Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07387-y

ORIGINAL ARTICLE

Evaluation of adversarial attacks sensitivity of classifiers with occluded input data

Korn Sooksatra[1] · Pablo Rivas[1]

Received: 21 July 2021 / Accepted: 28 April 2022

Abstract
With the noteworthy achievements of deep learning models, there are transformative applications that aim at cost reduction and the improvement in human quality of life. Nevertheless, recent work aimed at testing a classifier's ability to withstand targeted and black-box adversarial attacks demonstrated that deep learning models, in particular, are brittle and lack certain robustness that makes them particularly weak, and ultimately leading to a lack of trust. For this specific area, a question arises concerning certain regions' sensitivity in the input space against adversarial perturbations for a classification model. This paper aims to study such a problem by looking into a Sensitivity-inspired Constrained Evaluation Method (SICEM) to deterministically evaluate how much a region of the input space is vulnerable to adversarial perturbations compared to other regions and also the entire input space. Our experiments suggest that SICEM can accurately quantify region vulnerabilities on MNIST and CIFAR-10 datasets.

Keywords Adversarial attacks · Adversarial learning · Adversarial robustness

1 Introduction

Deep learning models have positively transformed many areas , including computer vision [32], natural language processing [20] and cybersecurity [35]. However, [11, 28] discovered that by adding carefully crafted tiny perturbations to a clean input, a strong deep learning classifier could misclassify the input, and this input is called an adversarial example. More specifically, such an action is a so-called adversarial attack. Although some of the existing works [5, 21, 24, 36, 38] proposed defensive mechanisms against the attack, they failed to defend against some kinds of adversarial attacks. For example, network distillation as proposed in [24] fails to defend against the Carlini and Wagner attack [4]. Also, adversarial training proposed in [11, 21] cannot defend against adversarial examples that are harder than the ones used for the training. Consequently, adversarial attacks are still effective for deep learning models and are a hot topic in the machine learning area.

Furthermore, in deep-learning-based malware detection [12, 13, 31, 37], an adversary can perturb only a part of the input (i.e., an application) to let the application and the malware properly function, and we call this an incomplete input scenario. Motivated by this, we aim to find how much a given part of the input is relatively effective compared to the complete input or other parts of the input if there are multiple given parts for the adversary to choose. However, in this paper, we instead used image datasets for our experiments since we could provide a better visualization than using the dataset for malware detection.

Since in the case of L_0 attack, an adversary can perturb only a few attributes and therefore adds a considerable perturbation to those attributes instead, perturbing only a few attributes in a given input is possible to create an adversarial example. For instance, Papernot et al. [23] showed that only perturbing some attributes of an input to which the output is much sensitive can create an adversarial example. Further, Eykholt et al. [10] produced a sticker for a stop sign to successfully mislead a deep learning model to classify it as a speed limit sign. Because

✉ Pablo Rivas
Pablo_Rivas@Baylor.edu

Korn Sooksatra
Korn_Sooksatra1@Baylor.edu

[1] Baylor University, One Bear Place #97141, Waco, Texas 76798-7141, USA

Published online: 01 June 2022

the sticker is just a part of the stop sign, such an attack also utilizes only a part of the input. Additionally, Su et al. [26] proposed an evolutionary mechanism that creates an adversarial example by perturbing only one attribute (e.g., one pixel in an image sample), and later, [14] claimed that an adversarial example was created by only perturbing some non-robust features of a given input.

These works mentioned above inspired us to study why some adversarial attacks fail to create adversarial examples under the incomplete input scenario while some are successful. Further, we invent and formulate the Sensitivity-Inspired Constrained Evaluation Method (SICEM) that can estimate how much a specific part of the input can create an adversarial example compared to the complete one and other parts.

In our experiment, we choose to use images as our inputs since it is easy to visualize our results and allow an adversary to perturb the whole images, their top halves, and their bottom halves. After that we constructed two experiments: individual-image basis and individual-class basis. The individual-image basis experiment used SICEM to evaluate each image and compared its result with the baseline (i.e., the amount of added perturbation). On the other hand, the individual-class basis experiment used SICEM to each image belonging to each class and computed the average result of each class. Then, we compared the average results to the baseline (i.e., success rate and the amount of added perturbation). As a result, SICEM achieved an agreement of 65.9 and 72.35% on MNIST [19] and CIFAR-10 [17] datasets, respectively, with the baselines on the individual-image basis. Also, it achieved the agreement of 67.5 and 87.5% on MNIST and CIFAR-10 datasets, respectively, with the baselines on an individual-class basis. We can summarize our contributions as follows:

- We briefly describe famous state-of-the-art adversarial attacks and how we modify them to address incomplete inputs.
- To the best of our knowledge, SICEM is the first work that adversarially evaluates a classifier with incomplete inputs, and we also show how to mathematically design it.
- We create and train classifiers for MNIST and CIFAR-10 dataset on which we extensively construct experiments of SICEM on individual-image and individual-class basis. We show that the results from SICEM are aligned with the one of the baselines defined in Sect. 6.
- We discuss our results obtained from experiments in terms of effects of incomplete inputs on adversarial attacks and agreement of SICEM. Implicitly, the results have impacts on both an adversary and a learning model developer.

The paper is organized as follows: Sect. 2 briefly describes some relevant works; Sect. 3 provides the preliminaries for the remainder of the paper; Sect. 4 describes and formulates all the chosen adversarial attacks under the incomplete input scenario in this paper; Sect. 5 explains and formulates SICEM in detail; Sect. 6 shows our extensive experiments and the results; Sect. 7 discusses the results from Sect. 6 and our future works, and everything is concluded in Sect. 8.

2 Related work

Since SICEM is based on saliency maps, there are still other works related adversarial examples that also depend on saliency maps. In 2019, Etmann et al. [9] studied the connection betweem adversarial robustness and saliency map interpretability and found that a robust model showed more interpretable saliency map than a non-robust one. In 2020, Mangla et al. [22] used saliency maps to improve adversarial training for a robust classifier. In 2021, Wang et al. [33] applied multilayer saliency features to propose adversarial example detection, and then Sun et al. [27] generated adversarial examples for facial expression classifiers by using a saliency map.

Moreover, some existing works focuses on evaluating a learning model in term of adversarial robustness. In 2019, Carlini et al. [3] have created a guideline to perform an adversarial-example defense evaluation and also provided a checklist for the most common errors that occurred in many defense evaluations. Later, Dong et al. [8] have reviewed some adversarial attacks and defenses and then evaluated those with two robustness curves as the fair-minded evaluation criteria. They also implemented an adversarial robustness platform named RealSafe for evaluation experiments.

Noticeably, all the previous works focused on evaluating a classifier with complete inputs. Without loss of generality, SICEM is the first method that evaluates a classifier with incomplete inputs for adversarial robustness.

3 Background

3.1 Notation

A deep learning model is function $F(\cdot)$ where $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, n is the size of its input, and m is the size of its output. Further, its output depends on its input and its parameters θ (i.e., weights). In this paper, we focus on a classification task; hence, $F(\cdot)_i$ is a probability of the input belonging to class i. Explicitly, the classifier is ended by a

Softmax activation function. In addition, $Z(\cdot)$, where $Z : \mathbb{R}^n \rightarrow \mathbb{R}^m$, is the classifier's output before the *Softmax* activation function, and also $Z(\cdot)_i$ is the classifier's output for class i. Further, the remainder of this work denotes x as an input, x^* as its adversarial example, y as the correct class of x without the confidence score and t as an adversary's targeted class. Note that the confidence score is obtained from the *Softmax* layer.

3.2 Dataset and classifier

In this work, we use MNIST [19] and CIFAR-10 [17] as our datasets for creating adversarial examples. It is worth noting that all attributes of all the images are normalized to range [0, 1] and although all the data are images, throughout this paper, we refer to input as a 1-D vector (i.e., $x \in \mathbb{R}^n$) for simplicity. MNIST dataset consists of 60000 training data and 10000 test data each of which is an image of a digit. Note that both the training dataset and test dataset are balanced. That is, the training dataset consists of 6000 images of each digit. Hence, we create a deep convolutional neural network and train it with the training data by using the cross-validation method [2]. The network consists of convolutional layers for extracting features, dense layers for classifying according to the extracted features and batch normalization [15] and dropout [25] which are powerful regularization techniques. Its architecture is illustrated in Fig. 12 where a dense layer and a convolutional layer are defined in the red rectangle. As a result, we test the classifier with the test data and achieve the accuracy of 99.39%.

Further, the CIFAR-10 dataset is composed of 50000 training data and 10000 test data each of which is an image of the classes as follows: airplane (class 0), automobile (class 1), bird (class 2), cat (class 3), deer (class 4), dog (class 5), frog (class 6), horse (class 7), ship (class 8) and truck (class 9). Note that both the training and test dataset are balanced. Thus, we design a larger deep convolutional network since a CIFAR-10 image is bigger than a MNIST image and train it with cross-validation method, and the network has the same components as our MNIST classifier does including max-pooling layers to minimize features' sizes. The network's architecture is summarized in Fig. 13. It is worth noting that we sometimes refer to the classes as their indexes started by 0. For example, we may refer to automobile class as an automobile or class 1 because its index is 1 for simplicity. Consequently, our classifier can achieve an accuracy of 85.59%.

The confusion matrices of both of our MNIST and CIFAR-10 classifiers on the test data are, respectively, shown in Figs. 1 and 2.

3.3 Adversarial example

An adversarial example is a sample added by carefully crafted small perturbation so that the sample is classified as a different class to the original sample. This was discovered by Szegedy et al. [28] in 2013, and then in 2014, Goodfellow et al. [11] further analyzed why adversarial examples could mislead classifiers. Also, they proposed a one-step attack called Fast Gradient Sign Method (FGSM) and adversarial training to defend against the attack. Later, FGSM was modified to be iterative to make it stronger in [18]. In 2016, Papernot et al. [23] developed an attack which altered only a few pixels in an image to create an adversarial example and called it Jacobian-based Saliency Map Attack (JSMA), and Kurakin et al. showed that adversarial examples could be effective in the real world even though a camera could reduce small perturbation. Next, in 2017, Carlini and Wagner [4] created a state-of-the-art adversarial attack by formulating an unconstraint optimization problem and using a gradient approach to solve it. More interestingly, in 2019, Su et al. [26] constructed an adversarial example that perturbed only one pixel by using an evolutionary algorithm.

