



# Uncertainty as a Swiss army knife: new adversarial attack and defense ideas based on epistemic uncertainty

Omer Faruk Tuna<sup>1</sup> · Ferhat Ozgur Catak<sup>2</sup> · M. Taner Eskil<sup>1</sup>

Received: 25 March 2021 / Accepted: 20 February 2022  
© The Author(s) 2022

## Abstract

Although state-of-the-art deep neural network models are known to be robust to random perturbations, it was verified that these architectures are indeed quite vulnerable to deliberately crafted perturbations, albeit being quasi-imperceptible. These vulnerabilities make it challenging to deploy deep neural network models in the areas where security is a critical concern. In recent years, many research studies have been conducted to develop new attack methods and come up with new defense techniques that enable more robust and reliable models. In this study, we use the quantified epistemic uncertainty obtained from the model's final probability outputs, along with the model's own loss function, to generate more effective adversarial samples. And we propose a novel defense approach against attacks like Deepfool which result in adversarial samples located near the model's decision boundary. We have verified the effectiveness of our attack method on MNIST (Digit), MNIST (Fashion) and CIFAR-10 datasets. In our experiments, we showed that our proposed uncertainty-based reversal method achieved a worst case success rate of around 95% without compromising clean accuracy.

**Keywords** Adversarial Machine Learning · Uncertainty · Security · Deep Learning

تعلم الآلة العدائية، ربيبة، حماية، تعلم عميق

## Introduction

In the last few years, deep learning models began to exceed human-level performances. For instance, in 2015, a deep learning model called ResNet [1] beat the human performance in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and the record was beaten by more advanced architectures later on. Similarly, Goodfellow et al. [2] proposed a system which outperforms human operators for the problem of reading address information from Google Street View imagery or solving CAPTCHAS. In the domain of game playing, an AI software named AlphaGo defeated the world Go champion in 2016 [3]. Today, we

observe that many advanced systems are being built upon deep learning models, offering a very high degree of success in various domains including medical diagnosis [4,5], autonomous vehicles [6,7], game playing [8] and machine translation [9,10]. However, during the rise of deep neural network (DNN), the researchers' main focus was to build more and more accurate models, and nearly no particular attention was paid to the robustness and reliability of these models. Deep learning models indeed require a more detailed evaluation since these models have some intrinsic vulnerabilities that let intruders easily exploit them.

By the end of 2013, researchers have brought to light that existing DNN models are vulnerable to carefully crafted attacks. Szegedy et al. [11] were among the first who observed the presence of adversarial examples in the image classification domain. The authors have shown that it is possible to perturb an image by a miniscule amount to change the decision of a deep learner. It turns out that a very small and quasi-imperceptible perturbation of input is sufficient to fool the most advanced classifiers and results in wrong classification. Since then, a great number of studies have been conducted in this new research field called *Adversarial*

✉ Omer Faruk Tuna  
omer.tuna@isikun.edu.tr

Ferhat Ozgur Catak  
f.ozgur.catak@uis.no

M. Taner Eskil  
taner.eskil@isikun.edu.tr

<sup>1</sup> Isik University, Istanbul, Turkey

<sup>2</sup> University of Stavanger, Rogaland, Norway

*Machine Learning* and these studies were not limited only to the image classification domain. For instance, in the NLP domain, Sato et al. [12] showed that it is possible to fool a sentiment analysis model which is trained on textual data by just changing only one word from the input sentence. Another example is in the audio domain [13], in which the authors constructed targeted adversarial audio samples in automatic speech recognition task by adding a very small perturbation to the original waveform. This study demonstrated that the target model can easily be manipulated to transcribe the input as any chosen phrase.

Adversarial machine learning attacks are based on perturbation of the input instances in a direction that maximizes the chances of wrong decision making, resulting in false predictions. These attacks can lead to a loss of the model's prediction performance as the algorithm can not predict the real output of the input instances correctly. Thus, attacks utilizing the vulnerability of DNNs can seriously undermine the security of these machine learning (ML) based systems, sometimes with devastating consequences. In medical applications, the perturbation attack might lead to an incorrect diagnosis of a disease. Consequently, it can cause severe harm to the patient's health and also damage the health-care economy [14]. Likewise, autonomous cars use ML to drive in traffic without human intervention. A wrong decision based on an adversarial attack for the autonomous vehicle could cause a fatal accident [15,16]. Hence, defending against adversarial attempts and increasing the robustness of ML models without compromising clean accuracy is of crucial importance. Assuming that these ML models will serve in critical areas, we should pay the greatest attention to not only ML models' performance but also the security concerns of these architectures.

In this study, we concentrate on adversarial attack and defense strategies based on epistemic uncertainty instead of traditional approaches which are solely based on the model's loss function. Up until now, the most prominent approach in adversarial attack studies is based on the maximization of model loss and aims to create carefully crafted adversarial counterparts of the given input. However as shown in Ref. [17], directions pointed by the model's loss gradient may not be accurate due to the inherent and unavoidable approximation error of the trained model. For this reason authors needed to validate them with the information obtained from the uncertainty's gradient. The new approach combines the traditional approach's strengths with the measure of uncertainty to produce more effective attacks.

