On the Dynamics of Adversarial Input Attacks

Yujie Ji
Lehigh University
yuj216@lehigh.edu

Ting Wang
Lehigh University
inbox.ting@gmail.com

***Abstract*—An intriguing property of deep neural networks (DNNs) is their inherent vulnerability to adversarial inputs, which are maliciously crafted inputs to trigger DNNs to misbehave. The threats of adversarial inputs significantly hinder the application of DNNs in security-critical domains. Despite the plethora of work on adversarial input attacks and defenses, many important questions remain mysterious, including (i) How are adversarial inputs crafted to trigger targeted DNNs to misbehave? (ii) How are adversarial inputs generated by varied attack models different in their underlying mechanisms? (iii) How are complicated DNNs more vulnerable to adversarial inputs? (iv) How are existing defenses often ineffective against adaptive attacks? (v) How are transferable adversarial inputs different from non-transferable ones?**

This work represents a solid step towards answering the above key questions. Rather than focusing on the *static* properties of adversarial inputs from an input-centric perspective (i.e., whether the given input can deceive a targeted DNN), we conduct the first study on their *dynamic* properties from a DNN-centric perspective (i.e., how the targeted DNN reacts to a given adversarial input). Specifically, using a data-driven approach, we investigate the information flows of normal and adversarial inputs within varied DNN models and conduct in-depth comparative analysis of their discriminative patterns. Our study sheds light on the aforementioned questions, and points to several promising directions for designing more effective defense mechanisms.

I. INTRODUCTION

Recent years have witnessed the abrupt advances in deep learning [1], which have led to breakthroughs in a number of long-standing artificial intelligence tasks (e.g., image classification, speech recognition, and even playing Go [2]).

However, designed to model highly non-linear, non-convex functions, deep neural networks (DNNs) are inherently vulnerable to adversarial inputs, which are maliciously crafted samples to trigger DNNs to misbehave [3]. Figure 1 shows a set of examples on the CIFAR10 dataset: the first row shows the original images, which are correctly recognized by a DNN; the following rows show the adversarial inputs (generated by five different attack models), which deceive the same DNN to misclassify, although the difference between the normal and adversarial inputs is indiscernible to human vision. With the increasing use of DNN-powered systems in security-critical domains, adversaries have strong incentive to manipulate such systems via forcing misclassification of inputs: web content filters that employ DNNs to discriminate inappropriate content may be bypassed [4]; biometric authentications that apply DNNs to validate human faces may be manipulated to allow improper access [5]; and autonomous vehicles that use DNNs to detect traffic signs may be misled to crashing.

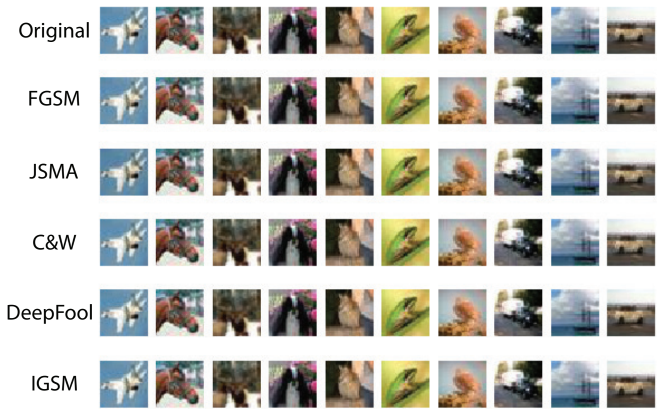

Figure 1: Examples of adversarial inputs on the CIFAR10 dataset. The first row shows the original images, and the following rows show the adversarial inputs generated by five different attack models.

The phenomena of adversarial inputs have attracted intensive research from the data science communities. One line of work focuses on developing new attack models [6]–[10], most of which attempt to find the minimum possible perturbation to normal inputs to deceive targeted DNNs. Another line of work attempts to improve DNN resilience against such attacks [6], [7], [11], [12] by augmenting training data [6], modifying objective functions [3], [13], [14], or knowledge transfer [12]. However, as shown in [10], [15], the defense-enhanced models, once deployed, can often be fooled by adaptively engineered inputs or by new attack variants.

Despite the plethora of existing work, we still lack sufficient understanding about the crucial properties of adversarial inputs. A number of important questions remain mysterious, such as: (i) How are adversarial inputs crafted to force DNNs to misclassify? (ii) How are adversarial inputs generated by varied attack models different in their underlying mechanisms? (iii) How are complicated DNNs more vulnerable to adversarial input attacks than simple DNNs? (iv) How are existing defenses (e.g., defensive distillation) often vulnerable to adaptive attacks? (v) How are transferable adversarial inputs different from non-transferable ones?

This work represents a solid step towards answering these key questions. We take a route completely different from existing work: instead of focusing on the static properties of adversarial inputs from an input-centric perspective (e.g., whether the given input can deceive a targeted DNN), we study the dynamic properties of adversarial inputs from a DNN-centric perspective (e.g., how the targeted DNN reacts

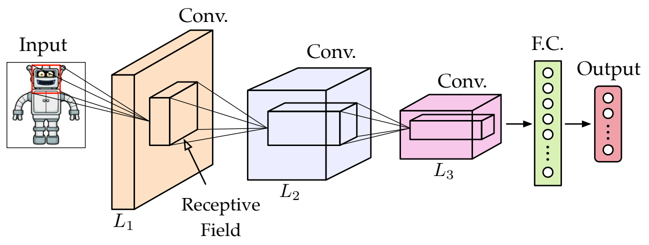

Figure 2: Illustration of a convNet architecture, which consists of an input layer, a sequence of convolution layers, a fully-connected layer, and an output layer.

to a given input, being normal or adversarial). To this end, using a data-driven approach, we investigate the information flows triggered by normal and adversarial inputs within the targeted DNN and conduct in-depth comparative studies of their discriminative patterns.

Our main findings are summarized as follows.

- The information flows of adversarial inputs deviate from that of normal inputs, while the attack process essentially corresponds to shifting the information flows away from normal inputs towards adversarial inputs.
- Adversarial inputs generated by varied attack models lead to drastically different information flows, implying that multiple defense or detection methods might be necessary to mitigate different attacks.
- It is easier to cause information flows to shift from normal inputs on more complicated DNN, indicating that more complicated models tend to be more vulnerable to adversarial input attacks.
- Many existing defenses, while effective against non-adaptive attacks, are ineffective to prevent adaptive attacks from shifting the information flows.
- The information flows of transferable adversarial inputs deviate further from normal inputs, compared with non-transferable ones, suggesting that training on ensemble models might be necessary to find transferable adversarial inputs.

All the source code of this paper will be released on GitHub after the double-blind review is complete.