4 Generating adversarial examples under incomplete input scenario

This section shows how each attack in popular metrics (i.e., L_0, L_∞ and L_2 attack) generates an adversarial example. Also, the incomplete input situation where an adversary has limited access to the input is considered. As described in [10], we assume that the adversary is given mask M which indicates which attributes in x the adversary can access and perturb. Specifically, $M \in \{0, 1\}^n$, and M_i denotes attribute i of M. M_i is 1 if the adversary is capable of perturbing x_i and 0 otherwise. In other words, if the adversary can perturb the whole input, M_i where $i \in \{0, 1, ..., n-1\}$ is 1. Further, we denote ϵ to indicate the perturbation bound. Then, we used the least required ϵ for generating adversarial examples to indicate the effectiveness of the attacks and encourage the readers to read the original papers of the following attacks for more details.

4.1 L_0 attack

The metric of this kind of attack is based on the number of perturbed attributes in x. In other words, $\epsilon = ||x^* - x||_0 = \sum_{i=0}^{n-1} [x_i^* - x_i \neq 0]$ where ϵ denotes the least required perturbation in term of its metric (i.e., L_0 in this case) for creating an adversarial example, and $[\omega] = 1$ if ω is true

and 0 otherwise. We choose the attack in [23] for this metric because it is effective in that work.

4.1.1 Jacobian-based saliency map attack (JSMA)

This attack [23] is based on Jacobian matrix and finds which attributes of x to which $Z(x)_t$ is sensitive are by using a saliency map where t is the targeted class which we would like to achieve. Specifically, first, the Jacobian matrix of $Z(x)$ is calculated, and then for each class j, $\frac{\partial Z(x)_j}{\partial x}$ is obtained. The saliency map can be computed by considering each attribute of x. For attribute i, if $\frac{\partial Z(x)_t}{\partial x_i} > 0$, and also $\sum_{j \neq t} \frac{\partial Z(x)_j}{\partial x_i} < 0$, the value in the map with respect to x_i is $\frac{\partial Z(x)_t}{\partial x_i} \cdot |\sum_{j \neq t} \frac{\partial Z(x)_j}{\partial x_i}|$, and 0 otherwise. After obtaining the map, the attribute which has the highest value in the map is increased, and then the map is created again. This method is iterated until the targeted classifier predicts x as class t. This is a targeted attack; however, we can adjust it to an untargeted attack.

To achieve the untargeted attack, we only need to change the condition for creating the saliency map as follows: for attribute i and given y as the correct class of x, if $\frac{\partial Z(x)_y}{\partial x_i} < 0$ and $\sum_{j \neq y} \frac{\partial Z(x)_j}{\partial x_i} > 0$, the value in the map with respect to x_i is $|\frac{\partial Z(x)_y}{\partial x_i}| \cdot \sum_{j \neq y} \frac{\partial Z(x)_j}{\partial x_i}$, and 0 otherwise. The rest of the procedure are the same as the targeted attack.

Further, under the incomplete input scenario, the step finding the highest-value in the saliency map only focuses on x_i where $M_i = 1$ because the adversary cannot alter attribute j where $M_j = 0$.

4.2 L_∞ attack

The metric used in this attack is the highest value of the perturbation vector. Specifically, it can be computed by $\epsilon = ||x^* - x||_\infty = \max(|x^* - x|)$. We use the state-of-art attacks described in [11] and [18] since one of them is effective and very fast, and the other one is slower; however, it is much more difficult for humans to notice the perturbations added by the latter.

4.2.1 Fast gradient sign method (FGSM)

Goodfellow et al. [11] showed how to create an adversarial example in one step with respect to L_∞ norm. First, they feed an input to the target classifier and then compute the gradient from the obtained loss. Then, they take only the sign of the gradient to know the direction of perturbation so that they can increase the loss. At last, they multiply the gradient's sign with ϵ which is the maximum perturbation according to L_∞ norm. Their procedure can be formulated as

$$x^* = \text{Clip}(x + \epsilon \cdot \text{sign}(\nabla_x J_y(x)), 1, 0), \tag{1}$$

where x is the input, y is the correct class of x, $J_y(x)$ is the loss function according to the output of x and class y, $\text{sign}(z)$ is a function that outputs only the sign of z, x^* is the adversarial example and $\text{Clip}(x, u, l)$ is a function that clips the attributes of x that are higher than u to u and are lower than l to l. This attack is untargeted.

Under the incomplete input situation, given mask M, (1) can be formulated as

$$x^* = \text{Clip}(x + \epsilon \cdot \text{sign}(M \odot \nabla_x J_y(x)), 1, 0),$$

where $a \odot b$ is an element-wise multiplication between a and b.

4.2.2 Iterative gradient sign method (IGSM)

In [18], this attack is named as basic iterative method. It simply modifies from FGSM by iteratively adding perturbations with a small step size. If a perturbation goes beyond ϵ, it will be clipped back to ϵ. Step i is to add a perturbation as

$$x^*(\tau + 1) = \text{Clip}(x^*(\tau) + \alpha \cdot \text{sign}(\nabla_{x^*(\tau)} J_y(x^*(\tau))), 1, 0), \tag{2}$$

where $x^*(\tau)$ is an adversarial example in step τ, $x^*(0) = x$, x is the clean input and y is the correct class of x.

Under the incomplete input scenario, given mask M, (2) can be reformulated as

$$x^*(\tau + 1) = \text{Clip}(x^*(\tau) + \alpha \cdot \text{sign}(M \odot \nabla_{x^*(\tau)} J_y(x^*(\tau))), 1, 0).$$

4.3 L_2 attack

The metric used in this attack is the Euclidean distance. Specifically, it can be computed by

$$||x^* - x||_2 = \sqrt{\sum_{i=0}^{n-1} (x_i^* - x_i)^2}.$$

Further, we choose a state-of-the-art attack in this metric for our experiment.

4.3.1 Carlini and Wagner attack (CWA)

This attack [4] is named after its inventors and showed that it is a state-of-the-art attack for generating adversarial examples. It also can be adjusted to become L_0 and L_∞ attack; nonetheless, they are based on L_2 attack. Specifically, this attack is modified from the attack in [28] by

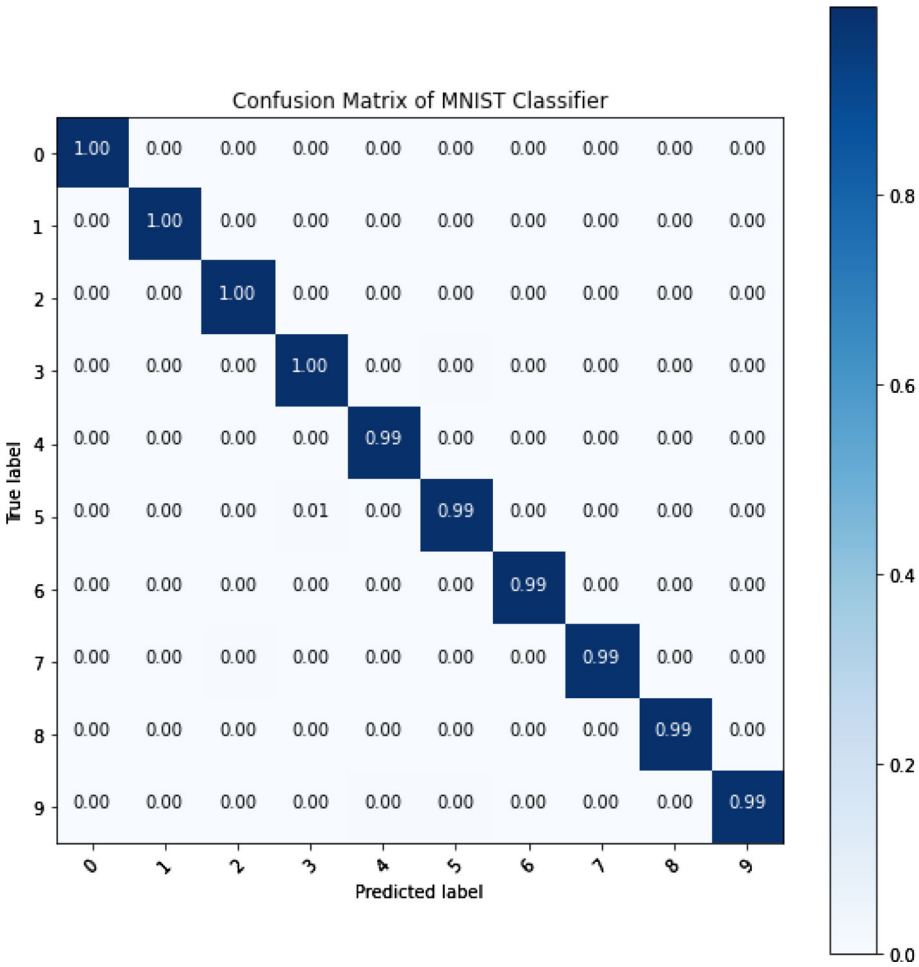


Fig. 1 Confusion matrix of our trained MNIST classifiers

omitting all the constraint by converting x^* to a function based on the hyperbolic tangent function of variable w so that the adversary can directly apply a gradient-based technique on w without clipping any value. Thus, the optimization problem of this attack is

$$\min \left\| \frac{1}{2}(\tanh(w)+1) - x \right\|_2 + \lambda \cdot L\left(\frac{1}{2}(\tanh(w)+1)\right), \quad (3)$$

where $w \in \mathbb{R}^n$, λ is to balance the distance between the minimization of the perturbation and the one of function $L(x^*)$ and $L(x^*)$ is a loss function; therefore, we use $L(x^*) = \max(\max_{j \neq t} Z(x^*)_j - Z(x^*)_t, -\kappa)$ recommended by [4] where t is the targeted class and κ measures how much confidence the adversary desires. Clearly, this is a targeted attack. Nevertheless, we can reformulate (3) to an untargeted attack by adjusting loss function $L(x^*)$ to

$$L(x^*) = \max(Z(x^*)_y - \max_{j \neq y} Z(x^*)_j, -\kappa),$$

where y is the correct class of x.