On the defense side, we propose a method in which before feeding any input to the DNN model, we try to revert it back to its original data manifold by minimizing its quantified uncertainty. Traditional approaches like model loss-based metrics would not successfully push the malicious input to its original data manifold. Real label information could not be used

because it is unknown by the model beforehand in inference time. On the other hand, the predicted label could not be used since any minimization attempt based on the predicted label would result in model wrongly classifying adversarial inputs even with more confidence. However, since uncertainty quantification metrics does not depend on the label information, by minimizing the uncertainty value, one can revert the input instance back to its original data manifold. Our codes are released on GitHub<sup>1</sup> for reproduction.

To summarize; our main contributions for this paper are:

- We introduce a novel attack method by utilizing the model's epistemic uncertainty, which yields more powerful adversarial impact with less amount of perturbation at each step.
- We introduce a new adversarial defense technique which provides a very high degree of robustness against some of the strongest attacks like Deepfool attack and Carlini and Wagner attack (under default setting). We tested the effectiveness of our defensive approach on both clean and perturbed data and verified that it does not have a negative effect on legitimate (clean) examples.

This study is organized as follows: "Related work" will introduce some of the known attack types and defense techniques in the literature. In "Preliminaries", we will introduce the concept of uncertainty together with main types and discuss how we can quantify epistemic uncertainty. "Approach" will give the details of our approach. We will present our experimental results in "Results" and conclude our work in "Conclusion".

## Related work

Since the discovery of DNN's vulnerability to adversarial attacks [11], a vast amount of research has been conducted in both developing new adversarial attacks and defending against these attacks with more robust DNN models [18–21]. We will treat the attack and defense studies separately and review some of the notable ones in "Adversarial attacks" to "Adversarial defense".

## Adversarial attacks

Deep learning models contain many vulnerabilities and weaknesses which make them difficult to defend in the context of adversarial machine learning. For instance, they are often sensitive to small changes in the input data, resulting in unexpected results in the model's final output. Figure 1

<sup>1</sup> <https://github.com/omerfaruktuna/uncertainty-based-attack-defense>.

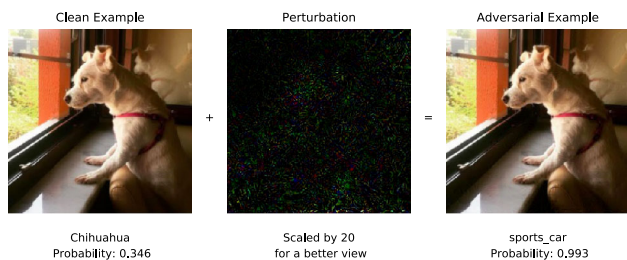


Fig. 1 An example of adversarial attack

shows how an adversary would exploit such a vulnerability and manipulate the model through the use of carefully crafted perturbation applied to the input data. The malicious perturbation is applied upon the original image and it manipulates the model in such a way that a “Chihuahua (Dog)” is wrongly classified as “Sports Car” with very high degree of confidence.

The attack strategies are mainly based on perturbing the input instance to maximize the model’s loss value in the prediction time. Many adversarial attack algorithms have been proposed in the literature in the past few years. The well-known adversarial attacks are Fast-Gradient Sign Method, Iterative Gradient Sign Method, Projected Gradient Descent, Jacobian Based Saliency Map, Carlini&Wagner, and DeepFool. “Fast-gradient sign method” to “Deepfool attack” briefly describes these six adversarial machine learning attacks.

### Fast-gradient sign method

Fast-gradient sign method (FGSM) [22] is one of the earliest and most popular adversarial attacks in adversarial machine learning. FGSM utilizes the derivative of the model’s loss function with respect to the input image to determine in which direction the pixel values of the input image should be altered to minimize the loss function of the model. Once this direction is extracted, it changes all pixels simultaneously in the adverse direction to maximize the loss value of the prediction. For a model with classification loss function described as  $L(\theta, \mathbf{x}, y)$  where  $\theta$  represents the parameters of the model,  $\mathbf{x}$  is the benign input to the model (sample input image in our case),  $y_{\text{true}}$  is the actual label of our input, we can generate adversarial samples using the formula below:

$$\mathbf{x}^* = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y_{\text{true}})). \tag{1}$$

One last key point about FGSM is that it is not designed to be optimal but fast. That means it is not designed to produce the minimum required adversarial perturbation. Besides, this method’s success ratio is relatively low in small  $\epsilon$  values compared to other attack types.