II. Background

In this section, we introduce a set of concepts and assumptions used throughout the paper.

A. Neural Networks

Without loss of generality, in our studies, we focus on convolutional neural networks (ConvNets), which are used in a range of applications (e.g., image classification), while our method can be extended to other types of DNNs as well (e.g., recurrent neural networks).

As illustrated in Figure 2, a ConvNet architecture comprises an input and an output layer, as well as multiple hidden layers. The hidden layers typically consist of convolutional layers, pooling layers, fully-connected layers, and normalization layers. In particular, each convolutional layer applies a convolution operation to its input and passes the output to the next layer; each neuron only processes a local region of the input (i.e., its receptive field). The output of each convolutional layer is a set of feature maps, each corresponding to a filter. As each convolutional layer is often followed by non-linear (e.g., ReLU) and pooling (e.g., average or max pooling) transformations, we refer to a convolutional layer and its subsequent transformation layers together as a conv layer.

B. Attack Models

In adversarial input attacks, given a targeted DNN model f and a normal input x, the attacker evaluates the sensitivity of f's output with respect to x, and determines the minimum perturbations r (e.g., by flipping a few pixels) to get an adversarial version of x, where $\hat{x} = x + r$, which forces f's misclassification, i.e., $f(x) \neq f(\hat{x})$. To maximize the attack evasiveness, the perturbation r is often imperceptible to human vision, i.e., $||r||_p$ is small for certain norm p (e.g., L^2 norm). A set of sample adversarial inputs are shown in Figure 1.

We differentiate two types of attacks. In *targeted attacks*, the attacker's goal is to enforce f to classify $\hat{x}$ into a specific class $\hat{y}$; in *untargeted attacks*, the attacker's goal is to simply enforce $\hat{x}$'s misclassification (i.e., $f(x) \neq f(\hat{x})$).

In our comparative studies, we will consider a variety of state-of-the-art adversarial attack models.

*Fast Gradient Sign Method (*FGSM*):* FGSM [6] is an one-shot crafting attack. It computes the gradient of the loss function ℓ with respect to the input image x and determines distortion r base on the gradient signs. For a given perturbation magnitude ϵ, FGSM is defined as:

$$\hat{x} = x + \epsilon \cdot \text{sign}(\nabla \ell(f(x), x)) \tag{1}$$

*Iterative Gradient Sign Method (*IGSM*):* IGSM [16] is the iterative version of FGSM. In each iteration, instead of directly taking a single step of length ϵ in the direction of the gradient sign, multiple smaller steps α are taken, and the resulting adversarial input is clipped by the same ϵ. Let $\hat{x}_0 = x$, and then the i-th iteration is defined as:

$$\hat{x}_i = \text{clip}_{x,\epsilon}\{\hat{x}_{i-1} + \alpha \cdot \text{sign}(\nabla \ell(f(\hat{x}_{i-1}), \hat{x}_{i-1}))\} \tag{2}$$

*Jacobian-based Saliency Map Attack (*JSMA*):* JSMA [9] uses an *adversarial saliency map* to guide the crafting. This map estimates the sensitivity of each input element with respect to f's classification. Unlike FGSM and IGSM, JSMA is a *targeted* attack. In each iteration, the pixels that maximally increase f's predicted probability for the targeted class and decrease f's probability for other classes are modified.

DeepFool: DeepFool [17] is an untargeted iterative attack optimized for the L^2 norm ($p = 2$). It approximates f using linear decision boundaries, and then searches for the minimum distortions r to cross that boundary (i.e., $f(x+r) \neq f(x)$).

*Carlini & Wagner (*C&W*):* C&W [18] is considered the state-of-the-art adversarial attack model. C&W uses a different form of loss function based on f's unnormalized outputs (logits). Let $Z(x)_i$ denotes the logit for the i-th class, and κ as an adversarial confidence controller. Essentially, C&W attack minimizes the following quantity:

$$||x - \hat{x}||_p + c \cdot \max(Z(\hat{x})_y - \max\{Z(\hat{x})_i : i \neq y\}, -\kappa) \quad (3)$$

III. Information Flow Model

To understand the dynamic properties of adversarial inputs, we take a DNN-centric perspective, i.e., how the targeted DNN reacts to a given input, being normal or adversarial. Specifically, we measure the information flows generated by different inputs within various DNN models and conduct in-depth comparative studies of their patterns. Next, we detail our approach of modeling information flows.

A. Mutual Information

Consider a ConvNet f comprising a sequence of K conv layers, where the output of k-th layer consists of n_k feature maps $\{m_i^{(k)}\}_{i=1}^{n_k}$. Let x be a given input (e.g., an image) to f. As x is of multiple channels (e.g., RGB), we may also consider x as a set of n_s feature maps $\{m_i^s\}_{i=1}^{n_s}$. To understand x's dynamic properties, i.e., how does f react to x, we will quantify x's information flow going through f. We thus measure the mutual information (MI) between each feature map and x. Intuitively, the MI of two random variables Z and Z', $I(Z; Z')$, captures the amount of information obtained about Z via observing Z' (and vice versa), which is formally defined as:

$$I(Z; Z') = \sum_{z \in Z} \sum_{z' \in Z'} p(z, z') \log \left(\frac{p(z, z')}{p(z)\, p(z')} \right)$$

In our setting, to measure the MI of two feature maps m and m', we distribute the values of m (and m') into the same set of buckets and treat m (and m') as a discrete distribution. Specifically, we first perform bucketization on m (and m'). We assume that m and m' are of the same size; otherwise, we first perform downsampling (max pooling in particular) over the larger feature map to ensure they are of equal size. Let $v_{\min}$ and $v_{\max}$ respectively be the minimum and maximum values in m (or m''). We divide the interval $[v_{\min}, v_{\max}]$ evenly into B buckets ($B = 8$ in our implementation) and replace each value v in m (or m') with its bucket number:

$$\mathrm{bid}(v) = \left\lceil \frac{B(v - v_{\min})}{v_{\max} - v_{\min}} \right\rceil$$

We then consider m (and m') as a sequence of data points sampled from a distribution over these buckets, and compute their joint distribution $p(m, m')$ as well as their marginal distributions $p(m)$ and $p(m')$.

For simplicity of notation, we use $S^{(k)}(i, j)$ to denote the MI $I(m_i^{(k)}, m_j^s)$. We can populate an $n_k \times n_s$ matrix $S^{(k)}$ with the i-th row and j-th column element as $S^{(k)}(i, j)$.

The symbols used in this paper are summarized in Table I.