Under incomplete input scenarios, we change problem (3) to

$$\min \ ||\delta||_2 + \lambda \cdot L(x+\delta),$$

where $\delta = M \odot (\frac{1}{2}(\tanh(w)+1) - x)$.

5 Sensitivity-inspired constrained evaluation method (SICEM)

We propose a method (SICEM) to evaluate the incomplete input against an adversarial attack compared to the complete one by utilizing the Jacobian matrix idea in [23] discussed in Sect. 4.1.1. Given input x, we feed forward x to the target classifier $F(x)$ and calculate the Jacobian matrix of $Z(x)$ with respect to x by $\frac{\partial Z(x)_y}{\partial x_i}$ for $i \in \{0, 1, ..., n-1\}$ where x_i is attribute i of x and y is the correct class of x. In other words, $\frac{\partial Z(x)_y}{\partial x_i}$ implies the sensitivity of $Z(x)_y$ with respect to x_i and therefore the sensitivity of $Z(x)_y$ to x_i can be approximately computed by

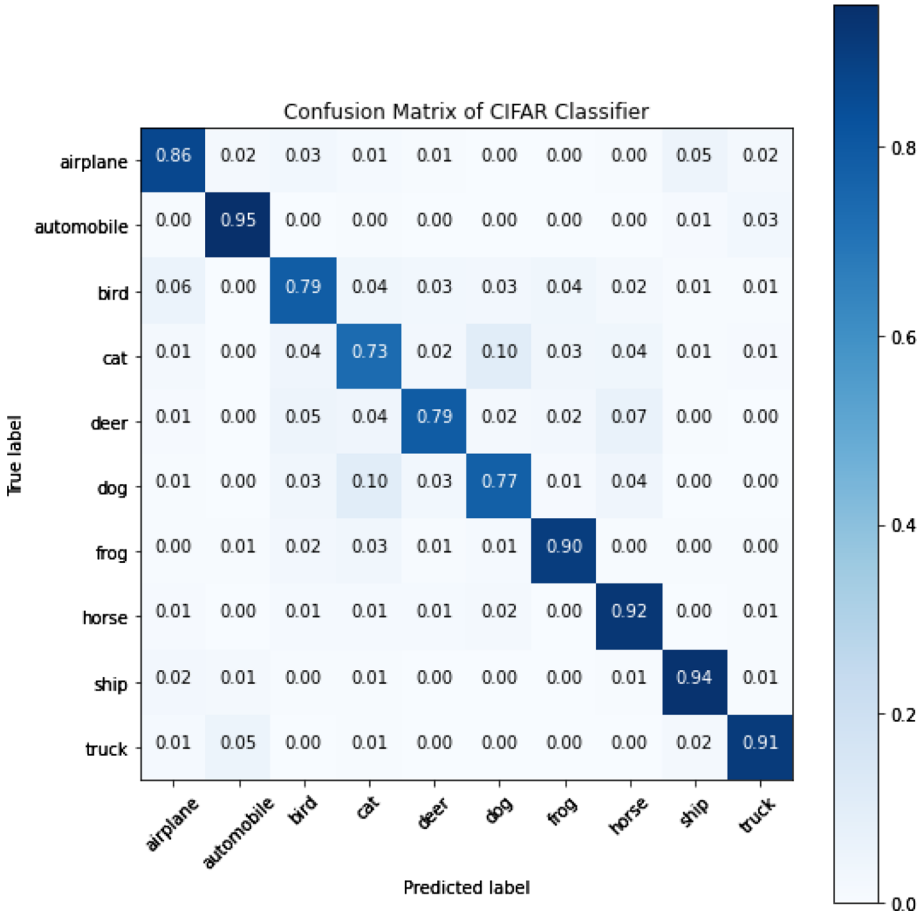


Fig. 2 Confusion matrix of our trained CIFAR-10 classifiers

$$s(x,y)_i = \left| \min\left(0, \frac{\partial Z(x)_y}{\partial x_i} \cdot \left(\sum_{y' \neq y} \frac{\partial Z(x)_{y'}}{\partial x_i}\right) \cdot C(y,1,0)_i\right)\right|, \tag{4}$$

where

$$C(y,u,l)_i = \begin{cases} u - x_i, & \text{if } \dfrac{\partial Z(x)_y}{\partial x_i} < 0 \\ x_i - l, & \text{otherwise} \end{cases},$$

which indicates how much x_i can be altered to the upper (lower) bound $u(l)$ if $\frac{\partial Z(x)_y}{\partial x_i} < 0$ ($\frac{\partial Z(x)_y}{\partial x_i} \geq 0$) to reduce $Z(x)_y$.

Specifically, to achieve the goal of adversarial attacks, it is intuitively suitable to change x_i so that $Z(x)_y$ and $Z(x)_i$ where $i \neq y$ can change in the opposite directions. Precisely, if they change in the same directions with respect to x_i, $\frac{\partial Z(x)_y}{\partial x_i} \cdot \left(\sum_{y' \neq y} \frac{\partial Z(x)_{y'}}{\partial x_i}\right)$ is positive, and therefore, $s(x,y)_i$ is zero as in (4). On the other hand, the value of $s(x,y)_i$ in the right-hand term in the minimization in (4) given as

$$\left(\left|\frac{\partial Z(x)_y}{\partial x_i} \cdot \left(\sum_{y' \neq y} \frac{\partial Z(x)_{y'}}{\partial x_i}\right) \cdot C(y,1,0)_i\right|\right)$$

consists of three parts. The first part is to quantify how much $Z(x)_y$ is sensitive to x_i as discussed earlier. The second part is to measure how much the other classes are sensitive to x_i, and this is guaranteed that the second part changes in the opposite direction to the first part due to the minimum function. The last part is also necessary because if the first two parts have high values; however, x_i cannot be altered in the desired direction due to the limitation at the upper bound (i.e., 1) or the lower bound (i.e., 0), x_i should not be significant, or in other words, $s(x,y)_i$ should not indicate the high sensitivity. Hence, $s(x,y)_i$ must include $C(y, 1, 0)$ for more accurate sensitivity. In addition, $s(x,y) = \{s(x,y)_i; 0 \leq i \leq n-1\}$ is the vector of the sensitivity of $Z(x)_y$ with respect to all the attributes of x.

Next, the sensitivity of $Z(x)_y$ for each x_i is used to compute the estimation of the sensitivity score of the given input x and its mask M as described in Sect. 4 as

$$S(x,M)_y = \sum_{i=0}^{n-1} (s(x,y)_i \cdot M_i). \quad (5)$$

Explicitly, in the case of the complete input, $M_i = 1$ for $i \in \{0, 1, ..., n-1\}$. Let M_c denote the mask for the complete input and hence, the sensitivity ratio of class y for an arbitrary mask (incomplete input) is

$$r(x,M)_y = \frac{S(x,M)_y}{S(x,M_c)_y},$$

which implies how much $F(x)_y$ is sensitive to incomplete x with respect to mask M compared to the complete x.

6 Experiments

In this section, we experiment SICEM discussed in Sect. 5 on individual-image and individual-class bases from MNIST and CIFAR-10 datasets with the chosen attacks described in Sect. 4 under the incomplete input scenario. The settings and success criterion of the aforementioned attacks are explained as follows:

- JSMA: Its step size is set to 0.5 to increase an attribute, and if the adversary creates an adversarial example with $\epsilon > 0.05 \cdot n$ (which implies that the adversary alters more than 5% of attributes of x), the adversary fails to create an adversarial example for x since the average ϵ in [23] is around 4%.
- FGSM and IGSM: ϵ is initialized to 0.02 and is iteratively increased by 0.02 until it successfully creates an adversarial example. However, if ϵ exceeds 0.16, the adversary fails to create an adversarial example for the given input. We choose the number 0.16 because if ϵ is greater than this, the perturbation may be too obvious. Moreover, for IGSM, the step size is 0.02, and the maximum number of iterations is $\lfloor \frac{\epsilon}{0.02} \rfloor$.
- CWA: λ is initialized to 0.01, and the maximum of the inner iteration is 50. λ is iteratively multiplied by two until the attack successfully alters the prediction of a classifier to a wrong class for x or λ exceeds 100. Furthermore, if λ exceeds 100, it is considered as a failure to create an adversarial example for the given input x.

It is worth noting that all the attacks in this experiment are untargeted because we simply would like to compare the attacks' performances in different uncovered parts of images to perturb. Also, we obtained all the hyperparameters of the attacks shown above from our empirical experiments that showed that attacks were effective and did not consume much time. Further, to verify the correctness of SICEM, we have different baselines for individual image and individual class bases.

6.1 Individual image

We randomly pick an image from the test dataset of MNIST and another one from the test dataset of CIFAR-10 which are, respectively, shown in Figs. 3 and 4. Clearly, the image of number 4 classified by our MNIST classifier with the confidence of 100% and the image of automobile classified by our CIFAR-10 classifier with the confidence of 94.75% are picked from MNIST and CIFAR-10 datasets, respectively. In this experiment, we assume that there are three scenarios for the adversary. The first one is that the adversary can access and perturb the whole input (the complete input). The second one is that the adversary can perturb only the top half of the input, and the last one is that the adversary can perturb only the bottom half of the input. Explicitly, the adversary is given incomplete inputs with totally different parts. In specific, mask M for the complete input scenario is M_c as discussed in Sect. 5, and we create mask M_t for the top half of an image by setting 1 to the top half of the mask and setting the other attributes to 0. On contrary, for the bottom half scenario, we set 1 to the bottom half of its mask and set 0 to the other attributes and denote it by M_b.