### Iterative gradient sign method

Kurakin et al. [23] proposed a small but effective improvement to the FGSM. In this approach, rather than taking only one step of size  $\epsilon$  in the gradient sign’s direction, the attacker takes several but smaller steps  $\alpha$ , and use the given  $\epsilon$  value to clip the result. This attack type is often referred to as basic iterative method (BIM), and it is merely FGSM applied to an input image iteratively. Generating perturbed images under  $L_{\text{inf}}$  norm for BIM attack is given by Eq. 2.

$$\begin{aligned} \mathbf{x}^* &= \mathbf{x} \\ \mathbf{x}_{N+1}^* &= \mathbf{x} + \text{Clip}_{x,\epsilon} \{ \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}_N^*, y_{\text{true}})) \}, \end{aligned} \tag{2}$$

where  $\mathbf{x}$  is the input sample,  $\mathbf{x}^*$  is the produced adversarial sample at  $i$ th iteration,  $L$  is the loss function of the model,  $y_{\text{true}}$  is the actual label for input sample,  $\epsilon$  is a tunable parameter, limiting maximum level of perturbation in given  $l_{\text{inf}}$  norm, and  $\alpha$  is the step size.

The success ratio of BIM attack is higher than the FGSM [24]. By adjusting the  $\epsilon$  parameter, the attacker can have a chance to manipulate how far an adversarial sample will be pushed past the decision boundary.

One can group BIM attacks under two main types, namely BIM-A and BIM-B. In the former type, we stop iterations as soon as we succeed in fooling the model (passing the decision boundary), while in the latter, we continue the attack till the end of the provided number of iterations so that we push the input further away the decision boundary.

### Projected gradient descent

This method, also known as PGD, has been introduced by Madry et al. [25]. It perturbs a clean image  $\mathbf{x}$  for several numbers of  $i$  iterations with a small step size in the direction of the gradient of the model’s loss function. Different from BIM, after each perturbation step, it projects the resulting adversarial sample back onto the  $\epsilon$ -ball of input sample, instead of clipping. Moreover, instead of starting from the original point ( $\epsilon = 0$ , in all dimensions), PGD uses random start, which can be described as:

$$\mathbf{x}_0 = \mathbf{x} + U(-\epsilon, +\epsilon), \tag{3}$$

where  $U(-\epsilon, +\epsilon)$  is the uniform distribution between  $(-\epsilon, +\epsilon)$ .

### Jacobian-based saliency map attack (JSMA)

This method, also known as JSMA, has been proposed by Papernot et al. [26]. It is designed to be used under  $L_0$  distance norm which takes total number of altered pixels into count, thereby restricting the attacker. It is a greedy algorithm

which selects two pixels at a time. The algorithm utilizes the gradient  $\nabla Z(x)_l$  to compute a saliency map, which shows each pixel's impact on the classification of each class. And the aim is to enhance the possibility of the target class while diminishing the possibility of other classes by selecting and updating two pixels at a time based on the saliency map. The attack is continued until either a predefined number of pixels is modified or the model is successfully fooled.

### Carlini and Wagner attack

This attack type has been introduced by Carlini and Wagner [27], and it is one of the most powerful attack types to date. Therefore, it is generally used as a benchmark for the adversarial defense research community that aims to create more robust DNN architectures resistant to adversarial attempts. CW attack achieves higher success rates on normally trained models for most well-known datasets. It can fool defensively distilled models as well, on which other attack types barely succeed in crafting adversarial examples.

The authors redefine the adversarial attack as an optimization problem which can be solved using gradient descent to craft more powerful and effective adversarial samples under different  $L_p$  norms. The algorithm has a parameter called *confidence* which is used to adjust the gap between the crafted adversarial example and decision boundary of the model. If one applies the attack with default settings, confidence parameter will be set to 0 and the algorithm will stop as soon as it crafts an adversarial sample after passing the decision boundary. There is a trade-off between producing more confident adversarial samples and achieving the highest possible attack success rate.

### Deepfool attack

This attack type has been proposed by Moosavi-Dezfooli et al. [28] and it is one of the powerful untargeted attack types in literature. It is designed to be used in different distance norm metrics such as  $L_{\text{inf}}$  and  $L_2$  norms.

Deepfool attack has been designed based on the assumption that neural network models behave as a linear classifier and the classes are separated by a hyperplane. The algorithm starts from the initial input point  $\mathbf{x}_t$  and at each iteration, it calculates the closest hyperplane and the minimum perturbation amount, which is the orthogonal projection to the hyperplane. Then the algorithm calculates  $\mathbf{x}_{t+1}$  by adding the minimal perturbation to the  $\mathbf{x}_t$  and checks whether misclassification is achieved.