Symbol	**Definition**
f	DNN model
x	normal input
y	x's ground-truth class
$\hat{x}$	adversarial input
n_s	number of feature maps in input layer
n_t	number of feature maps in output layer
$m_i^{(k)}$	the i-th feature map of the k-the conv layer
$S^{(k)}$	source MI matrix of the k-th conv layer
$T^{(k)}$	target MI matrix of the k-th conv layer

Table I. Symbols and notations.

Further, to get a complete view of x's information flows, we will also quantify the MI between each feature map and x's output y. One may simply use the output of f's last layer as y. However, in the setting of classification, f's last layer is typically a softmax layer, whose output is of limited information. Instead, we use the output of f's last conv layer as y, which consists of a set of n_t feature maps $\{m_{(k)}^t\}_{k=1}^{n_t}$. We use $T^{(k)}(i, j)$ to denote the MI $I(m_i^{(k)}, m_j^t)$, while $\{T^{(k)}(i, j)\}_{i,j}$ populate an $n_k \times n_t$ matrix $T^{(k)}$.

B. Information Paths

We refer to $S^{(k)}$ and $T^{(k)}$ respectively as the source and target MI matrices of the k-th layer. Equipped with $S^{(k)}$ and $T^{(k)}$, we are able to depict the "information paths" (IPs) from a given input x to its output y in a layer-wise manner, that is, how the feature maps at the k-th layer capture the information in x and transform it into y.

Note that our approach is inspired by the information bottleneck methods [19], [20]. However, different from [19], [20], which treats inputs as individual data points, we consider each input as a discrete distribution, which allows us to investigate the information flow at the level of individual inputs.

More specifically, we construct a set of IPs:

- *Input information path* (IIP) measures the relevance of the feature maps at each layer with the input, defined as the sequence of $\{(k, \mu_s^{(k)})\}_k$, where $\mu_s^{(k)}$ is the mean of $S^{(k)}$.
- *Output information path* (OIP) measures the relevance of the feature maps at each layer with the output, defined as the sequence of $\{(k, \mu_t^{(k)})\}_k$, where $\mu_t^{(k)}$ is the mean of $T^{(k)}$.
- *Input-Output contrast* (IOC) correlates the input and output information paths at each layer, defined as a sequence of $\{(\mu_s^{(k)}, \mu_t^{(k)})\}_k$.

We compute the distance of two PIs (e.g., two IIPs) using their average difference across all the layers. Formally, let $\{(k, \mu_1^{(k)})\}_k$ and $\{(k, \mu_2^{(k)})\}_k$ denote two IPs. Their distance is calculated as: $\frac{1}{K} \sum_k |\mu_1^{(k)} - \mu_2^{(k)}|$.

C. Aggregated Information Paths

Further, for our empirical study, we also devise the model of aggregated IPs to summarize the IPs of a set of similar inputs (e.g., adversarial inputs generated by the same attack).

To this end, we apply the method of Gaussian process regression [25]. We consider each MI measure (e.g., $\mu_s^{(k)}$) as a random variable and assume that a collection of such random variables indexed by k follow a multivariate normal distribution; therefore, each IP is a random sample from a

Gaussian process. Under this model, we use the mean of the estimated Gaussian process to represent the aggregated IP.

IV. Analysis

Equipped with the aforementioned measurement tools, we conduct one of the first empirical studies on the dynamic properties of adversarial inputs. Our study is designed to provide a new perspective on a set of key questions about adversarial inputs.

A. Experimental Setting

Datasets and DNN Models: We mainly use the CIFAR10 dataset [21], which consists of 32×32 color images drawn from ten classes (e.g., 'airplane', 'automobile', 'bird'). The dataset is split into 50K training and 10K testing samples. The dataset has been centered and normalized to $[-1, 1]$.

We use the DNN model in [22], which attains the accuracy of 90.24% on CIFAR10. The detailed architecture of this DNN is specified in Appendix.

Attack Models: To make a fair comparison among different attack models, we make the following assumptions. First, we focus on untargeted attacks. For targeted attack models (e.g., JSMA), we select the class $\hat{y}$ (different from $\boldsymbol{x}$'s ground-truth class y) that requires the minimum perturbation as its targeted class. Second, we require different attack models to have similar perturbation magnitude, by setting the same threshold on the relative perturbation magnitude $||r||_2/||x||_2$ across different models.

Specifically, for FGSM, we set $\epsilon = 0.025$; for IGSM, we set $\alpha = 0.002$ and $\epsilon = 0.025$; for JSMA, we set the magnitude of each update as 0.1 and allow only less than 10% of the pixels to be perturbed; for C&W, we use l_2 norm for distortion evaluation and set learning rate $c = 0.25$ and $\kappa = 0$. For FGSM, IGSM, JSMA, and C&W attacks, we use the implementation of Cleverhans [23]. For DEEPFOOL, we use the implementation of Foolbox [24].

Measurements: In our study, given a DNN model f and an input x, we collect x's feature maps generated at f's each conv layer, measure its source and target MI matrices $\{S^{(k)}, T^{(k)}\}_k$, and compute x's IPs (IIP, OIP, and IOC) within f.

B. Experimental Results

In the following, we present the results of our empirical study and report our findings.

Q1: How are adversarial inputs crafted to trigger DNNs to misbehave?

In the first set of experiments, to understand the process of crafting adversarial inputs, we study the information flows corresponding to a series of inputs: $\{\hat{x}_0, \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$, where $\hat{x}_0$ and $\hat{x}_n$ are the original and adversarial inputs respectively, while $\{\hat{x}_i\}_{i=1}^{n-1}$ are a set of intermediate inputs generated during the iterative attack process.

Setting: Without loss of generality, here we use the JSMA attack as an example. In JSMA, at each iteration, two pixels that maximally influence the classification output are perturbed. We record the intermediate input after every iteration and measure its information flow.

Findings: Figure 3 shows the results of applying the JSMA attack on a randomly selected input. The plots in the top row show its IIP, OIP, and IOC respectively. Observe that across all three cases, during the crafting process (i.e., i increases from 0 to n), the IPs of $\hat{x}_i$ gradually deviate from that of $\hat{x}_0$ (normal input) and converge towards that of $\hat{x}_n$ (adversarial input). We also measure the overall deviation between the IIPs and OIPs of $\hat{x}_i$ and $\hat{x}_0$, with results shown in the bottom row of Figure 3. Observe that as the perturbation proceeds, the IPs of $\hat{x}_i$ gradually deviate from $\hat{x}_0$. We can thus conclude that (i) normal and adversarial inputs feature different IPs, and (ii) the adversarial crafting process corresponds to gradually shifting the normal IPs towards adversarial ones.