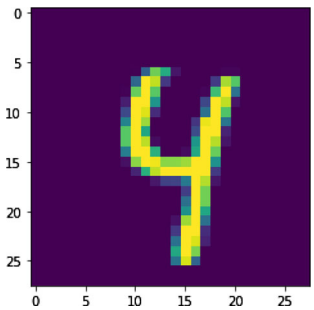


Fig. 3 Clean image of number four with 100% confidence classified by our MNIST classifer

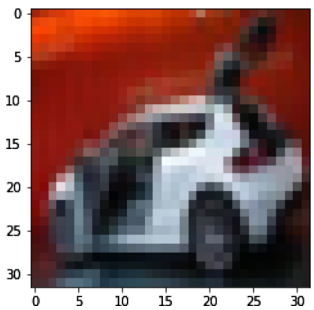
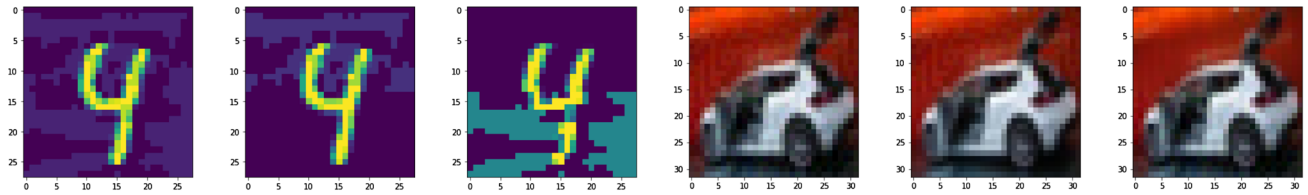


Fig. 4 Clean image of automobile with 94.75% confidence classified by our CIFAR-10 classifier



(a) $\epsilon = 0.10$ and predicted class is 9 with 84.16% confidence

(b) $\epsilon = 0.14$ and predicted class is 9 with 71.89% confidence

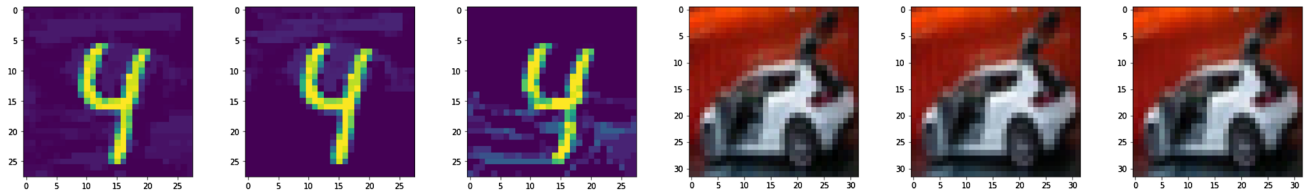
(c) $\epsilon = 0.46$ and predicted class is 8 with 61.75% confidence

(d) $\epsilon = 0.02$ and predicted class is truck with 100% confidence

(e) $\epsilon = 0.02$ and predicted class is truck with 97.5% confidence

(f) $\epsilon = 0.02$ and predicted class is truck with 100% confidence

Fig. 5 FGSM applied on **a**, **d** complete images, **b**, **e** top halves of images and **c**, **f** bottom halves of images



(a) $\epsilon = 0.08$ and predicted class is 9 with 53.12% confidence

(b) $\epsilon = 0.10$ and predicted class is 9 with 61.51% confidence

(c) $\epsilon = 0.32$ and predicted class is 3 with 59.07% confidence

(d) $\epsilon = 0.02$ and predicted class is truck with 100% confidence

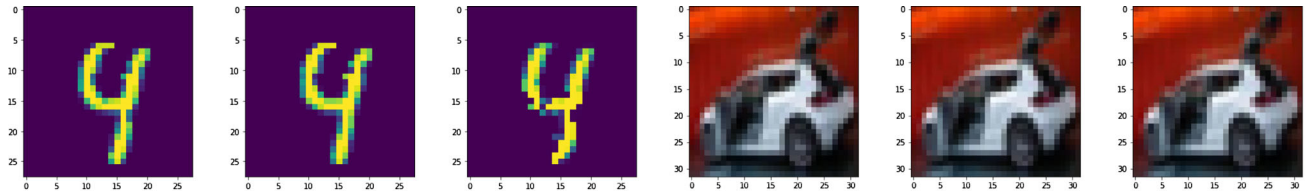
(e) $\epsilon = 0.02$ and predicted class is truck with 93.2% confidence

(f) $\epsilon = 0.02$ and predicted class is truck with 99.9% confidence

Fig. 6 IGSM applied on **a**, **d** complete images, **b**, **e** top halves of images and **c**, **f** bottom halves of images

Figures 5, 6, 7 and 8, respectively, show FGSM, IGSM, CWA and JSMA performed on the image of number 4 and automobile. When the adversary is allowed to perturb the whole image and the top half of the image of number four, all of them are misclassified as number nine because this image easily turns to number nine by connecting the heads of number four with small perturbations. On the other hand, when the adversary is allowed to perturb the bottom half of the image, the image is misclassified as several numbers (i.e., number three, number five and number eight) since there is no similar number that is very close to four by perturbing its bottom half. Furthermore, the perturbations in Figs. 5c and 8c are very obvious and certainly perceptible by humans especially the adversarial example in Fig. 8c which is considered as a failure because ϵ exceeds 5% of the image and clearly looks like number three. Explicitly, the adversary allowed to perturb the top halves requires higher ϵ than the one allowed to perturb the complete images to successfully create those adversarial examples in Figs. 5b, 6b, 7b and 8b. On the other hand, the adversary allowed to perturb the bottom halves also needs higher ϵ than the one allowed to perturb the top halves to achieve those adversarial example in Figs.5c, 6c, 7c and 8c. This implies that for this image of number four, perturbing the complete image is easier than perturbing only the top half of the image for creating an adversarial example. However, perturbing only the bottom half of the image is the most difficult among these complete and incomplete input scenarios. Moreover, this claim is empirically confirmed in Fig. 9. That is, the least required ϵ of the bottom half is higher than the ones of whole image and top half under both FGSM and IGSM. Another thing that can be implied from Fig. 9 is FGSM requires higher ϵ to mislead our MNIST classifier than IGSM where $FGSM$ and $IGSM$ denote the confidence score of the correct class (i.e., number four) by performing FGSM and IGSM, respectively, and $\overline{FGSM}$ and $\overline{IGSM}$ denote the highest confidence score among the other classes by performing FGSM and IGSM, respectively. Therefore, FGSM is weaker than IGSM. Further, perturbing only the top half is much easier than perturbing only the bottom half because it requires lower ϵ to mislead the classifier than the bottom half.

To formally explain why allowing to perturb only the bottom half of number four is more difficult than perturbing only the top half to create an adversarial example, we apply SICEM explained in Sect. 5 and create the map of $s(x, 4)$ shown in Fig. 11a where x is the image of number four. Noticeably, the top half of the map is darker



(a) $\epsilon = 1.68$ and predicted class is 9 with 59.23% confidence

(b) $\epsilon = 1.90$ and predicted class is 9 with 53.45% confidence

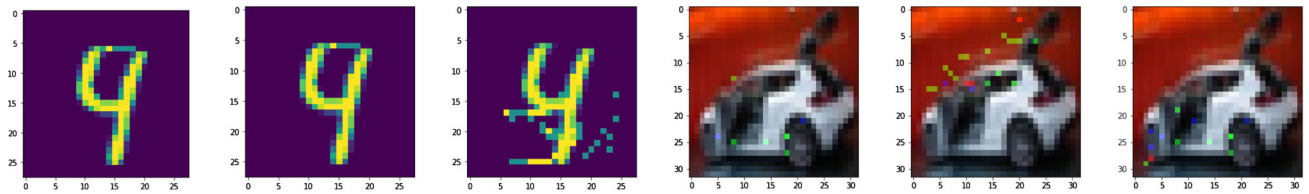
(c) $\epsilon = 2.94$ and predicted class is 5 with 44.45% confidence

(d) $\epsilon = 0.039$ and predicted class is truck with 50.5% confidence

(e) $\epsilon = 0.088$ and predicted class is truck with 50% confidence

(f) $\epsilon = 0.045$ and predicted class is truck with 53.7% confidence

Fig. 7 CWA applied on **a**, **d** complete images, **b**, **e** top halves of images and **c**, **f** bottom halves of images



(a) $\epsilon = 5$ and predicted class is 9 with 58.33% confidence

(b) $\epsilon = 5$ and predicted class is 9 with 58.33% confidence

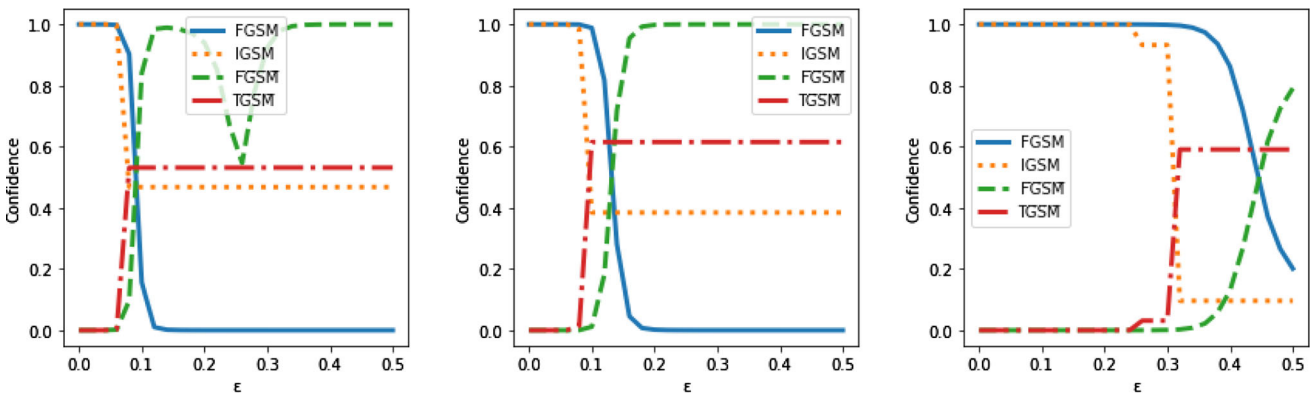
(c) $\epsilon = 41$ and predicted class is 3 with 47.37% confidence

(d) $\epsilon = 7$ and predicted class is truck with 52% confidence

(e) $\epsilon = 21$ and predicted class is truck with 60% confidence

(f) $\epsilon = 12$ and predicted class is truck with 55% confidence

Fig. 8 JSMA applied on **a**, **d** complete images, **b**, **e** top halves of images and **c**, **f** bottom halves of images



(a) Complete image **(b)** Top half **(c)** Bottom half

Fig. 9 These graphs summarize the confidences of number four and the highest-confidence class excluding number four over ϵ

than the bottom half, and $r(x, M_t)$ is higher than $r(x, M_b)$. This implies that $Z(x)_4$ is easier to be reduced by perturbing the attributes in the top half than perturbing the ones in the bottom half. According to [14], intuitively, the attributes in the top half of the image create more non-robust useful features than the ones in the bottom half. This analysis with SICEM guarantees that our claim discussed earlier is correct.