There is only a very limited number of studies that make use of uncertainty to craft adversarial examples to our knowledge. Liu et al. [29] proposed a universal adversarial perturbation (UAP) method that utilizes a metric called virtual Epistemic uncertainty obtained from the model's

structural activation rather than from the final softmax scores. However, estimating the model's uncertainty involves aggregating all the neurons' virtual Epistemic uncertainties, which is computationally costly. Tuna et al. [17] proposed several iterative attack variants based on the model's quantified epistemic uncertainty. But, the perturbation introduced at each iteration in their most effective hybrid approach is larger than the conventional loss-based BIM attack. Qu et al. [30] proposed an adversarial attack method against deep reinforcement policies using entropy of action distribution to quantify uncertainty and used it to identify the frames that were most vulnerable to attack. It is known that DNN models suffer from over or under-confidence predictions [31,32]. Therefore, quantifying uncertainty based only on entropy derived from the final softmax score output of the model might not always be the best option.

### Adversarial defense

In this section, we summarize some of the important adversarial defense approaches proposed in recent years.

#### Defensive distillation

This technique was proposed by Papernot et al. [33]. The first step of the proposed algorithm is to train a model called *teacher model* on the training set using a high temperature ( $T$ ) value in the softmax function. Then, this previously trained teacher model is used to label each of the samples in the training set with soft labels computed in prediction time. Next, the *distilled model* is trained using the soft labels that we obtained from the teacher model again using high temperature ( $T$ ) value in the softmax. In addition to increasing clean data accuracy on the test set, the Distillation technique was also shown to reduce the success rate of JSMA attacks' ability to craft adversarial examples considerably. However, later, it was shown that more effective attack types like CW attack could easily break the defensive distillation technique.

#### Adversarial training

Adversarial training is considered as an intuitive defense approach in which the robustness of a DNN model is enhanced by training the model with adversarial data samples. We can mathematically represent this approach as a Minimax game as in Eq. 4:

$$\min_{\theta} \max_{\|\delta\| \leq \epsilon} \ell(h_{\theta}(x + \delta), y), \quad (4)$$

where  $h$  is our model,  $\ell$  is the loss function of the model,  $\theta$  is the weights of the model and  $y$  is the actual label.  $\delta$  represents the amount of adversarial perturbation added to



benign input  $x$  and it is limited with some  $\epsilon$  value. Maximization of the inner objective is accomplished by applying the strongest attack possible which is generally approximated by some known adversarial attack types. The outer minimization objective is used to train the model to minimize the resulted loss from the inner maximization step. The outcome of this process is a model which is expected to be resistant to adversarial attacks used in the model training phase. Goodfellow et al. [22] used adversarial samples generated by FGSM attack for adversarial training, whereas Madry et al. [25] used PGD attack to produce more robust models but at the cost of more computational resource consumption. Although adversarial training is accepted to be one of the most powerful defenses against adversarial attacks [34,35], models that are adversarially trained are still vulnerable to attacks like CW.

### Magnet

Meng et al. [36] proposed a defense method which consists of two components: detector and reformer. The former is used to inspect input samples and determine if they are benign or not and the latter is used to take inputs classified as benign by the detector and reform them to remove any remaining adversarial nature. Although the authors show the effectiveness of their defense against different adversarial attacks, later it was shown that their defense method is vulnerable to the CW attack [37].

Adversarial ML is a highly active research area and we see new adversarial defense techniques are being proposed intensely which we could not mention here. Some notable ones are [38–40].

### Preliminaries

Traditionally, predictive models used to be forced to decide even in ambiguous cases where the model is not sure about its prediction. The quality of such predictions is expected to be low. Assuming the model’s prediction is always correct without any reasoning on the model’s uncertainty may result in catastrophic results. This fact led the researchers to suggest abstaining models based on certain conditions like when the model’s uncertainty is high, thus improving the reliability [41,42].

In this section, we will first introduce the two types of uncertainty in machine learning. And then, we will present how we can quantify Epistemic Uncertainty in the context of deep learning.

### Uncertainty in machine learning

There are two different types of uncertainty in machine learning: epistemic uncertainty and aleatoric uncertainty [43–45].

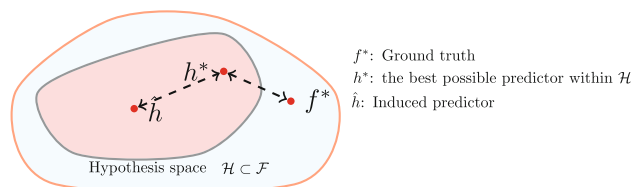


Fig. 2 Different types of epistemic uncertainty

### Epistemic uncertainty

Epistemic uncertainty refers to uncertainty caused by a lack of knowledge and limited data needed for a perfect predictor [46]. It can be categorized under 2 groups as *approximation uncertainty* and *model uncertainty* as depicted in Fig. 2.