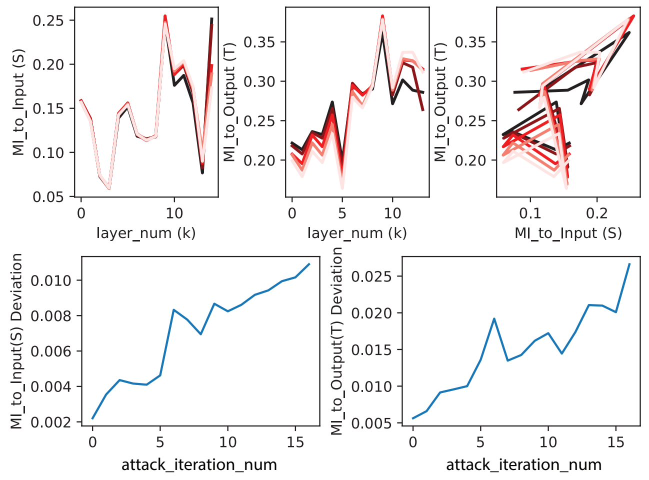

Figure 3: Top row: the IIP, OIP, and IOC of a randomly selected input. In each plot, the black line represents the normal input $\hat{x}_0$, the lines with lighter colors correspond to larger i, the line with the lightest color represents the adversarial input $\hat{x}_n$. Bottom row: the overall deviation between the IIPs and OIPs of $\hat{x}_i$ and $\hat{x}_0$.

Q2: How are adversarial inputs generated by varied attacks different in their underlying mechanisms?

Despite the same objective (i.e., crafting adversarial inputs with minimum possible perturbation), varied attack models achieve this objective via fairly different routes (e.g., one-shot perturbation versus iterative perturbation). Here, we contrast different attack models by examining the IPs of their generated adversarial inputs.

Setting: We generate adversarial inputs using four different attacks, namely, FGSM, IGSM, DEEPFOOL, and C&W, on the same set of 200 normal inputs, which form the profiles of different attack models. The first block of Table II summarizes the statistical properties of the generated adversarial inputs. To make a comparison, we also randomly select 200 normal inputs from the training set of CIFAR10 to form the profile of normal inputs. We then compute the aggregated IPs of the adversarial inputs by each attack.

Findings: Figure 4 compares the aggregated IIPs, OIPs, and IOCs of the normal inputs and the adversarial inputs generated by each attack model. We have the following key observations.

First, the aggregated IIPs of different inputs are fairly similar to each other. Note that due to the minimality principle underlying all the attacks (i.e., with the goal of minimizing the perturbation magnitude), the adversarial inputs and the normal inputs are often indiscernible. Thus, this observation indicates that the adversarial feature maps at each layer are similar to their normal counterparts. Second, the adversarial inputs generated by all the attacks indeed cause the OIPs to shift away from the normal inputs. This observation suggests that while the adversarial feature maps are only slightly different from their normal counterparts, the difference is significant from the perspective of the output. Also note that different attacks result in OIPs with varying deviation from the normal OIP. For example, the OIP of FGSM is much closer to the normal OIP, compared with other attacks. which is quantitatively verified in Table III. This difference also partially explains that effectively defending against a range of attacks requires combining multiple defense strategies. For example, in [26], two complementary defense strategies are proposed, one targeting attacks such as FGSM, and the other targeting attacks such as DEEPFOOL.

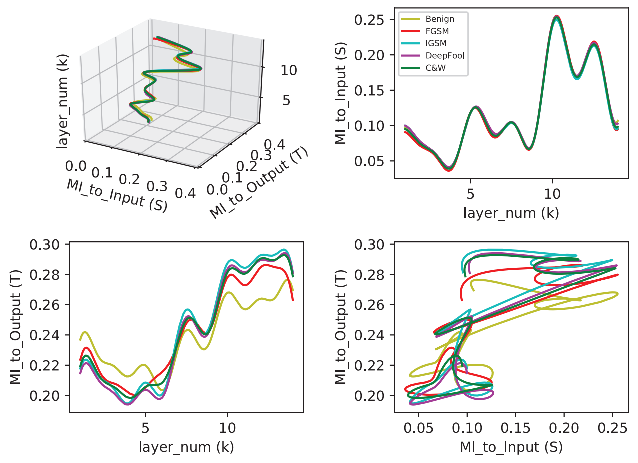

Figure 4: Aggregated IIPs, OIPs, IOCs of the normal inputs and the adversarial inputs generated by FGSM, IGSM, DEEPFOOL, and C&W attacks.

Q3: How are complicated DNN models more vulnerable to adversarial inputs?

Previous work (e.g., [17], [18]) suggests that it seems easier for the adversary to craft adversarial inputs on complicated datasets than simple ones. For instance, for the same attack to achieve similar attack success rates, much smaller perturbation is required on IMAGENET than CIFAR10 and MNIST. We speculate that the required smaller perturbation stems from both the higher dimensionality of the dataset and the more complicated architecture of the DNN model. In this set experiment, we focus on studying the influence of DNN model complexity on the crafting of adversarial inputs.

Setting: We consider a pair of models of similar architectures but different complexities. To this end, we compress the original model using the approach proposed in [27]. The compressed model retains the accuracy of the original model, and features a similar architecture (only the number of neurons in convolutional layers or fully-connected layers are reduced). Table IV compares the parameter numbers and the accuracy of the two models. It is observed that the compressed model attains accuracy comparable to the original model but with 10% fewer neurons.

We apply three different attacks (FGSM, JSMA and C&W) on the two DNNs. The statistics of the generated adversarial inputs are summarized in Table II. Note that the column of "distortion (l_2)" indicates that we enforce similar perturbation magnitude across different attacks. Similar to the previous experiments, we compute the IPs for individual adversarial inputs, and apply Gaussian process regression to compute the aggregated IPs.

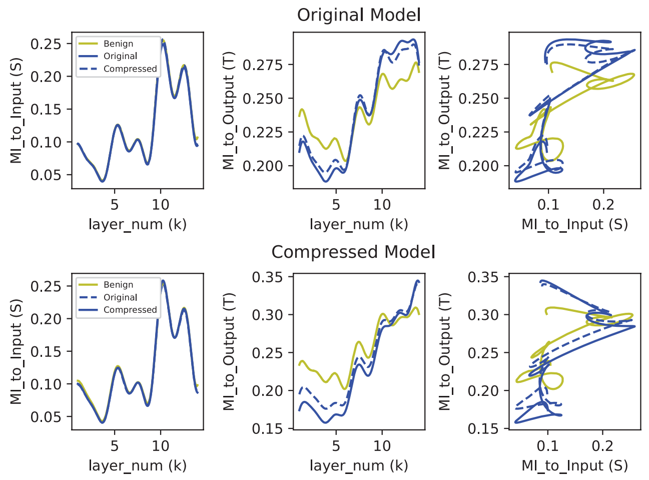

Figure 5: Aggregated IIPs, OIPs, and IOCs of normal inputs and adversarial inputs generated by JSMA. The solid blue line represents adversarial inputs targeting the inference model, while the dash blue line represents adversarial inputs targeting the other model.

