of $f_{i+1}$. For an input variable $X$ (i.e., the image), a label variable $Y$, the corresponding $i$-th middle layer’s output is $T_i$, calculated by

$$T_i = f_i(f_{i-1}(\cdots f_1(X)\cdots)), i = 1, 2, \cdots, L \tag{4}$$

Especially, the input of $f_1$ is $X$, and the output of $f_L$ represents the final output of a DNN. Obviously, for deterministic DNNs, $T_i$ is a function of $X$.

For two random variables $X$ and $Y$, the joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$, the entropy $H(X)$ is a measure of the uncertainty of the random variable $X$. Notably, for brevity, we use $H$ to denote both the discrete entropy and differential entropy. This is sufficient because we estimate the mutual information in deterministic DNNs, and the entropy of mixture distributions ($\gamma$-mixtures) is estimated by KDE \cite{42}. In \cite{22}, \cite{25}, \cite{26}, researchers transform deterministic DNNs into stochastic ones by artificially injecting Gaussian noise into the middle layer’s output to avoid the infinite value when estimating the mutual information. Concretely, in deterministic DNNs, the middle layer’s output $T_i = f_i(T_{i-1})$, but in stochastic DNNs, $T_i = f_i(T_{i-1}) + Z$, $Z$ is Gaussian noise, and $Z \sim \mathcal{N}(0, \sigma^2 I)$, as shown in Fig. 2. Following the setting \cite{25}, in this work the noise is added solely when estimating mutual information and is not present during the training or testing phase. The middle layer’s output of a mini-batch can be regarded as a multivariate Gaussian mixture distribution in estimation, then we have

$$I(T_i; X) = H(T_i) - H(T_i \mid X) = H(T_i) - H(Z) \tag{7}$$

When $H(Z)$ is directly computed and the upper and lower bounds of the entropy of $H(T_i)$ are estimated by KDE methods, the upper and lower bounds of mutual information $I(T_i; X)$ will be known. The formula is as follows:

$$\hat{I}(T_i; X) = \hat{H}(T_i) - \hat{H}(T_i \mid X) \tag{8}$$
where \( N \) denotes the number of samples in a mini-batch, \( \ell^{(j)}_i \) is the output of \( i \)-th hidden layer for sample \( j \), and \( \gamma = 4 \) for computing the lower bound (\( \gamma = 1 \) for computing the upper bound). Furthermore, the lower and upper bounds of \( I(T_i; Y) \) are as follows:

\[
\hat{I}(T_i; Y) = \hat{H}(T_i) - \hat{H}(T_i | Y)
\]

\[
= \frac{1}{N} \sum_j \log \frac{1}{N} \sum_k \exp \left( -\frac{1}{2} \frac{\| \ell^{(j)}_i - \ell^{(k)}_i \|_2^2}{\gamma \cdot \sigma^2} \right) - \frac{1}{m} \sum_m \frac{N_m}{N} \sum_{y^{(j)}_m = m} \log \frac{1}{N_m} \sum_{y^{(k)}_m = m} \exp \left( -\frac{1}{2} \frac{\| \ell^{(j)}_i - \ell^{(k)}_i \|_2^2}{\gamma \cdot \sigma^2} \right),
\]

where \( N_m \) is the number of samples with class label \( m \) and \( y^{(j)}_m \) is the corresponding label for \( x^{(j)} \). More details for estimating \( I(T_i; X) \) and \( I(T_i; Y) \) by KDE methods are shown in Algorithm 1.

**The binning method.** The binning method can be adopted to discretize continuous values (e.g., the middle layer’s output), and indirectly avoid the infinite differential entropy of continuous random variables. In [20], the layers’ arctan output activations are binned into 30 equal intervals between -1 and 1. After counting the activations in each interval, the empirical joint distribution and marginal distributions can be calculated for computing the mutual information, a simple illustration shown in Fig. 3. As the bin size becomes smaller, the empirical distribution from sampling results of continuous variables will be more accurate but closer to the uniform distribution [26]. More details for estimating \( I(T_i; X) \) and \( I(T_i; Y) \) by the binning method are shown in Algorithm 2.

**Algorithm 1** Estimate \( I(T_i; X) \) and \( I(T_i; Y) \) by KDE methods

**Input:** \( N \) \( i \)-th middle layer’s output \( T_i \) in a mini-batch, \( \ell^{(1)}_i, \ell^{(2)}_i, \ldots, \ell^{(N)}_i \), \( N \) corresponding labels \( y^{(1)}, y^{(2)}, \ldots, y^{(N)} \), the bound selector \( \gamma \).