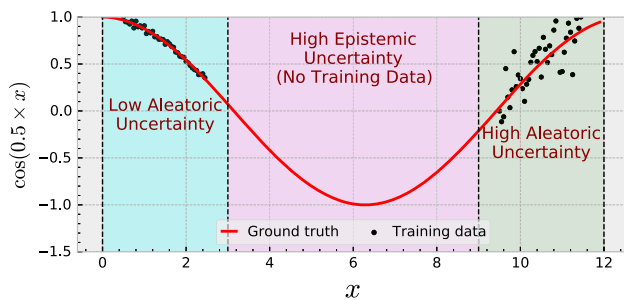
#### Approximation uncertainty

In a conventional machine learning task, the learner is given data points from an independent, identically distributed dataset. Then he/she tries to induce a hypothesis  $\hat{h}$  from the hypothesis space  $\mathcal{H}$  by picking a proper learning method with its related hyperparameters and minimizing the expected loss (risk) with a selected loss function,  $\ell$ . However, what the learner does is to try to minimize the *empirical risk*  $R_{emp}$  which is an estimate of real risk  $R(h)$ . The induced  $\hat{h}$  is an approximation of the  $h^*$  which is the optimum hypothesis within  $\mathcal{H}$  and the real risk minimizer. This fact results in an approximation uncertainty. Therefore, the induced hypothesis’s quality is not perfect, and the learned model will always be prone to errors.

#### Model uncertainty

Suppose the chosen hypothesis space  $\mathcal{H}$  does not include the perfect predictor. In that case, the learner has no chance to realize his/her object of discovering a hypothesis function that can successfully map all possible inputs to outputs. This drives to an inconsistency between the ground truth  $f^*$  and the best possible function  $h^*$  within  $\mathcal{H}$ , called model uncertainty.

However, Universal Approximation Theorem states that for any target function  $f$ , a neural network can approximate  $f$  [47,48]. The hypothesis space  $\mathcal{H}$  is huge for deep neural networks. Hence it will not be wrong to assume that  $h^* = f^*$ . One can disregard the model uncertainty for deep neural networks, and may only care about the approximation uncertainty. Consequently, in deep learning tasks, the actual source of epistemic uncertainty is linked to approximation uncertainty. Epistemic uncertainty points to the confidence a model has about its prediction [49]. The underlying cause is the uncertainty about the parameters of the model. This type of uncertainty is apparent in the regions where we have limited training data and the model weights are not optimized correctly.



**Fig. 3** Illustration of the epistemic and aleatoric uncertainty

### Aleatoric uncertainty

Aleatoric uncertainty refers to the variability in an experiment's outcome, which is due to the inherent random effects [50]. This type of uncertainty can not be reduced albeit having enough training samples [51]. An outstanding exemplification of this phenomenon is the noise observed in the measurements of a sensor.

Figure 3 shows a simple nonlinear function ( $\cos(0.5 \times x)$  where  $x \in [0, 12]$ ) plot. As shown in the region where data points are populated at right ( $9 < x < 12$ ), the noisy samples are clustered, leading to high aleatoric uncertainty. For example, these points may represent a faulty sensor measurement; one can conclude that the sensor produces errors around  $x = 10.5$  for some inherent reason. We can also conclude that the middle regions of the figure represent the high epistemic uncertainty areas. Because there are not enough training samples for our model to describe the data best. Moreover, we can claim that the high epistemic uncertainty area represents the low prediction accuracy area.

### Quantifying epistemic uncertainty in deep neural networks

Using techniques that help us to quantify the uncertainty of the model is necessary for healthy decision making. Assuming that we, as humanity, will use deep learning models in the areas where safety and reliability is a critical concern, such as autonomous driving, medical applications, researchers need to be very careful and pay utmost attention to prediction uncertainty. This will help us to increase the quality of the predictions.

In recent years, many research studies have been conducted to quantify uncertainty in deep learning models. Most of the work was based on Bayesian Neural Networks, which learn the posterior distribution over weights to quantify predictive uncertainty [52]. However, the Bayesian NN's come with additional computational cost and inference issue. Therefore, several approximations to Bayesian methods have been developed which make use of variational inference [53–56]. On the other hand, Lakshminarayanan et al. [57] used

the deep ensemble approach as an alternative to Bayesian NN's to quantify predictive uncertainty. But this requires training several NN's which may not be feasible in practice. A more efficient and elegant approach was proposed by Gal et al. [58]. The authors showed that a neural network model with inference time dropout is equivalent to a Bayesian approximation of the Gaussian process. And the prediction hypothesis uncertainty can be approximated by averaging probabilistic feed-forward Monte Carlo dropout sampling during the prediction time.