Findings: Figure 5 shows the aggregated IPs of normal and adversarial inputs generated by JSMA. For comparison, with respect to a given DNN f (original or compressed), besides the IPs of adversarial inputs targeting f, we also compute the IPs of adversarial inputs targeting the other DNN (compressed or original). It is observed that the two sets of inputs have a larger gap between their IPs on the compressed model than on the original model, with the quantitative measurement shown in Table V. This observation indicates that adversarial inputs targeting the compressed model tend to attain more similar IPs, compared with adversarial inputs on the original model itself. Note that successful attacks on one model do not guarantee to work on the other model; yet, smaller deviation implies higher probability that adversarial inputs may transfer across

Model (Target)	Attack	Norm	Distortion	Distortion (l_2)	Confidence	Success Rate
Original	FGSM	l_∞	0.0263	0.0556	0.7148	0.4667
Original	IGSM	l_∞	0.0069	0.0102	0.5716	0.6333
Original	JSMA	l_0	0.0506	0.0452	0.3570	0.4833
Original	DEEPFOOL	l_2	0.0094	0.0094	0.4372	0.9833
Original	C&W	l_2	0.0433	0.0433	0.5185	0.8778
Compressed	FGSM	l_∞	0.0264	0.0566	0.9281	0.4457
Compressed	JSMA	l_0	0.0434	0.0429	0.4915	0.3261
Compressed	C&W	l_2	0.0548	0.0548	0.7707	1.0

Table II. Summary of adversarial inputs generated by different attack models on CIFAR10, targeting the original DNN or its compressed version. Note that the distortion is normalized (i.e., $||r||_p/||x||_p$, where p is determined by the attacks).

	FGSM	IGSM	DEEPFOOL	C&W
Input	0.0040	0.0043	0.0030	0.0036
Output	0.0117	0.0162	0.0157	0.0140

Table III. Difference of IIPs and OIPs between normal inputs and adversarial inputs generated by different attacks.

CIFAR10		
Model	**# Parameters**	**Accuracy**
Original	1,071,542	0.9024
Compressed	890,411	0.9078

Table IV. Comparison of original and compressed models.

Model	IIP Distance	OIP Distance	Average Distance
FGSM			
Original	0.0026	0.0066	0.0046
Compressed	0.0021	0.0028	0.0025
JSMA			
Original	0.0019	0.0039	0.0029
Compressed	0.0029	0.0116	0.0072
C&W			
Original	0.0010	0.0040	0.0025
Compressed	0.0015	0.0056	0.0036

Table V. Distance of IIPs and OIPs of adversarial inputs targeting original and compressed models.

Attack	Model (Target)	Model (Measurement)	Transferability
FGSM	Original	Compressed	0.8397
	Compressed	Original	0.86
JSMA	Original	Compressed	0.2137
	Compressed	Original	0.3579
C&W	Original	Compressed	0.1362
	Compressed	Original	0.25

Table VI. Percentage of transferrable adversarial inputs.

different models. To empirically validate our proposition, we measure the transferability of adversarial inputs targeting the two models, with results shown in Table VI, where adversarial inputs achieve higher transferability on the original model than on the compressed model. Similar phenomena are observed for adversarial inputs generated by FGSM and C&W attacks, which are shown in Figure 8 and Figure 9 in Appendix.

Q4: How are existing defenses mechanisms often vulnerable to adaptive attacks?

As pointed out in [15], most defenses are vulnerable to adaptive attacks, in which the adversary creates adversarial inputs tailored to both the targeted DNN model and the defensive model. We will investigate the reason behind this vulnerability from the perspective of IPs.

Setting: We take defensive distillation [12] for a case study, which is a popular defense algorithm. To defensively distill a DNN, one first trains a model with identical architecture on the training data, but replaces the softmax function with a smooth version during training (by dividing logits by some temperature parameter T):

$$f(x) = \left[\frac{e^{Z_i(x)/T}}{\sum_{l=0}^{N-1} e^{Z_l(x)/T}}\right]_{i\in[0,1,\ldots,N-1]} \quad (4)$$

where $f(x)$ is the prediction probability of the DNN, $Z_i(x)$ is the i-th logit of input x, and N is the number of classes.

Specifically, we first generate soft training labels, which are the outputs of all the training instances by the first step training. We then train the second model by feeding these soft training labels into training, instead of original one-hot encoded labels. This helps to train a second model with the same behavior like the first one, and the soft labels transfer additional information learned by the first model. In the inference step, we only use the second model, and reset $T = 1$ to obtain the final classification results. In our implementation, as in [12], we set $T = 40$. The generated defensively distilled model achieves 91.24% accuracy.

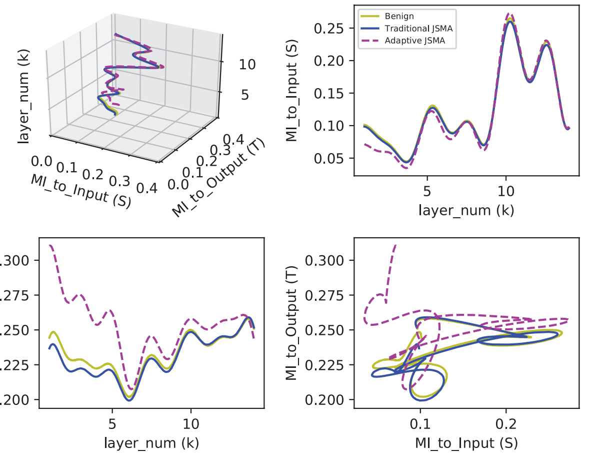

Figure 6: Aggregated IIPs, OIPs, and IOCs of normal inputs and adversarial inputs generated by the regular and adaptive JSMA attacks.

One reason that defensive distillation can mitigate adversarial inputs generated by certain attacks (e.g., JSMA) is as follows: it vanishes gradient of the DNN model, which makes it hard for attack mechanisms to perturb normal inputs according to the gradient information. However, Carlini *et al.* [10] defeat defensive distillation by adjusting JSMA attack by replacing the softmax function with (4) in the attack.

Following the setting of [10], we set the update magnitude as 0.4 and the percentage of changeable pixels as 20% for both the regular and adaptive JSMA attacks. The regular JSMA fails to generate any successful adversarial inputs, while the adaptive JSMA achieve 31.32% attack success rate on the defensively distilled model.