**Output:** The estimated mutual information, \( \hat{I}(T_i; X) \) and \( \hat{I}(T_i; Y) \).

1: Calculate \( \hat{I}(T_i; X) \) by (8);
2: Calculate \( \hat{I}(T_i; Y) \) by (9);
3: return \( \hat{I}(T_i; X) \) and \( \hat{I}(T_i; Y) \).

**Algorithm 2** Calculate \( I(T_i; X) \) and \( I(T_i; Y) \) by the binning method

**Input:** \( N \) \( i \)-th middle layer’s output \( T_i \) in a mini-batch, \( \ell^{(1)}_i, \ell^{(2)}_i, \ldots, \ell^{(N)}_i \), and corresponding labels \( y^{(1)}, y^{(2)}, \ldots, y^{(N)} \), total number of classes \( M \), the binning interval size \( \lambda \).

**Output:** The estimated mutual information, \( \hat{I}(T_i; X) \) and \( \hat{I}(T_i; Y) \).

1: Initialization;
2: \( T_i = [\ell^{(1)}_i, \ell^{(2)}_i, \ldots, \ell^{(N)}_i] \);
3: \( y = [y^{(1)}, y^{(2)}, \ldots, y^{(N)}] \);
4: \( H_T = 0, H_{T|Y} = 0, S = [] \);
5: function DISCRETE_ENTROPY(L)
6: \( L \) different elements \( l, v^{(1)}, v^{(2)}, \ldots, v^{(K)} \), and corresponding occurrence times, \( o^{(1)}, o^{(2)}, \ldots, o^{(K)} \);
7: Calculate the entropy by empirical distribution as
\[
H = -\sum_{i=1}^{K} \frac{o^{(i)}}{\text{len}(L)} \log \frac{o^{(i)}}{\text{len}(L)};
\]
8: return \( H \);
9: end function
10: for \( j = 1 \rightarrow N \) do
11: Discrete the continuous value as
\[
\ell^{(j)}_i = \left\lfloor \frac{\ell^{(j)}_i}{\lambda} \right\rfloor;
\]
12: end for
13: \( H_T = \text{DISCRETE_ENTROPY}(T_i) \);
14: for \( m = 1 \rightarrow M \) do
15: \( S[m] = \text{find all elements whose class label is } m \) in \( T_i \);
16: \( H = \text{DISCRETE_ENTROPY}(S[m]) \);
17: Calculate the conditional entropy as
\[
H_{T|Y} = H_T - \frac{\text{len}(S[m])}{N} \cdot H;
\]
18: end for
19: return \( \hat{I}(T_i; X) = H_T, \hat{I}(T_i; Y) = H_T - H_{T|Y} \).

**IV. EXPERIMENTS**

**A. Empirical Setting**

1) **Datasets:** The train and test sets of MNIST [43], CIFAR-10 [44], and STL-10 [45] are used in this work. The MNIST’s train and test sets, both from 10 classes, contain 60,000 and 10,000 \( 1 \times 28 \times 28 \) images, respectively. The CIFAR-10’s train and test sets, both from 10 classes, contain 50,000 and 10,000 \( 3 \times 32 \times 32 \) images, respectively. The STL-10’s train and test sets, both from 10 classes, contain 5,000 and 8,000 \( 3 \times 96 \times 96 \) images, respectively.

2) **Models:** Two classic models are trained in this section: (a) the LeNet-5 [43], a simple CNN, achieves over 98% top-1 accuracy on MNIST; (b) the WideResNet [46] based on ResNet [14], improvement with respect to performance and training speed by increasing network width, achieves over 85% top-1 accuracy on CIFAR-10 and STL-10. The architectures of the models are specified in Table [II]. To reduce the amount of data, we do not obtain the output of all layers, but instead, select some important layers. In this work, for each model, the mutual information is estimated on 5 layers, as specified in Table [II].