This method resembles the ensemble approach in machine learning. In each single ensemble model, the system has to drop out different neurons in each layer according to the dropout ratio in the prediction time. Let  $\mathcal{D} = (x_i, y_i)_{i=1}^N$  is the dataset consisting of input samples  $x_i \in \mathbb{R}^d$  and one-hot encoded outputs  $y_i \in \mathbb{R}^k$ ,  $\theta$  denotes the model weights and  $T$  is the number of predictions of the MC dropouts, then the prediction of a model for any test sample  $\hat{x}$  given the weights of the model is denoted by  $p(\hat{y}^{(k)} = 1 | \hat{x}, \theta, \mathcal{D})$ , and it is a vector of output probability scores for  $k$  classes. The predictive mean is the average of the prediction softmax scores over dropout iterations,  $T$ , and the predictive mean is used as the final output probability distribution, for the input instance  $\hat{x}$  in the dataset. The overall prediction uncertainty is approximated by computing the variance of the probabilistic feed-forward Monte Carlo (MC) dropout sampling during prediction time. The prediction mean is described as follows:

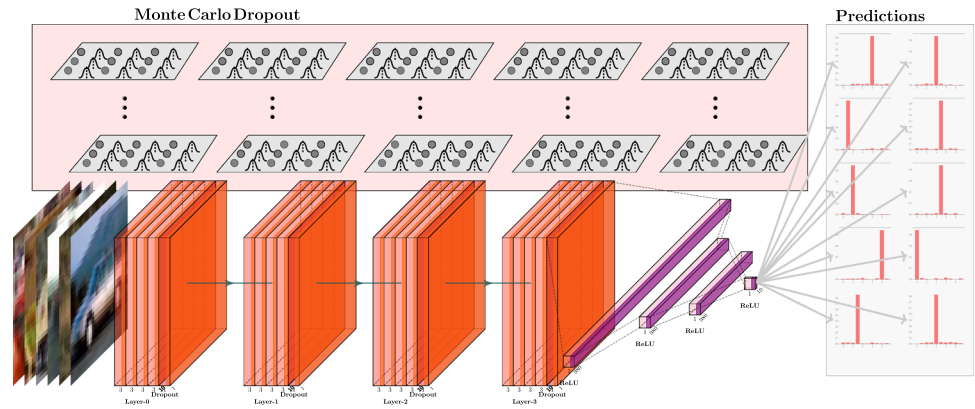
$$\hat{\mu}_{\text{pred}} = \frac{1}{T} \sum_{t=1}^T p(\hat{y}^{(k)} = 1 | \hat{x}, \theta, \mathcal{D}) \quad (5)$$

The final label of input instance  $\mathbf{x}$  can be estimated by finding the argmax of the mean of MC dropouts predictions, which will be done  $T$  times.

Figure 4 presents the general overview of the Monte Carlo dropout-based classification algorithm. In the prediction time, random neurons in each layer are dropped out (based on the dropout ratio  $p$ ) from the base neural network model to create a new model. As a result,  $T$  different classification models can be used to predict the input instance's class label and uncertainty of the overall prediction. The predicted output score is assigned with the highest of the prediction mean for each testing input sample  $\mathbf{x}$ . The variance of the  $p(\hat{y})$  is used to quantify epistemic uncertainty.

We have chosen the MC dropout method due to its simplicity and effectiveness. Although it requires a certain number of feed-forward queries, it is still more efficient than using Bayesian networks or variational inference techniques which compute or approximate the posterior distribution of weights to quantify predictive uncertainty. The approach needs only a single trained model to measure the prediction's uncertainty, while other techniques such as deep ensemble need multiple

**Fig. 4** Illustration of the Monte Carlo dropout-based Bayesian prediction



models. Secondly, since the function used to calculate the variance is convex and smooth [29], one can take the backward derivative of the computed variance term for each input instance and use it to craft adversarial examples to evade the model.

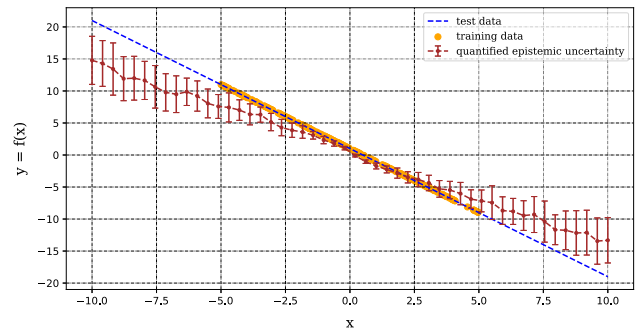
**Approach**

Model uncertainty is higher in the regions where there is a limited number of training instances. Since we lack ground truth in these regions, we cannot obtain a perfect model to predict every possible test data accurately. Figure 5 shows a regression model’s prediction outputs trained on a limited number of data points constrained on some interval. In this simple example, we trained a single hidden layer neural network with fifteen neurons to learn a linear function  $y = -2 \times x + 1$ . As can be seen from the figure, in the areas where we do not have any training examples, the uncertainty values obtained from MC dropout estimates of the model are high, which can be interpreted as the quality of the prediction is low, and the model is having difficulties in deciding the correct output values. Harmoniously, we observe high loss values in these areas. For this reason, one can conclude that the high epistemic uncertainty area represents the low prediction accuracy area. Consequently, pushing the model’s limits by testing it in extreme conditions with input that the model has never seen before will result in failure of model prediction output [17]. Likewise, pulling the input instances back to the regions where the model has been trained on will result in more accurate predictions.