Next, we focus on the image of an automobile from CIFAR-10. As seen in Figs. 5, 6, 7, the adversarial

examples generated by FGSM and IGSM require the exact same ϵ which is 0.02, and also ϵ that CWA requires is very low. Explicitly, they need much lower ϵ than the image of number four. This is because of the layer-wise linearity of the classifier as discussed in [11]. In other words, the larger the input is, the less ϵ is required to create an adversarial example. Further, the image of an automobile whose size is 3072 is much larger than the image of number four whose size is 784. Nevertheless, JSMA needs higher ϵ for creating an adversarial example for the image of an automobile than the image of number four due to the metric of L_0 norm.

It is not clear to compare the hardness of creating an adversarial example in different scenarios by performing FGSM and IGSM since they require the same ϵ. However, the confidence score in Figs. 5e and 6e are lower than the ones under other scenarios. This may imply that perturbing only the top half is the most difficult scenario to create an adversarial example. Additionally, the results from CWA and JSMA in Figs. 7 and 8 encourage that allowing to perturb only the attributes in the top half is the most difficult for creating an adversarial example of this image due to their required ϵ. Moreover, Fig. 10 shows an interesting phenomenon. Such a phenomenon is that by performing FGSM, the adversary can successfully create an adversarial example with $\epsilon = 0.02$; nonetheless, when the adversary increases ϵ, the confidence of automobile surprisingly increases, and it is classified as automobile again at $\epsilon = 0.14$. The explanation of this is that the gradients' signs are computed obtained from the current input at that time. Because a deep learning model is a very complicated function, at any point of the input, the gradients' signs can be different. Thus, increasing ϵ does not guarantee successfully finding adversarial examples. This is also a reason why IGSM is stronger than FGSM due to its refinement step. Note that automobile is class 1 in the CIFAR-10 dataset.

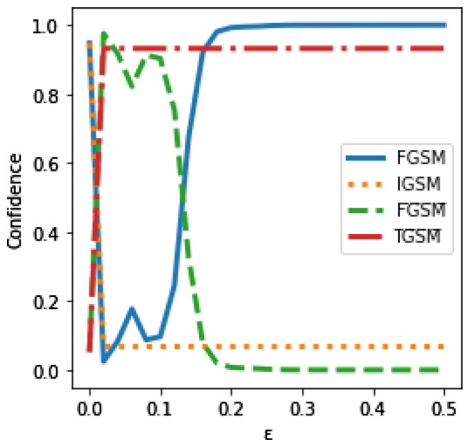


Fig. 10 This graph shows the confidences of automobile and the highest-confidence class excluding automobile over ϵ where only top half is allowed to perturb

To formally conclude this, the sensitivity map of the image is created in Fig. 11b. Note that for each pixel, we perform the summation of the sensitivity across all the channels (i.e., RGB). It can be noticed that the bottom half is much darker than the top half, and yet, $r(x, M_t)_1 < r(x, M_b)_1$. This implies that perturbing the top half is more difficult than perturbing the bottom half of the image to find an adversarial example.

After that we further experiment on it by randomly picking 200 images from the test data of MNIST and 200 images from the test data of CIFAR-10. Then, we evaluate the images one by one whether SICEM correctly tells if their top halves are easier than their bottom halves to find adversarial examples. First, we have to discuss three definitions below which define the baseline of this basis and other important terms.

Definition 1 (Baseline for individual-image basis) Given image x and attack a, x's top half is more difficult to attack by a when at least one of the following is true:

- The attack successfully finds an adversarial example on the bottom half and cannot find one on the top half.
- The attack requires less ϵ on the bottom half than the top half to find an adversarial example.

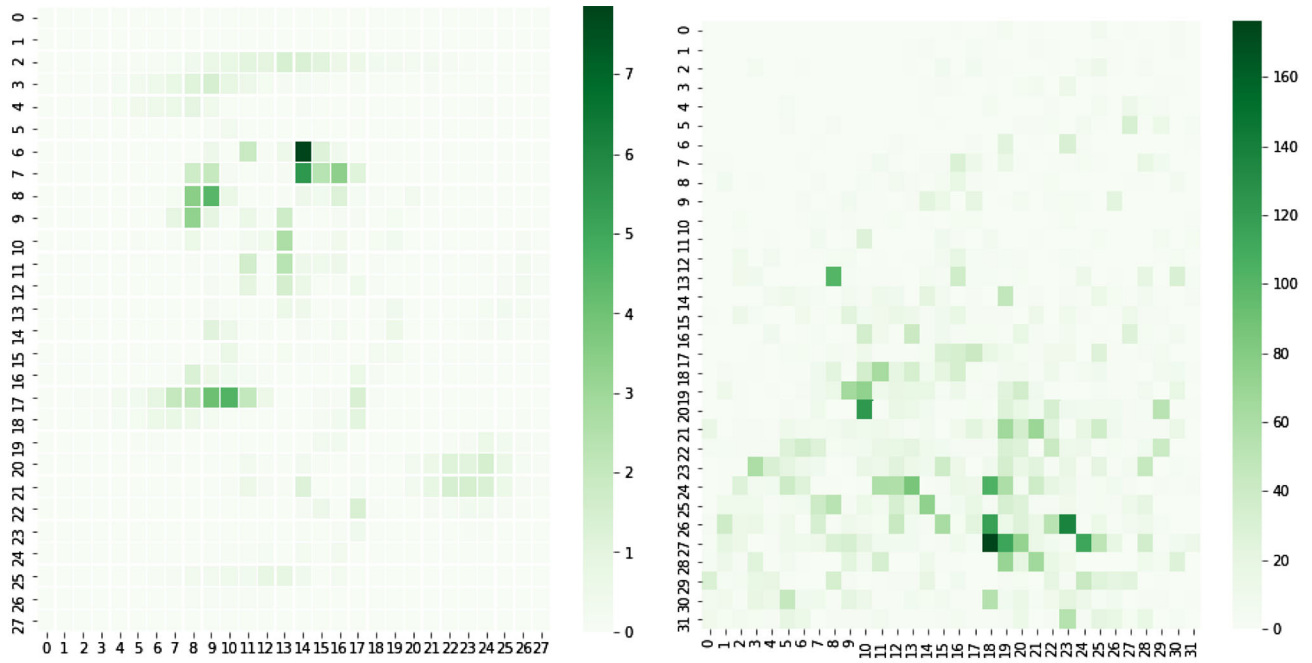
Similarly, x's bottom half is more difficult to attack by a when at least one of the following is true:

- The attack successfully finds an adversarial example on the top half and cannot find one on the bottom half.
- The attack requires less ϵ on the top half than the bottom half to find an adversarial example.

Definition 2 (Decision by using SICEM for individual-image basis) Given image x, x's top half is more difficult to find an adversarial example than its bottom half when the sensitivity score of its top half is lower than its bottom half. Similarly, x's bottom half is more difficult to find an adversarial example than its top half when the sensitivity score of its bottom half is lower than its top half.

Definition 3 (Agreement for individual-image basis) Given image x and attack a, SICEM has an agreement on image x with a baseline when both of them have the same result about image x. For example, SICEM results that x's top half is more difficult to attack than the bottom half, and the baseline also results in the same; then, SICEM has an agreement with the baseline.

Definition 4 ($\beta\%$ agreement for individual-image basis) Given a set of images and a set of attacks, SICEM has $\beta\%$ agreement with a baseline when the average percentage of images in the set, on which SICEM has agreements with



(a) Number four: $r(x, M_t)_4 = 0.61$ and $r(x, M_b)_4 = 0.39$

(b) Automobile: $r(x, M_t)_1 = 0.18$ and $r(x, M_b)_1 = 0.82$

Fig. 11 Sensitivity maps: **a** the top half appears to have more sensitive regions; **b** shows sensitivity near the bottom half

the baseline is β where $\beta \in [0, 100]$. For instance, given set of images $\mathcal{X}$ and set of attack $\mathcal{A} = \{a_1, a_2, a_3\}$, with attack a_1, a_2 and a_3, there are, respectively, 70%, 65% and 84% of images in $\mathcal{X}$ on which SICEM has agreements with the baseline. Thus, SICEM has $(70 + 65 + 84)/3 = 73\%$ agreement with the baseline.

According to Definition 4, the agreement of SICEM with the baseline for MNIST dataset by performing the four attacks is 65.9% and for the CIFAR-10 dataset is 72.35.%.

6.2 Individual class

This experiment uses the 400 images picked in Sect. 6.1 to find their sensitivity and perform the attacks to determine the success rates and average required ϵ in each class. Then, we will evaluate how much SICEM is accurate with respect to each class.