With the same procedure in Q2 and Q3, using Gaussian process regression, we compute the aggregated IPs of adversarial inputs (including successful and failed ones) generated by the regular and adaptive JSMA attacks on the defensively distilled DNN model.

Findings: The results are shown in Figure 6. Note that from the IP perspective, adversarial inputs generated by the adaptive JSMA deviate much further from normal inputs. Meanwhile, adversarial inputs crafted by the regular JSMA show IPs similar to normal ones. This difference is further quantitatively validated: the distance between the aggregated OIPs of adversarial and normal inputs is 0.0033 for the regular JSMA and 0.0219 for the adaptive JSMA. This large difference may partially explain why the adaptive JSMA can still successfully deceive defensively distilled DNN models. Recently, several defense mechanisms have been proposed to obfuscate gradient information [28]–[35]; yet, Anish *et al.* successfully circumvent all these attacks [36]. We consider investigating these attack and defense models from the IP perspective as our ongoing research.

Model	Accuracy	# Parameters
good_init	0.9024	1071542
deep_conv	0.838	2915114
vgg_like	0.9096	15001418

Table VII. Summary of models on CIFAR10 dataset.

Q5: How are transferable adversarial inputs different from non-transferrable ones?

One significant feature of adversarial inputs is that they are often misclassified by a variety of classifiers, which are of different architectures or trained on different training data [3]. This property is referred to as *transferability*, which can be utilized to launch black-box attacks [37], [38]. Yanpei *et al.* [39] proposed to craft transferable adversarial inputs by attacking ensemble models. Here we show why this approach works by comparing the IPs of adversarial inputs targeting different DNNs.

Setting: We build three different DNN models, "good_init" [22], "deep_conv" [40], and "vgg_like", which are summarized in Table VII. We then generate adversarial inputs targeting these DNNs, which are summarized Table VIII. We then compute their aggregated IPs. Note that we only consider normal inputs that can successfully generate adversarial inputs on all three DNNs.

Findings: Figure7 shows the aggregated IPs of adversarial inputs targeting three DNNs by JSMA. Observe that the OIPs of adversarial inputs targeting the inference model deviate further away from normal inputs, compared with that of adversarial inputs targeting a model different from this inference model. This observation is quantitatively validated in Table IX. Furthermore, the OIPs of transferrable adversarial inputs deviate much further compared with non-transferrable ones. Therefore, in order to create transferrable adversarial inputs, it is sensible to attack ensemble models (i.e., training on the three DNNs simultaneously). Similar phenomena are observed on adversarial inputs generated by FGSM and C&W, which are shown in Figure 10 and Figure 11 in Appendix.

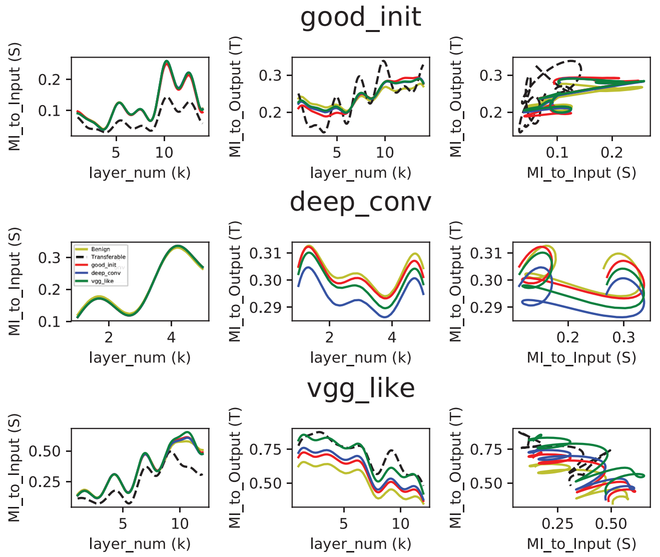

Figure 7: Aggregated IIPs, OIPs, and IOCs of adversarial inputs targeting three DNN models by the JSMA attack. Note that JSMA fails to generate transferable adversarial samples targeting "deep_conv".

V. Additional Related Work

Here we review two categories of related work, namely, adversarial deep learning and information flow models.

A. Adversarial Deep Learning

Lying at the core of many security-critical domains, machine learning systems are increasingly becoming the targets of malicious attacks [46]–[48]. Two primary threat models are considered in literature: (i) poisoning attacks, in which the attackers pollute the training data to eventually compromise the learning systems [49]–[51], and (ii) evasion attacks, in which the attackers modify the input data at test time to trigger the learning systems to misbehave [52]–[54]. Yet, for ease of analysis, most of the work assumes simple learning models (e.g., linear classifier, support vector machine, logistic regression) deployed in adversarial settings.

Addressing the vulnerabilities of deep learning systems to adversarial inputs is more challenging for they are designed to model highly nonlinear, nonconvex functions [55], [56]. One line of work focuses on developing new attacks against DNNs [6]–[10], most of which attempt to find the minimum possible modifications to the input data to trigger the systems to misclassify. The detailed discussion of representative attack models is given in § III. Another line of work attempts to improve DNNs resilience against such adversarial attacks [6], [7], [11], [12]. However, these defense mechanisms often require significant modifications to either DNN architectures

Attack	Model	Distortion	Distortion_L_2	Confidence	Success_Rate
FGSM	good_init	0.0263	0.0556	0.7148	0.4667
	deep_conv	0.0265	0.0559	0.7557	0.7176
	vgg_like	0.0266	0.0558	0.8399	0.6333
JSMA	good_init	0.0506	0.0452	0.3570	0.4833
	deep_conv	0.0591	0.0518	0.3699	0.7118
	vgg_like	0.0560	0.0495	0.3879	0.7722
C&W	good_init	0.0443	0.0443	0.5185	0.8778
	deep_conv	0.0669	0.0669	0.5118	1.0
	vgg_like	0.0679	0.0679	0.63	1.0

Table VIII. Summary of adversarial inputs targeting three DNN models.

Target / Inference	good_init	deep_conv	vgg_like
good_init	**0.017**	0.012	0.011
deep_conv	0.002	**0.009**	0.004
vgg_like	0.058	0.058	**0.165**

Table IX. Distance between the OIPs of normal inputs and adversarial inputs by JSMA attack.

or training processes, which may negatively impact the classification accuracy of DNNs. Moreover, as shown in § IV, the defense-enhanced models, once deployed, can often be fooled by adaptively engineered inputs or by new attack variants. To our best knowledge, this work represents an initial step to attack-agnostic defenses against adversarial attacks.

Finally, one active line of research explores the theoretical underpinnings of DNN robustness. For example, Fawzi *et al.* [57] explore the inherent trade-off between DNN capacity and robustness; Feng *et al.* [58] seek to explain why neural nets may generalize well despite poor robustness properties; and Tanay and Griffin [59] offer a theoretical explanation for the abundance of adversarial inputs in the input manifold space.