The aim of adversarial attacks is to find the least perturbation amount ( $\delta$ ) constrained to some interval ( $\epsilon$ ), resulting in maximum loss, thus fooling the classifier. We can express this mathematically as in Eq. 6, where  $h_\theta(\mathbf{x})$  is our neural network model.

$$\arg \max_{\|\delta\| \leq \epsilon} \ell(h_\theta(x + \delta), y) \tag{6}$$

In most attack types, the attackers perturb the input instances in a direction that maximizes the loss, and this



**Fig. 5** Uncertainty values obtained from a regression model

direction is calculated using the gradient of the loss function. However, it was shown that instead of using the model’s loss function, a practical approach would be to use the model’s epistemic uncertainty [17]. We propose an attack method that combines the model’s epistemic uncertainty and its loss as tools for creating successfully manipulated adversarial input instances.

**Proposed epistemic uncertainty-based attack and defense**

Existing attacks have been designed to exploit the model loss and aimed at maximizing the model loss value within a constrained neighbourhood of the input data points. However, one possible drawback for these kinds of attacks is that they solely rely on the trained ML model, which inevitably suffers from the approximation error. We can overcome this problem by utilizing an additional metric, namely epistemic uncertainty of the model. This additional uncertainty information has a correcting effect and sometimes points to the directions which yield higher loss values [17]. At each iteration step of adversarial sample generation, we will make use of the gradient of quantified epistemic uncertainty as in Eq. 7 in addition to model loss as in Eq. 8.

Example of simple uncertainty-based attack:

$$\mathbf{x}^* = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} U(\mathbf{x}, h, p, T)). \tag{7}$$

Example of simple loss-based attack:

$$\mathbf{x}^* = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}, y))), \quad (8)$$

where  $\mathbf{x}$  is the input (benign) instance,  $\mathbf{x}^*$  is the perturbed instance,  $U$  is the uncertainty metric (mean variance) obtained from  $T$  different MC dropout estimates,  $h$  is the prediction model,  $p$  is the dropout ratio used in the dropout layers and  $T$  is the number of MC dropout samples in model training mode.

Consistent with [59], calculation steps for our Uncertainty metric  $U$  (mean variance of  $T$  predictions) is as follows:

**Step 1:** For any input image  $\mathbf{x}$ ,  $T$  different predictions is obtained  $P_t(\mathbf{x})$  by MC Dropout sampling where each prediction is a vector of output probability scores for  $C$  classes.

$$P_t(\mathbf{x}) = h(\mathbf{x}, p, T)$$

where  $h$  is the prediction model in training mode,  $p$  is the dropout ratio used in the dropout layers,  $T$  is the number of MC dropout samples in model training mode

**Step 2:** The next step is to compute mean prediction score for the  $T$  different outputs:

$$P_T(\mathbf{x}) = \frac{1}{T} \sum_{t \in T} P_t(\mathbf{x})$$

**Step 3:** Then, we compute the variance of the  $T$  predictions for each class.

$$\sigma^2(P_T(\mathbf{x})) = \frac{1}{T} \sum_{t \in T} (P_t(\mathbf{x}) - P_T(\mathbf{x}))^2$$

**Step 4:** As a final step, we compute the expected value of variance over all classes by taking their average.

$$U(\mathbf{x}, h, p, T) = E(\sigma^2(P_T(\mathbf{x})))$$

However, we will be using only a subset of directions based on some defined logic. We name our proposed attack algorithm as rectified basic iterative attack (rectified-BIM). Because the direction pointed out by the gradient of the loss function is updated using the reference information obtained from the quantified epistemic uncertainty.

### Rectified basic iterative attack

Algorithm 1 shows the traditional Loss-Based BIM attack's pseudocode. Algorithm 2 shows our proposed uncertainty-based Rectified BIM attack. The algorithm is designed under  $L_\infty$  norm.

---

**Algorithm 1:** Algorithm for BIM attack  $\mathbf{x}$  is the benign image,  $y_{true}$  is the actual label for  $\mathbf{x}$ ,  $h$  is the hypothesis function learnt during training,  $N$  is the number of iterations,  $\epsilon$  is the maximum allowed perturbation,  $\alpha$  is the step size.