3) **Adversarial attack and Training:** Adversarial attack and adversarial training are both based on the PGD method. But for different datasets and models, parameter settings for generating adversarial examples will be slightly different, the details as shown in Table [III].
4) Saturation and Patch-Shuffling: The saturation mainly changes the texture-based information of the image. For each pixel $v$ in an image, we change the saturation of $v$ as follows

$$v' = \text{sign}(2v - 1) \times |2v - 1|^2 \times 0.5 + 0.5,$$  \hspace{1cm} (13)

where $v'$ denotes the pixel with the saturation and $p$ is the saturation level. When $p = 2$, the original image does not undergo saturation and the saturation adjustment range is $[2, 8, 16, 64, 1024]$.

The patch-shuffling mainly changes the shape-based information of the image. For an image, we first evenly split it into $k \times k$ patches, and randomly combine these $k \times k$ patches into a new image. Note that when $k = 0$, it means that the original image will not be split. The size of $k$ is set to $[0, 2, 4, 8]$.

B. Information Flows in Normal Training and Adversarial Training

In this section, to explore the differences and similarities from the information perspective between NT-CNN and
Fig. 5. The plots of $I(T_i; X)$ and $I(T_i; Y)$ when the input suffers from different types of information distortion.

### TABLE I

**NORMAL TRAINING AND ADVERSARIAL TRAINING PARAMETERS.**

| Parameter      | MNIST Model | CIFAR-10 Model | STL-10 Model |
|----------------|-------------|----------------|--------------|
| Epochs         | 200         |                |              |
| Optimizer      | SGD         |                |              |
| Batch Size     | 128         |                |              |
| Learning Rate  | 0.1         |                |              |
| Momentum       | 0.9         |                |              |
| Milestone      | 20, 60      |                |              |
| Gamma          | 10.3        |                |              |
| Epsilon        | 45/255      | 8/255          | 4/255        |
| Alpha          | 8/255       | 2/255          | 2/255        |
| Step           | 7           | 7              | 7            |

### TABLE II

**THE DETAILS OF NETWORK ARCHITECTURES.** * denotes the layer that will be probed to obtain the output.

| LeNet-5 MNIST          | WideResNet CIFAR-10 | WideResNet STL-10 |
|-----------------------|---------------------|--------------------|
| Conv (5 × 5, 6) *     | Conv (3 × 3 × 16) * | Conv (3 × 3 × 16) * |
| Conv (5 × 5, 16) *    | Conv (3 × 3 × 16) * | Conv (3 × 3 × 16) * |
| Conv (5 × 3 × 32) *   | Conv (3 × 3 × 32) * | Conv (3 × 3 × 32) * |
| Conv (5 × 3 × 64) *   | Conv (3 × 3 × 64) * | Conv (3 × 3 × 64) * |
| AvgPool (8 × 8)       | AvgPool (8 × 8)     |                   |
| FC (64 × 10) *        | FC (64 × 10) *      |                   |
| FC (576 × 10) *       | FC (576 × 10) *     |                   |
AT-CNN in their responses to the original and adversarial examples, we conduct experiments on three data sets: MNIST, CIFAR-10, and STL-10; and two models: LeNet-5 and WideResNet. During the normal training and adversarial training, for different examples (i.e., original and adversarial examples), we used estimators mentioned in Section III-B to estimate the mutual information $I(T_i; X)$ between the i-th middle layer’s output $T_i$ and the input $X$, and the mutual information $I(T_i; Y)$ between the i-th middle layer’s output $T_i$ and the label $Y$ in each training epoch. When calculating mutual information, 5,000 examples are fed into the models to make the empirical distribution closer to the real distribution. The experimental results are shown in Fig. 4. It is worth noting that the last two columns of the first row of each subfigure overview the trends towards mutual information of each layer during training. Then the last three rows are information planes (the x-axis represents $I(T_i; X)$ and the y-axis represents $I(T_i; Y)$) of each layer, calculated by the three estimators, respectively. We obtain the following observations after analyzing the experimental results.

- With different training methods, over different datasets, on different CNN architectures, and estimating on different middle layers, there is a great diversity in the trends of the mutual information $I(T_i; X)$ and $I(T_i; Y)$. However, whether original or adversarial examples are the input, the trends of the mutual information are nearly consistent when other settings are the same. For example, for WideResNet normally trained on CIFAR-10, the trends of $I(T_i; X)$ and $I(T_i; Y)$ at all layers show a growth. When adversarially training on CIFAR-10, trends vary across layers. When WideResNet is trained on STL-10, the trends towards $I(T_i; X)$ and $I(T_i; Y)$ of some layers are increasing (e.g., the 1-st layer and 5-th layer) and ones of some layers are obscure. However, the trend on the original examples is almost in line with the one on adversarial examples.