Table 1 shows the average sensitivity score, computed from (5), for each class of the datasets where M_t is the mask for the top half and M_b is the mask for the bottom half. Explicitly, the scores of images of the CIFAR-10 dataset are much higher than the ones of images of the MNIST dataset. This is the main reason that a CIFAR-10 image is much easier to find an adversarial example than a MNIST image. Further, it can be noticed from the table that images in MNIST of which top halves are easier to find their adversarial example are in class 3, 5 and 6. Similarly, perturbing the top halves of CIFAR-10 images in classes 2, 3, 4, 5, 6 and 7 is easier to find adversarial examples than perturbing the bottom halves. Next, we will check if this conclusion is correct.

Table 1 Average sensitivity score of each class with mask M_t and M_b of MNIST and CIFAR-10 datasets

Mask	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
MNIST dataset										
M_t	16.33	30.04	27.57	**45.82**	48.66	**53.62**	**75.65**	20.4	38.66	36.9
M_b	30.07	56.79	67.08	**43.06**	54.32	**40.46**	**38.51**	49.53	57	41.3
CIFAR-10 dataset										
M_t	1556.4	3800.1	**3930**	**2952.4**	**5583.1**	**8088.3**	**6387.6**	**2638**	3927.4	3457.6
M_b	1975.7	11299	**3057.3**	**2213.6**	**5418.4**	**5138.8**	**4510.3**	**2525.5**	9046.2	8918.5

We perform the attacks on the images under complete and incomplete scenarios and list all success rates and average required ϵ of finding adversarial examples in each class for comparison. Tables 3 and 4 show the success rates and average required ϵ, respectively, to find adversarial examples after performing the adversarial attacks on the test images of MNIST dataset where t-*attack* denotes that the adversary performs *attack* with mask M_t and b-*attack* denotes that the adversary performs *attack* with mask M_b. Similarly, Tables 5 and 6 show the success rates and average required ϵ, respectively, after performing the attacks on the test images of CIFAR-10 dataset. Note that we consider only the images that are correctly classified by our classifiers. We define the baseline for this basis and some important terms below.

Definition 5 (Baseline for individual-class basis) Given class c and attack a, c's top half is more difficult to attack by a when at least one of the following is true:

- The success rate of the top half is less than the one of the bottom half.
- If the success rates of the top and bottom halves are equal, the average required ϵ to find adversarial examples of the top half is greater than the one of the bottom half.

Similarly, c's bottom half is more difficult to attack by a when at least one of the following is true:

- The success rate of the bottom half is less than the one of the top half.
- If the success rates of the top and bottom halves are equal, the average required ϵ to find adversarial examples of the bottom half is greater than the one of the top half.

Definition 6 (Decision by using SICEM for individual-class basis) Given class c, the top halves of the images belonging to class c are more difficult to find an adversarial example than their bottom halves when the average sensitivity score of the top halves is lower than their bottom halves. Similarly, the bottom halves of the images belonging to class c are more difficult to find an adversarial example than their top halves when the average sensitivity score of the bottom halves is lower than their top half.

Definition 7 (Agreement for individual-class basis) Given class c and attack a, SICEM has an agreement on class c with a baseline when both of them have the same result about class c. For example, SICEM results that c's top half is more difficult to attack than the bottom half, and the baseline also results in the same; then, SICEM has an agreement with the baseline.

Definition 8 ($\beta\%$ agreement for individual-class basis) Given a set of classes and attack a, SICEM has $\beta\%$ agreement with a baseline when the percentage of classes in the set, on which SICEM has agreements with the baseline is β where $\beta \in [0, 100]$. For example, given set of classes $\mathcal{C} = \{c_1, c_2, c_3\}$ and attack a, SICEM has agreements with the baseline on class c_1 and c_2. Then, SICEM has $\frac{2}{3} \cdot 100 = 66.67\%$ agreement with the baseline.

According to Definitions 5 and 8, for each attack, the classes of MNIST images, top halves of which are easier to create adversarial examples than their the bottom halves, are: class 1, 3, 4, 5, 6 and 9 for FGSM and IGSM (70% agreement for each); class 1, 3, 4, 5, and 9 for CWA (60% agreement); class 1, 2, 3, 4, 5 and 6 for JSMA (70% agreement).

Hence, for the MNIST dataset, in terms of average sensitivity scores, success rates and required ϵ, the mean agreement of SICEM is $(70 + 70 + 60 + 70)/4 = 67.5\%$. Next, we will consider the CIFAR-10 dataset.

As shown in Table 1, the images in CIFAR-10, the top halves of which are easier to attack than the bottom are in classes 2, 3, 4, 5, 6 and 7. After this, we will evaluate how much correct SICEM is for the CIFAR-10 dataset.

To evaluate which part of an image is easier to attack, we use the same criterion in MNIST earlier. Note that we do not count class 0 and 7 in FGSM and class 7 in IGSM since their top halves are as vulnerable as their bottom halves. Essentially, for each attack, the classes of CIFAR-10 images, top halves of which are easier to create adversarial examples than their the bottom halves, are: class 0, 2, 3, 5, 6 and 8 for FGSM (60% agreement); class 2, 3, 4, 5, 6 and 7 for IGSM (100% agreement); class 2, 3, 4, 5, 6 and 7 for CWA (100% agreement); class 3, 4, 5, 6 and 7 for JSMA (90% agreement).

In summary, the mean agreement of SICEM on the CIFAR-10 dataset is $(60 + 100 + 100 + 90)/4 = 87.5\%$ which is significantly higher than the agreement on the MNIST dataset. The reason behind this will be discussed in Sect. 7.

At last, derived from all the results in Sects. 6.1 and 6.2, all the agreement are summarized in Table 2.

7 Discussion and future works

7.1 Effect of incomplete input

According to the results, limiting an adversary by allowing him/her to perturb only some attributes of an input significantly affects the performance of the attacks, especially FGSM, because their success rates are lower than or equal to the ones with the complete input. However, the results of

Table 2 Agreement of SICEM in each attack on each dataset

Dataset	Agreement				
	JSMA (%)	FGSM (%)	IGSM (%)	CWA (%)	Mean (%)
Individual image					
MNIST	64.5	67.8	67.8	63.5	**65.9**
CIFAR-10	72.5	71.2	71.2	74.5	**72.35**
Individual class					
MNIST	70	70	70	60	**67.5**
CIFAR-10	90	60	100	100	**87.5**

JSMA are surprising because complete images in some classes are more difficult to attack than their incomplete ones to find adversarial examples. The reason behind this is that JSMA does not consider function $C(y, u, l)$ as we do in Sect. 5. Thus, JSMA may choose an attribute that can be altered a little; nevertheless, it is not worth increasing ϵ. In summary, SICEM cannot be always effective for L_0 attack that does not consider function $C(y, u, l)$ (e.g., JSMA) when comparing the sensitivity scores between two parts that share some attributes.

7.2 Agreement of SICEM

SICEM can evaluate how difficult a given part of an input can be used to create an adversarial example compared to another part or the whole input, especially for L_2 and L_∞ attacks. Further, as seen in the experiments of MNIST and CIFAR-10 dataset, we can conclude that SICEM can work better (higher agreement) with a larger input (i.e., a CIFAR-10 image) than with a smaller input (i.e., a MNIST image), except for L_0 attacks (e.g., JSMA). The reason is that an attack requires less ϵ for a bigger input to find an adversarial example as explained in Sect. 6.1, and SICEM measures an instant sensitivity of a classifier's output with respect to the input. Therefore, a small change by the attack on a big input corresponds to what SICEM measures; then, the larger input is, the higher agreement SICEM can achieve. However, L_0 attacks do not care how much each attribute changes. Thus, they usually add huge perturbation for a pixel. That is, SICEM may not be aligned with L_0 attacks.

Furthermore, we plan to test SICEM with ImageNet [7] to empirically prove that it can perform better in a larger input and improve SICEM not to only depend on the instant sensitivity of the output with respect to the input. However, with the limitation of our resources, we leave this for future work.

7.3 Generative models

Additionally, since a lot of applications depend on generative models based on deep learning (e.g., compression [30], realistic video generation [34] and adversarial neural cryptography [1, 6]), we plan to find an evaluation method for complete and incomplete input adversarial attacks [16, 29] in the future.

7.4 AI ethics statement

First, we have used the dataset (i.e., MNIST and CIFAR-10) which is publicly available through the Tensorflow platform. We did not produce or use any sensitive or private information, and therefore, we do not anticipate any potential harm from using any data from this research.

Further, to ensure that the classifiers work well with the dataset, we utilized architectures of strong classifiers. Also, at first, we started using the simplest attack (i.e., FGSM) to make sure that an adversarial attack works fine for the classifiers and dataset. Then, we implemented more attacks and applied SICEM to them.

Although SICEM can be a guide for an adversary to attack a classifier with limited access to its input, we intend to let the learning classifiers' developers use SICEM to improve the classifiers and make them robust to adversarial attacks. That is, an adversary cannot efficiently exploit SICEM to improve his/her attack if the target classifier's developer is aware of utilizing SICEM to implement his/her classifier.