B. Information Flow Models

The underlying mechanisms of the effectiveness of DNN models remain mysterious today. Recent research has suggested using information flow models as tools to understand such mechanisms.

Mehta and Schwab [41] revealed the mapping between deep learning and variational renormalization groups in theoretical physics. Tishby and Zaslavsky [42] proposed a new view of DNNs as Markov chains of successive representations of inputs and introduced the concept of Information Plane to address the trade-off between the compression and prediction, and meanwhile suggested using Information Bottleneck (IB) [19] based learning mechanism to achieve the optimal representation of inputs.

Khadivi *et al.* [43] analyzed information flow across every two consecutive layers of a DNN by comparing their relative entropy and conditional entropy of information changes. Mototake and Ikegami [44] measured information flows of a DNN by calculating singular values and singular vectors of the Jacobian matrices between every two consecutive layers.

Shwartz-Ziv and Tishby [20] applied the IB theory to evaluate the DNN training process and claimed that DNN models first fit data and subsequently compress information during the Stochastic Gradient Descent (SGD) training process. However, there are debates [45] on several claims of this theory (e.g., in the case of full-connected ReLU).

Different from the previous work on information flow models, this work focuses on measuring and comparing the information flows caused by normal and adversarial inputs, and extracting their discriminative patterns. Thus, while the IB theory considers inputs as individual data points, we consider them as discrete distributions, which allow us to track the information flows of individual inputs.

VI. Conclusion

In this paper, we present the first empirical study on the dynamic properties of adversarial input attacks against DNN models. Specifically, using a data-driven approach, we measure the information flows of adversarial inputs within DNN models and conduct the in-depth comparative study on their discriminative patterns. Our study sheds light on a set of key questions about adversarial inputs, and also points to several promising directions for designing more effective defense mechanisms.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Sang-Hun, "Google's Computer Program Beats Lee Se-dol in Go Tournament," http://www.nytimes.com/2016/03/16/world/asia/korea-alphago-vs-lee-sedol-go.html, 2016.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *ArXiv e-prints*, 2013.

[4] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial Perturbations Against Deep Neural Networks for Malware Classification," *ArXiv e-prints*, 2016.

[5] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, 2016.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *ArXiv e-prints*, 2014.

[7] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvari, "Learning with a Strong Adversary," *ArXiv e-prints*, 2015.

[8] P. Tabacof and E. Valle, "Exploring the Space of Adversarial Images," *ArXiv e-prints*, 2015.

[9] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swamil, "The limitations of deep learning in adversarial settings," in *Euro S&P*, 2016.

[10] N. Carlini and D. Wagner, "Defensive Distillation is Not Robust to Adversarial Examples," *ArXiv e-prints*, 2016.

[11] S. Gu and L. Rigazio, "Towards Deep Neural Network Architectures Robust to Adversarial Examples," *ArXiv e-prints*, 2014.

[12] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *S&P*, 2016.

[13] U. Shaham, Y. Yamada, and S. Negahban, "Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization," *ArXiv e-prints*, 2015.

[14] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional Smoothing with Virtual Adversarial Training," *ArXiv e-prints*, 2015.

[15] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 3–14.

[16] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[17] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. EPFL-CONF-218057, 2016.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[19] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 1999.

[20] R. Shwartz-Ziv and N. Tishby, "Opening the Black Box of Deep Neural Networks via Information," *ArXiv e-prints*, 2017.

[21] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[22] A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," *Technical report, University of Toronto*, 2009.

[23] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.

[24] N. Papernot, N. Carlini, I. Goodfellow, R. Feinman, F. Faghri, A. Matyasko, K. Hambardzumyan, Y.-L. Juang, A. Kurakin, R. Sheatsley *et al.*, "cleverhans v2. 0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.

[25] J. Rauber, W. Brendel, and M. Bethge, "Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131

[26] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[27] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[28] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[29] A. Roy, C. Raffel, I. Goodfellow, and J. Buckman, "Thermometer encoding: One hot way to resist adversarial examples," 2018.

[30] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, M. E. Houle, G. Schoenebeck, D. Song, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," *arXiv preprint arXiv:1801.02613*, 2018.

[31] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[32] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[33] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[34] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint arXiv:1710.10766*, 2017.

[35] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," 2018.

[36] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *arXiv preprint arXiv:1802.00420*, 2018.

[37] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[38] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint*, 2016.

[39] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[40] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[41] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ASIACCS*, 2006.

[42] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *AISec*, 2011.

[43] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[44] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML*, 2012.

[45] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomput.*, vol. 160, no. C, pp. 53–62, 2015.

[46] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: Understanding and defending against poisoning of anomaly detectors," in *IMC*, 2009.

[47] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *KDD*, 2004.

[48] D. Lowd and C. Meek, "Adversarial learning," in *KDD*, 2005.

[49] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar, "Query strategies for evading convex-inducing classifiers," *J. Mach. Learn. Res.*, vol. 13, pp. 1293–1332, 2012.

[50] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015.

[51] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial Manipulation of Deep Representations," in *ICLR*, 2016.

[52] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *ArXiv e-prints*, 2015.

[53] J. Feng, T. Zahavy, B. Kang, H. Xu, and S. Mannor, "Ensemble Robustness of Deep Learning Algorithms," *ArXiv e-prints*, 2016.

[54] T. Tanay and L. Griffin, "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples," *ArXiv e-prints*, 2016.

[55] P. Mehta and D. J. Schwab, "An exact mapping between the variational renormalization group and deep learning," *arXiv preprint arXiv:1410.3831*, 2014.

[56] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Information Theory Workshop (ITW), 2015 IEEE*. IEEE, 2015, pp. 1–5.

[57] P. Khadivi, R. Tandon, and N. Ramakrishnan, "Flow of information in feed-forward deep neural networks," *arXiv preprint arXiv:1603.06220*, 2016.

[58] Y. Mototake and T. Ikegami, "The dynamics of deep neural networks," http://sacral.c.u-tokyo.ac.jp/pdf/mototake_AROB_2015.pdf, 2015.

[59] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *International Conference on Learning Representations*, 2018.