- The compression phase claimed in [18] is almost not observed and the data processing inequality is not always clear.

- In contrast with NT-CNNs, the mutual information estimated in AT-CNNs, whether original or adversarial examples are as the input, shows a lagging advancement, and final results are at a lower level.

C. Texture and Shape Distortion

In this section, to explore the information extraction bias of NT-CNNs and AT-CNNs, we conducted experiments on two data sets, CIFAR-10 and STL-10, and one model, WideResNet. Concretely, for the model after 200 epochs of the normal and adversarial training, the examples (both original and adversarial examples) will suffer from texture or shape information distortion, then they are fed into the model. We calculate the lower bound of $I(T_i; X)$ and $I(T_i; Y)$ on the probed middle layers’ output and simultaneously record the classification accuracy and average cross-entropy loss. The experimental results are shown in Fig. 5. The first row of each subfigure displays the accuracy and average loss. The second row shows the mutual information of middle layers of models (both NT-CNNs and AT-CNNs) on examples suffering varying levels of texture-based and shape-based information distortions. We obtain the following observations after analyzing the experimental results.

- The patch-shuffling mainly changes the shape-based information contained in examples, but at the same time, the smaller patches are, the severely the texture-based information in the examples is distorted. The fact is the same for the saturation. As the saturation level increases, the texture-based information fades away and the shape information may also be wiped out.

- The NT-CNNs are more sensitive to texture-based information and the AT-CNNs are more sensitive to shape-based information. The discrimination between the responses of the AT-CNNs to the patch-shuffling and saturation operation is obvious. Whether the input is the original or adversarial example, as the patches become smaller, $I(T_i; X)$ and $I(T_i; Y)$ at the last layer fall down. Instead, the effect of the saturation is unapparent. Concretely, as the saturation level is enhanced, the mutual information shows slight decreases.

- The NT-CNNs’ responses to the patch-shuffling and saturation are more complicated. Specifically, higher saturation and smaller patches, for adversarial examples, may remove the texture-based information about other labels from the adversarial perturbations, and for original examples, may remove the texture-based information about the true label from themselves. For example, the experimental results of normally trained WideResNet on CIFAR-10 and STL-10 show that the higher the saturation, the smaller $I(T_i; X)$ at the last layer compared to the patch-shuffling. At the same time, the mutual information curve of $I(T_i; Y)$ shows a rather complicated situation due to the two-sided effects of the saturation and patch-shuffling setting.

V. Discussion

Recall the previous formulas (8) in the KDE method and (10) in the binning method for estimating the mutual information. We simplify (8) and have

$$I(T_i; X) = - \log \frac{1}{N} - \frac{1}{N} \sum_j \log \sum_k \exp \left( - \frac{1}{2} \frac{||t_i^{(j)} - t_i^{(k)}||^2}{\gamma^2} \right).$$

(14)

It is not difficult to find that $||t_i^{(j)} - t_i^{(k)}||^2$ is the key part of (14). Namely, the Euclidean distance between $t_i^{(j)}$ and $t_i^{(k)}$ decides the value of the mutual information. Meanwhile, when calculating $I(T_i; X)$ by the binning method the entropy of $X$ is calculated. If the bin size is specified, the empirical distribution depicted by sampling data points, the foundation
of the entropy, is closely related to the Euclidean distance between $t_i^{(j)}$ and $t_k^{(k)}$.

Specifically, given $N$ examples $\{x_j\}_{j=1,\ldots,N}$ sampled from $P(X)$ and corresponding $N$ $i$-th middle layer’s output $\{t_i^{(j)}\}_{j=1,\ldots,N}$, the mutual information $I(T_i;X)$ calculated by the KDE method is closely related to the Euclidean distance between $t_i^{(j)}$ and $t_k^{(k)}$. The farther two data points are from each other, the larger the mutual information $I(T_i;X)$ will be. The fact is the same for the binning method, the empirical distribution will be closer to the uniform distribution as long as a farther distance between $t_i^{(j)}$ and $t_k^{(k)}$, and the mutual information calculated by the binning method will be larger, which is shown in Fig. 6(a) and (b).