8 Conclusion

This paper has presented the formulation of state-of-art adversarial attacks for classification models under an incomplete input by inducing occlusion. Further, we have created a Sensitivity-Inspired Constrained Evaluation Method (SICEM) for analyzing any given part of input space and quantify the likelihood of an adversary to produce an adversarial example successfully. While our hypothesis empirically holds for the average case, SICEM seems to experience a certain degree of difficulty when L_0-based attacks are implemented that do not consider the upper and lower bounds of attributes' valid values. SICEM performs well with images from MNIST, and CIFAR-10

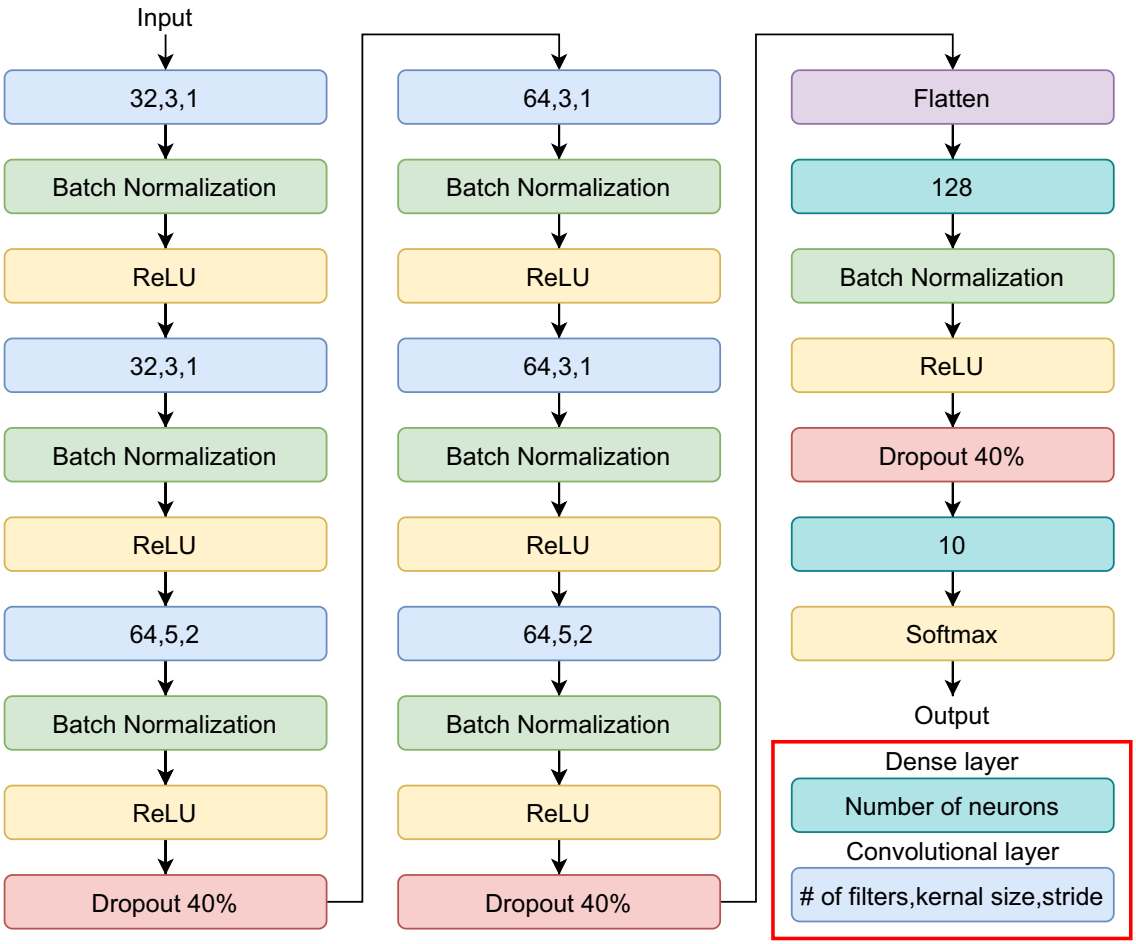


Fig. 12 Architecture of our MNIST classifier. Note that a dense layer and a convolutional layer are defined in the red rectangle

datasets picked at random. Further, SICEM performs significantly better for larger inputs. Further research involves experimenting with ImageNet data to confirm the findings in this paper; then a similar evaluation of generative models [1, 6, 30] under the incomplete input scenarios seems a logical next step.

A Code to reproduce experiments

The code to reproduce all the experiments in this paper can be found in the attached supplementary zip file entitled:

`sicem.zip`

This .zip includes the weights for MNIST and CIFAR-10 trained classifiers; please change the path variable in the code to your working space to avoid overwriting the pre-trained classifiers. Also, beware that training the deep convolutional network, calculating the success rate, and individual-image agreement section consume a significant amount of time. Please make sure you have sufficient time. However, all the results are already shown in the python notebooks, and you do not need to re-run all the experiments. Nonetheless, if you do want to re-run everything, please follow our advice above (Figs. 12, 13).

B Architectures of MNIST and CIFAR-10 classifiers

C Success rate and average required ϵ

Tables 3 and 4 show the success rates and average required ϵ, respectively, to find adversarial examples after performing the adversarial attacks on 200 images of MNIST dataset where t-*attack* denotes that the adversary performs *attack* with mask M_t and b-*attack* denotes that the adversary performs *attack* with mask M_b. Similarly, Tables 5 and 6 shows the success rates and average required ϵ,

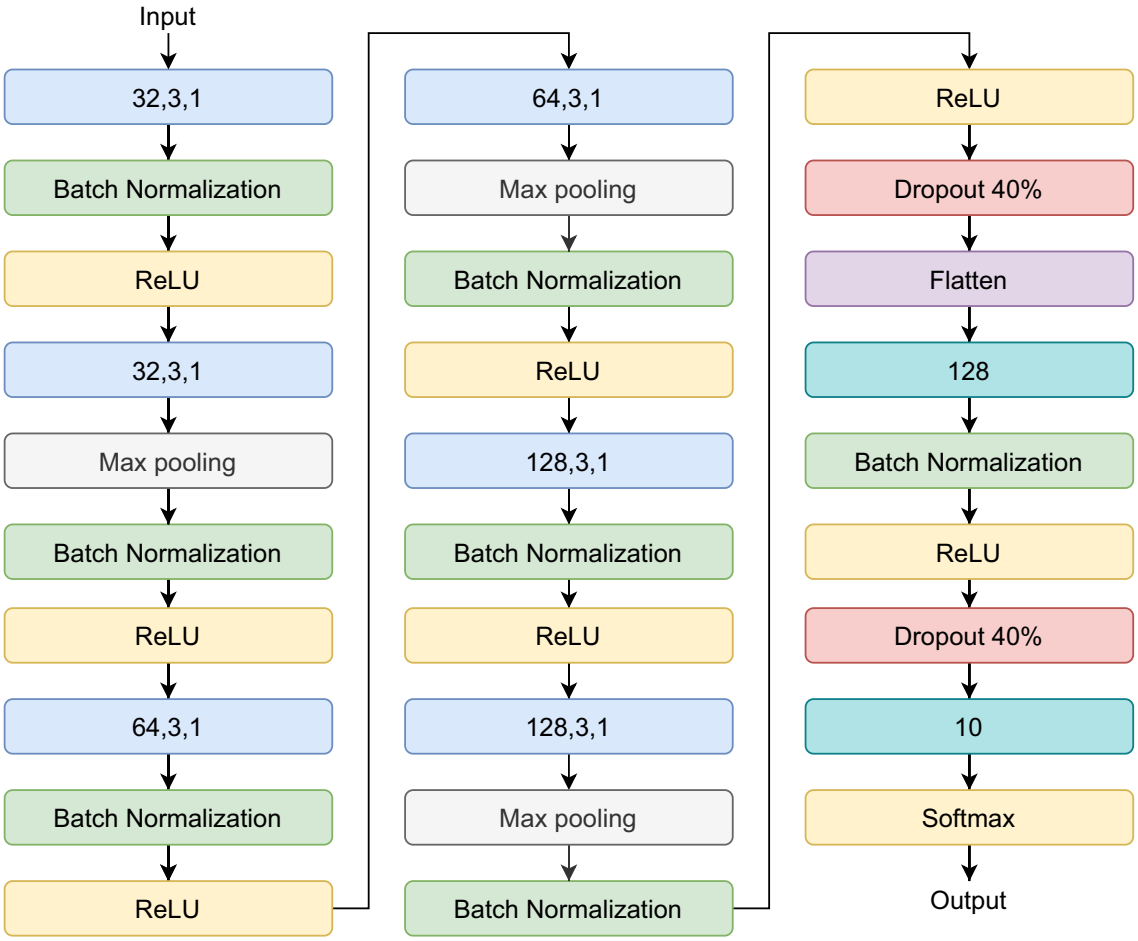


Fig. 13 Architecture of our CIFAR-10 classifier. Note that a dense layer and a convolutional layer are defined in the red rectangle in Fig. 12

Table 3 Success rate of each attack on MNIST

Attack	Class 0 (%)	Class 1 (%)	Class 2 (%)	Class 3 (%)	Class 4 (%)	Class 5 (%)	Class 6 (%)	Class 7 (%)	Class 8 (%)	Class 9 (%)	All (%)
L_0 attack											
JSMA	84.87	99.91	85.37	99.1	98.46	99.89	99.48	99.61	63.12	98.7	92.92
t-JSMA	47.65	99.82	58.82	96.21	98.87	97.3	99.9	79.92	19.52	92.28	79.15
b-JSMA	80.06	99.56	41.86	87.92	64.55	86.95	31.37	99.51	45.76	99.6	74.15
L_∞ attack											
FGSM	27.2	93.46	24.42	46.21	68.03	69.63	29.28	76.28	36.57	75.85	55.18
t-FGSM	1.84	34.19	1.84	23.25	33.09	18.79	6.82	4.82	3.41	27.56	15.78
b-FGSM	6.03	31.98	3.97	7.88	7.99	7.99	3.15	48.13	9.71	22.75	15.39
IGSM	70.14	100	69.09	93.61	97.75	97.41	75.45	99.11	77.38	98.4	87.95
t-IGSM	5.62	88.6	6.1	57.88	61.99	40.94	19.1	14.57	9.09	60.82	37.16
b-IGSM	23.01	81.89	16.38	21.96	21.11	29.81	9.02	92.03	26.24	44.39	37.51
L_2 attack											
CWA	96.01	97.35	95.74	96.51	98.46	95.39	96.33	97.64	98.35	99.2	97.11
t-CWA	90.49	92.67	39.27	94.21	95.59	72.89	84.89	71.16	89.05	97.49	82.77
b-CWA	94.58	85.69	92.44	58.48	92.42	56.92	85.94	95.28	92.77	93.49	85.09