APPENDIX

Layer	Layer_Info	# Parameters
0	Input	0
1	Conv2D: 3×3, 'relu', 32	896
2	Conv2D: 3×3, 'relu', 32	9248
3	Conv2D: 3×3, 'relu', 32	9248
4	Conv2D: 3×3, 'relu', 48	13872
5	Conv2D: 3×3, 'relu', 48	20784
6	MaxPooling2D: 2 × 2	0
7	Dropout: 0.25	0
8	Conv2D: 3×3, 'relu', 80	34640
9	Conv2D: 3×3, 'relu', 80	57680
10	Conv2D: 3×3, 'relu', 80	57680
11	Conv2D: 3×3, 'relu', 80	57680
12	Conv2D: 3×3, 'relu', 80	57680
13	MaxPooling2D: 2 × 2 kernel	0
14	Dropout: 0.25	0
15	Conv2D: 3×3, 'relu', 128	92288
16	Conv2D: 3×3, 'relu', 128	147584
17	Conv2D: 3×3, 'relu', 128	147584
18	Conv2D: 3×3, 'relu', 128	147584
19	Conv2D: 3×3, 'relu', 128	147584
20	GlobalMaxPooling2D	0
21	Dropout: 0.25	0
22	FullyConnected: 500 , 'relu'	64500
23	Dropout: 0.25	0
24	FullyConnected: 10 , 'softmax'	5010

Table X. The architecture of the DNN used in our studies.

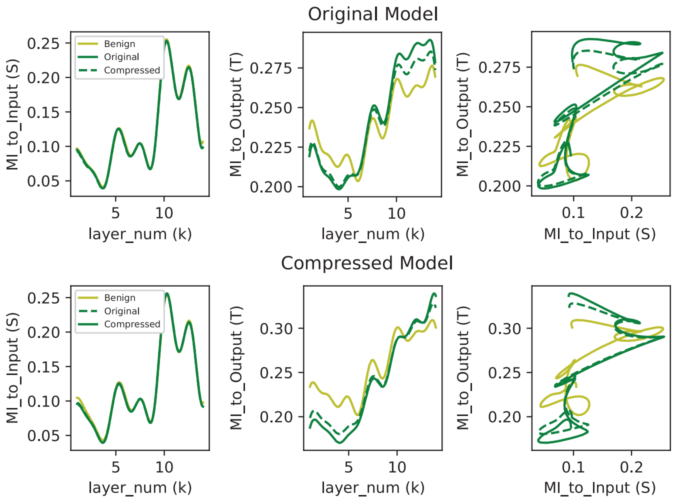

Figure 9: Aggregated IIPs, OIPs, and IOCs of normal inputs and adversarial inputs generated by C&W. The solid green line represents adversarial inputs targeting the inference model, while the dashed green line represents adversarial inputs targeting the other model.

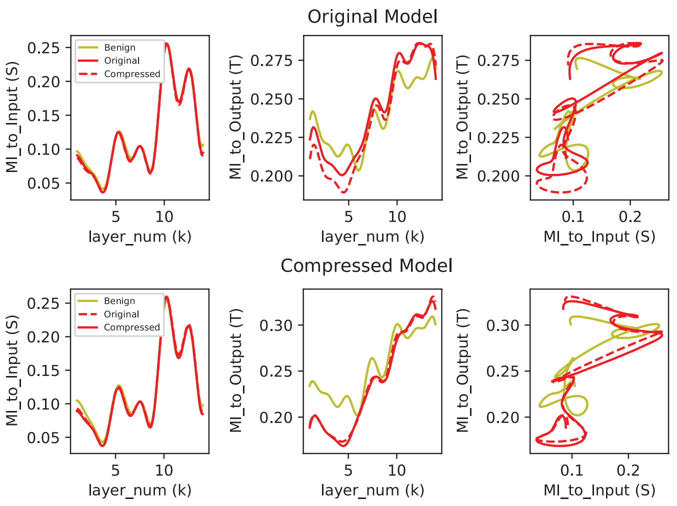

Figure 8: Aggregated IIPs, OIPs, and IOCs of normal inputs and adversarial inputs generated by FGSM. The solid red line represents adversarial inputs targeting the inference model, while the dashed red line represents adversarial inputs targeting the other model.

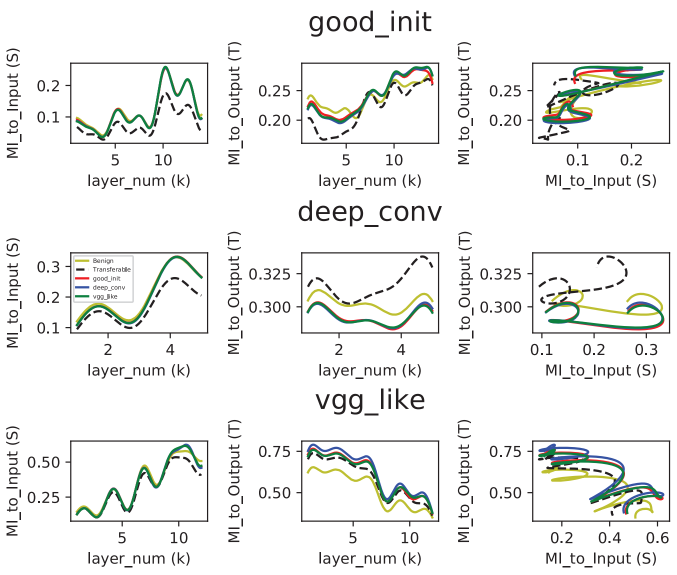

Figure 10: Aggregated IIPs, OIPs, and IOCs of adversarial inputs targeting three different models on three DNNs by FGSM attack.

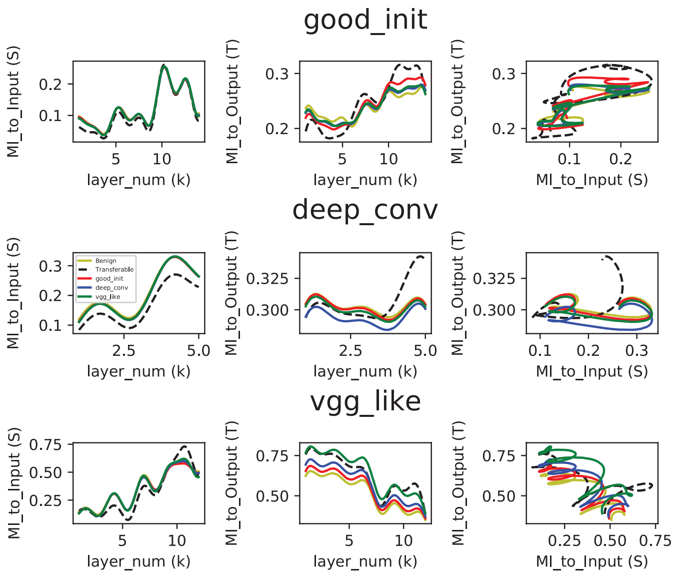

Figure 11: Aggregated IIPs, OIPs, and IOCs of adversarial inputs targeting three different models on three DNNs by C&W attack.