$I(T_i;Y)$ is mostly similar to $I(T_i;X)$, but is not only related to the overall distribution of $T_i$, but also influenced by the distribution of $T_i$ within each class. When the distribution of the overall $T_i$ is determined (i.e., $H(T_i)$ is known), for each class, the closer the distance between $t_i^{(j)}$ and $t_i^{(k)}$ within this class, the smaller the $H(T_i|Y)$, the larger the $I(T_i;Y)$ estimated by KDE and binning methods, which is a little different from $I(T_i;X)$ and shown in Fig. 6(c) and (d).

According to the above analysis and Fig. 6, the mutual information $I(T_i;X)$ and $I(T_i;Y)$ computed by the KDE and the binning methods can directly or indirectly portray the geometric properties of the $i$-th middle layer’s output $T_i$.

VI. CONCLUSION

In this work, we explored the middle layer’s information plane of models during training with different training methods, on different types of examples, and on different CNN architectures. We also investigate the information curves while NT-CNNs and AT-CNNs receive the input that suffers from information distortion. We preliminarily study the mechanisms behind adversarial examples and adversarial training from the mutual information perspective. With different training methods, CNNs can be seen as information extractors with different information extraction biases. The phenomena observed in previous experiments [11], [15]–[17] are also reflected from the mutual information perspective. The KDE and binning estimators are not only a measure of the mutual information but also can depict the geometric properties of the middle layer’s output based on the Euclidean distance and empirical distribution. In the future, we will utilize more accurate mutual information estimators to obtain more revealing results. Besides, it is also significant to explore adversarial examples and DNNs’ robustness from the geometric properties of the middle layer’s output. The resistance to adversarial examples in stochastic DNNs [21], [22] also deserves our attention [47].

APPENDIX A

FOR DISCRETE VARIABLES

For the sake of readability, we suppress the layer index $i$. For the discrete random variable $X$, and the discrete variable $T = f(X)$, we have $H(T | X) = 0$, and $I(T; X) = H(T)$ as long as $H(T)$ is finite. The expression of $I(T; X)$ is as follows

$$I(T; X) = H(T) - H(T | X)$$

$$= - \sum_{t \in T} p(t) \log p(t) + \sum_{x \in X} p(x) \sum_{t \in T} p(t | x) \log p(t | x).$$

(15)

Because $T$ is a deterministic function of $X$, when $X = x$, the probability density function $p(t | x)$ is

$$p(t | x) = \begin{cases} 1, & t = f(x), \\ 0, & t \neq f(x). \end{cases}$$

(16)

We define $0 \log 0 = 0$, so $H(T | X)$ is

$$H(T | X) = - \sum_{x \in X} p(x) \left( p(f(x) | x) \log p(f(x) | x) \right)$$

$$= - \sum_{x \in X} p(x) \left( \log 1 \right) = 0,$$

and $I(T; X) = H(T)$.
Fig. 7. The plot of probability density function $p(t \mid x)$.

For the continuous random variable $X$, and the continuous variable $T = f(X)$, we have $H(T \mid X) = -\infty$, and $I(T; X) = +\infty$ as long as $H(T)$ is finite. The expression of $I(T; X)$ is as follows

$$I(T; X) = H(T) - H(T \mid X) = -\int p(t) \log p(t) dt + \int p(x) dx \int p(t \mid x) \log p(t \mid x) dt.$$  \hfill (18)

The expression of $H(T \mid X)$ is as follows

$$H(T \mid X) = -\int p(x) dx \int p(t \mid x) \log p(t \mid x) dt.$$  \hfill (19)

As shown in Fig. 7 while $X = x$, we assume the distribution of $p(t \mid x)$ is as following

$$p(t \mid x) = \begin{cases} \frac{1}{\Delta t}, & |t - f(x)| \leq \frac{1}{2} \Delta t, \\ 0, & |t - f(x)| > \frac{1}{2} \Delta t, \end{cases}$$  \hfill (20)

and $\Delta t$ is a very small and positive value. For the probability density function $p(x)$, we have

$$\int p(x) dx = 1.$$  \hfill (21)

After plugging (20) and (21) into (19), we have

$$H(T \mid X) = -\int p(x) dx \int p(t \mid x) \log p(t \mid x) dt = -\lim_{\Delta t \rightarrow 0^+} \left( \Delta t \cdot \frac{1}{\Delta t} \cdot \log \frac{1}{\Delta t} \right) = -\infty.$$  \hfill (22)

This means that $H(T \mid X) \rightarrow -\infty$ and $I(T; X) \rightarrow +\infty$ while $p(t \mid x)$ approaches the delta function.
Z. Goldfeld, E. van den Berg, K. H. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polanskiy, “Estimating information flow in deep neural networks,” in Proc. Int. Conf. Mach. Learn. (ICML), vol. 97, pp. 2299–2308, 2019.