Table 4 Average ϵ of successful attacks on MNIST

Attack	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	All
L_0 Attack											
JSMA	18.51	12.16	19.29	13.75	15.71	13.63	14.38	11.08	15.77	15.31	14.93
t-JSMA	11.4	13.25	14.48	13.76	12.79	16.18	14.44	14.71	5.1	18.41	13.45
b-JSMA	17.15	14.04	10.19	17.38	15.73	16.35	7.09	10.92	11.31	14.99	13.5
L_∞ Attack											
FGSM	0.034	0.092	0.028	0.0472	0.0727	0.076	0.0349	0.0731	0.0407	0.0744	0.0577
t-FGSM	0.0016	0.0416	0.0019	0.0251	0.0366	0.0222	0.0073	0.0043	0.0038	0.0321	0.0179
b-FGSM	0.0069	0.0383.	0.0042	0.0077	0.0083	0.0082	0.0033	0.0514	0.0112	0.0238	0.0168
IGSM	0.0874	0.0814	0.0838	0.0973	0.098	0.0972	0.0921	0.0892	0.088	0.0917	0.0904
t-IGSM	0.0069	0.1054	0.0075	0.0672	0.0722	0.0506	0.0239	0.0163	0.0114	0.0742	0.0444
b-IGSM	0.0305	0.0974	0.0207	0.0266	0.0243	0.0376	0.0109	0.0981	0.0329	0.0493	0.0439
L_2 Attack											
CWA	2.9584	2.0118	3.342	2.6368	2.3762	3.3345	2.8919	2.2474	2.385	2.5532	2.6258
t-CWA	3.2467	2.2268	1.6202	2.5411	2.477	2.6164	2.8617	2.4528	2.5626	2.4433	2.4936
b-CWA	3.0243	2.0205	3.3613	2.281	2.8305	2.2126	2.9567	2.1454	2.6353	2.2222	2.5971

Table 5 Success rate of each attack on CIFAR-10

Attack	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	All
L_0 Attack											
JSMA	97.79%	100%	100%	100%	100%	100%	100%	100%	99.15%	100%	99.68%
t-JSMA	93.15%	58.53%	98.87%	100%	100%	99.87%	100%	99.89%	91.15%	97.46%	93.43%
b-JSMA	95.7%	100%	99.75%	99.76%	99.87%	98.06%	99.78%	99.57%	99.25%	99.67%	99.15%
L_∞ Attack											
FGSM	99.88%	99.58%	98.6%	99.59%	99.87%	100%	98.78%	99.89%	94.14%	99.12%	98.89%
t-FGSM	87.34%	28.95%	80.89%	91%	87.31%	96.38%	87.24%	71.4%	42.54%	88.2%	74.85%
b-FGSM	77.82%	94.63%	72.61%	83.63%	89.47%	92.24%	64.04%	82.56%	73.67%	63.95%	79.21%
IGSM	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
t-IGSM	100%	98.84%	100%	100%	100%	100%	100%	100%	99.89%	100%	99.86%
b-IGSM	100%	100%	100%	100%	100%	99.87%	100%	100%	100%	100%	99.99%
L_2 Attack											
CWA	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
t-CWA	99.88%	100%	100%	100%	100%	100%	100%	100%	99.89%	100%	99.98%
b-CWA	99.88%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99.99%

Table 6 Average ϵ of successful attacks on CIFAR-10

Attack	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	All
L_0 Attack											
JSMA	37.13	27.88	15.7	16.23	12.77	16.87	24.22	23.57	25.35	31.87	23.61
t-JSMA	37.48	38.39	16.9	17.23	12.94	18.14	22.4	32.59	49.1	33.99	28.74
b-JSMA	44.93	27.76	23.29	21.21	19.28	26.91	31.8	29.24	22.5	40.46	29.01
L_∞ Attack											
FGSM	0.0229	0.0225	0.0266	0.0213	0.0208	0.0211	0.0202	0.0225	0.032	0.0238	0.0235
t-FGSM	0.0425	0.0068	0.026	0.0256	0.027	0.0254	0.032	0.0226	0.0246	0.0441	0.0276

Table 6 (continued)

Attack	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	All
b-FGSM	0.0197	0.0328	0.0283	0.0244	0.0277	0.0337	0.0166	0.0338	0.033	0.0199	0.027
IGSM	0.0222	0.0223	0.0227	0.021	0.0204	0.021	0.0206	0.022	0.0259	0.0226	0.0221
t-IGSM	0.0367	0.0541	0.03	0.0252	0.0258	0.0244	0.0282	0.0351	0.0522	0.0383	0.0357
b-IGSM	0.0293	0.0315	0.0346	0.0286	0.0266	0.0332	0.0324	0.0359	0.0339	0.0351	0.0322
L_2 Attack											
CWA	0.1712	0.259	0.1378	0.1076	0.1014	0.1303	0.1525	0.2156	0.1911	0.2252	0.1731
t-CWA	0.2777	0.4817	0.1929	0.1408	0.1408	0.1673	0.2064	0.3186	0.3392	0.3541	0.27
b-CWA	0.2268	0.3476	0.2171	0.1761	0.1557	0.2561	0.2556	0.3247	0.2465	0.3146	0.2561

respectively, after performing the attacks on 200 images of CIFAR-10 dataset.

Author Contributions K. Sooksatra executed the research. P. Rivas directed the research.

Funding This material is based upon work supported by the National Science Foundation under Grant CHE-1905043 and CNS-2136961.

Availability of data and material For our experiments, we use MNIST and CIFAR-10 data, which are publicly available.

Code availability Code is made available as explained in Appendix A.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Abadi M, Andersen DG (2016) Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610.06918
2. Browne MW (2000) Cross-validation methods. J Math Psychol 44(1):108–132
3. Carlini N, Athalye A, Papernot N, Brendel W, Rauber J, Tsipras D, Goodfellow I, Madry A, Kurakin A (2019) On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705
4. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy (sp), pp. 39–57. IEEE
5. Cohen J, Rosenfeld E, Kolter Z (2019) Certified adversarial robustness via randomized smoothing. In: international conference on machine learning, pp. 1310–1320. PMLR
6. Coutinho M, de Oliveira Albuquerque R, Borges F, Garcia Villalba LJ, Kim TH (2018) Learning perfectly secure cryptography to protect communications with adversarial neural cryptography. Sensors 18(5):1306
7. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. IEEE
8. Dong Y, Fu QA, Yang X, Pang T, Su H, Xiao Z, Zhu J (2020) Benchmarking adversarial robustness on image classification. In: proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 321–331
9. Etmann C, Lunz S, Maass P, Schönlieb CB (2019) On the connection between adversarial robustness and saliency map interpretability. arXiv preprint arXiv:1905.04172
10. Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao C, Prakash A, Kohno T, Song D (2018) Robust physical-world attacks on deep learning visual classification. In: proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1625–1634
11. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572
12. Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P (2017) Adversarial examples for malware detection. In: European symposium on research in computer security, pp. 62–79. Springer
13. Hardy W, Chen L, Hou S, Ye Y, Li X (2016) Dl4md: A deep learning framework for intelligent malware detection. In: proceedings of the international conference on data mining (DMIN), p. 61. The steering committee of the World congress in computer science
14. Ilyas A, Santurkar S, Tsipras D, Engstrom L, Tran B, Madry A (2019) Adversarial examples are not bugs, they are features. In: Advances in neural information processing systems, pp. 125–136
15. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: international conference on machine learning, pp. 448–456. PMLR
16. Kos J, Fischer I, Song D (2018) Adversarial examples for generative models. In: 2018 IEEE security and privacy workshops (spw), pp. 36–42. IEEE
17. Krizhevsky A, Hinton G, et al. (2009) Learning multiple layers of features from tiny images
18. Kurakin A, Goodfellow I, Bengio S (2016) Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533
19. LeCun Y (1998) The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/
20. Lopez MM, Kalita J (2017) Deep learning applied to nlp. arXiv preprint arXiv:1703.03091
21. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2017) Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083
22. Mangla P, Singh V, Balasubramanian VN (2020) On saliency maps and adversarial robustness. In: joint European conference on machine learning and knowledge discovery in databases, pp. 272–288. Springer
23. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings.

In: 2016 IEEE European symposium on security and privacy (EuroS &P), pp. 372–387. IEEE
24. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE symposium on security and privacy (SP), pp. 582–597. IEEE
25. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
26. Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23(5):828–841
27. Sun Y, Yin J, Wu C, Zheng K, Niu X (2021) Generating facial expression adversarial examples based on saliency map. Image Vis Comput 116:104318
28. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2013) Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199
29. Tabacof P, Tavares J, Valle E (2016) Adversarial images for variational autoencoders. arXiv preprint arXiv:1612.00155
30. Theis L, Shi W, Cunningham A, Huszár F (2017) Lossy image compression with compressive autoencoders. arXiv preprint arXiv:1703.00395
31. Vinayakumar R, Alazab M, Soman K, Poornachandran P, Venkatraman S (2019) Robust intelligent malware detection using deep learning. IEEE Access 7:46717–46738
32. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep learning for computer vision: a brief review. Comput Intell Neurosci **2018**
33. Wang S, Gong Y (2021) Adversarial example detection based on saliency map features. Appl Intell pp. 1–14
34. Wang TC, Liu MY, Zhu JY, Liu G, Tao A, Kautz J, Catanzaro B (2018) Video-to-video synthesis. arXiv preprint arXiv:1808.06601
35. Xin Y, Kong L, Liu Z, Chen Y, Li Y, Zhu H, Gao M, Hou H, Wang C (2018) Machine learning and deep learning methods for cybersecurity. IEEE Access 6:35365–35381
36. Xu W, Evans D, Qi, Y (2017) Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155
37. Yuan Z, Lu Y, Xue Y (2016) Droiddetector: android malware characterization and detection using deep learning. Tsinghua Sci Technol 21(1):114–123
38. Zheng S, Song Y, Leung T, Goodfellow I (2016) Improving the robustness of deep neural networks via stability training. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4480–4488