N. Carlini and D. A. Wagner, “Towards evaluating the robustness of neural networks,” in Proc. IEEE Symp. Secur. Privacy, pp. 39–57, 2017.

A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world.” 2016.

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “DeepFool: A simple and accurate method to fool deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 2574–2582, 2016.

Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, “Boosting adversarial attacks with momentum,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 9185–9193, 2018.

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2015.

F. Liao, T. Pang, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” IEEE Trans. Evol. Comput., vol. 23, no. 5, pp. 828–841, 2019.

P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, “ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models,” in Proc. ACM Workshop Artif. Intell. Secue., pp. 15–26, 2017.

W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2018.

A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, “Adversarial training for free!,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), pp. 3353–3364, 2019.

D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, “You only propagate once: Accelerating adversarial training via maximal principle,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), pp. 227–238, 2019.

C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Li, “FreeLB: Enhanced adversarial training for natural language understanding,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

T. M. Cover and J. A. Thomas, Elements of Information Theory (2. Ed.), 2006.

A. Kolchinsky and B. D. Tracey, “Estimating mixture entropy with pairwise distances,” Entropy, vol. 19, no. 7, p. 361, 2017.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2323, 1998.

A. Krizhevsky, “Learning multiple layers of features from tiny images,” in Proc. Int. Conf. Artif. Intell. Stat. (AISTATS), vol. 15, pp. 215–223, 2011.

S. Zagoruyko and N. Komodakis, “Wide Residual Networks,” in Proc. Br. Mach. Vis. Conf. (BMVC), 2016.

I. Korsunova, D. Stutz, A. A. Alemi, O. Wiles, and S. Gowal, “A closer look at the adversarial robustness of information bottleneck models,” CoRR, vol. abs/2107.05712, 2021.

Jiabei Zhang received the B.E. degree in information management & information system from Anhui University of Technology, Anhui, China, in 2020. He is currently pursuing the master’s degree in computer science with the School of Information Science and Engineering, Yunnan University, Kunming 650500, China. His research interests include deep learning, AI security, and explainable AI.

Jinde Cao (Fellow, IEEE) received the B.S. degree from Anhui Normal University, Wuhu, China, the M.S. degree from Yunnan University, Kunming, China, and the Ph.D. degree from Sichuan University, Chengdu, China, all in mathematics/applied mathematics, in 1986, 1989, and 1998, respectively. He is an Endowed Chair Professor, the Dean of the School of Mathematics, the Director of the Jiangsu Provincial Key Laboratory of Networked Collective Intelligence of China, and the Director of the Research Center for Complex Systems and Network Sciences at Southeast University. Prof. Cao was a recipient of the National Innovation Award of China, the Gold medal of the Russian Academy of Natural Sciences, the Obada Prize, and the Highly Cited Researcher Award in Engineering, Computer Science, and Mathematics by Thomson Reuters/Clarivate Analytics. He is elected as a member of the Academy of Europe, a foreign member of the Russian Academy of Sciences, a foreign member of the Russian Academy of Engineering, a member of the European Academy of Sciences and Arts, a foreign fellow of the Pakistan Academy of Sciences, a fellow of African Academy of Sciences, a foreign member of the Lithuanian Academy of Sciences, and an IASCYS academician.

Dan Xu received the M.Sc. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 1993 and 1999, respectively. She is currently a Professor at the School of Information Science and Engineering, Yunnan University, Kunming, China. She has authored or co-authored over 100 papers in refereed international journals. Her research interests include different aspects of image-based modeling and rendering, image processing and understanding, computer vision.

Jinhua Qian received the M.S. degree from Yunnan University, Kunming, China, in 2005, and the Ph.D. degree from Yunnan University, Kunming, China, in 2010. He is currently a professor at Computer Science and Engineering Department of Yunnan University, Kunming China. From 2017 to 2021, he was a Postdoctoral Research Fellow with the Department of Automation, Southeast University, Nanjing, China. He has authored or co-authored over 60 papers in refereed international journals. His current research interests include computer vision, image processing, and cultural computing.