---

**Input:**  $\mathbf{x} \in \mathbb{R}^m, y_{true}, h, N, \epsilon, \alpha$

**Output:**  $\mathbf{x}_{t+1}$

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2 while  $n < N$  do
   // update  $\mathbf{x}$  using below formula,  $h$  in
   // evaluation mode
3    $\mathbf{x}_{(t+1)} = \text{clip}_{\mathbf{x}, \epsilon}(\mathbf{x}_t + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))))$ 
4 return  $\mathbf{x}_{t+1}$ 

```

---



---

**Algorithm 2:** Algorithm for Rectified-BIM attack:  $\mathbf{x}$  is the benign image,  $y_{true}$  is the actual label for  $\mathbf{x}$ ,  $h$  is the hypothesis function learnt during training,  $p$  is the dropout ratio used in dropout layers,  $T$  is the number of MC dropout samples in model training mode,  $N$  is the number of iterations,  $\epsilon$  is the maximum allowed perturbation,  $\alpha$  is the step size,

---

**Input:**  $\mathbf{x} \in \mathbb{R}^m, h, p, T, N, \epsilon, \alpha$

**Output:**  $\mathbf{x}_{t+1}$

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2  $condition \leftarrow False$ 
3 while  $n < N$  do
4   Compute  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))$  while  $h$  in evaluation mode
5   Compute  $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$  while  $h$  in training mode
6   if  $\text{arg max}(h(\mathbf{x}_{t+1})) \neq y_{true}$  then
7      $condition = True$ 
8   if  $condition = False$  then
9     Update all elements of  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))$  to 0 where
9      $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true})) \neq \nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$ 
9     // update  $\mathbf{x}$ 
10     $\mathbf{x}_{(t+1)} = \text{clip}_{\mathbf{x}, \epsilon}(\mathbf{x}_t + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))))$ 
11  else
12    Update all elements of  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t), y_{true})$  to 0 where
12     $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true})) = \nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$ 
12    // update  $\mathbf{x}$ 
13     $\mathbf{x}_{(t+1)} = \text{clip}_{\mathbf{x}, \epsilon}(\mathbf{x}_t + \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))))$ 
14 return  $\mathbf{x}_{t+1}$ 

```

---

At each iteration of the Rectified-BIM algorithm, we calculate  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{true}))$  and  $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$ . We then get the sign of these gradient vectors which show the sub-directions for each input dimension. If the sign of the gradient for any input pixel is positive, it means that we can increase loss or uncertainty by increasing the value of that pixel. For any input sample fed to the attack algorithm, we start perturbing the input image by first using the intersection of the sub-directions pointed by the derivative of loss and uncertainty information. This is valid until the input sample is pushed off of the decision boundary. Once the input sample passes the decision boundary, this time we use the



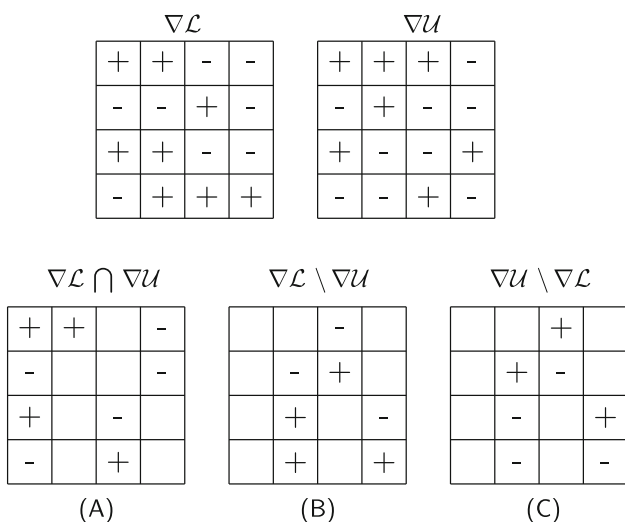


Fig. 6 Sub-directions (sign of the gradients) used in attack and defence purposes

sub-directions of loss’ gradient which are not shared by the uncertainty’s gradient. This idea can be understood better by looking at Fig. 6a, b. In the first part of the proposed attack: instead of trusting only loss or only uncertainty, we trust the information provided by both. This way, we only use a subset of sub-directions. Thus, at each iteration, the amount of perturbation applied to the input sample is being decreased without compromising the adversarial attacking strength.

In the low confidence regions where the input sample is close to the model’s decision boundary, the uncertainty has a friction effect against loss. It is known that the quantified uncertainty is higher near the decision boundaries [42]. Thus, for adversarial attack purposes, perturbing the input sample in the direction of uncertainty’s gradient is not a good idea after passing the decision boundary. This is because the direction in which the gradient of uncertainty points will keep the sample near the boundary regions and prevent the perturbed sample from being pushed far away. Therefore, for the second part of our Rectified-BIM Attack, we consciously decided not to use the common sub-directions which are shared by both loss’ and uncertainty’s gradients, instead we used the sub-directions which are left from the loss’ gradient after discarding the common sub-directions as in Fig. 6b. By gradients, here we mean the sign of the gradient vectors.

Figure 7 shows some examples of adversarial samples crafted using different methods mentioned in this study, including our proposed Rectified-BIM attack.

The idea that we have used for our rectified-BIM attack can be easily applied to other loss-based attacks as well. In our proposed attack (Algorithm 2), if we set the number of iterations as 1 and  $\alpha$  equals to  $\epsilon$ , we can simply switch to Rectified-FGSM attack. And, if we use random restart instead of starting from the original point, we can simulate Rectified-

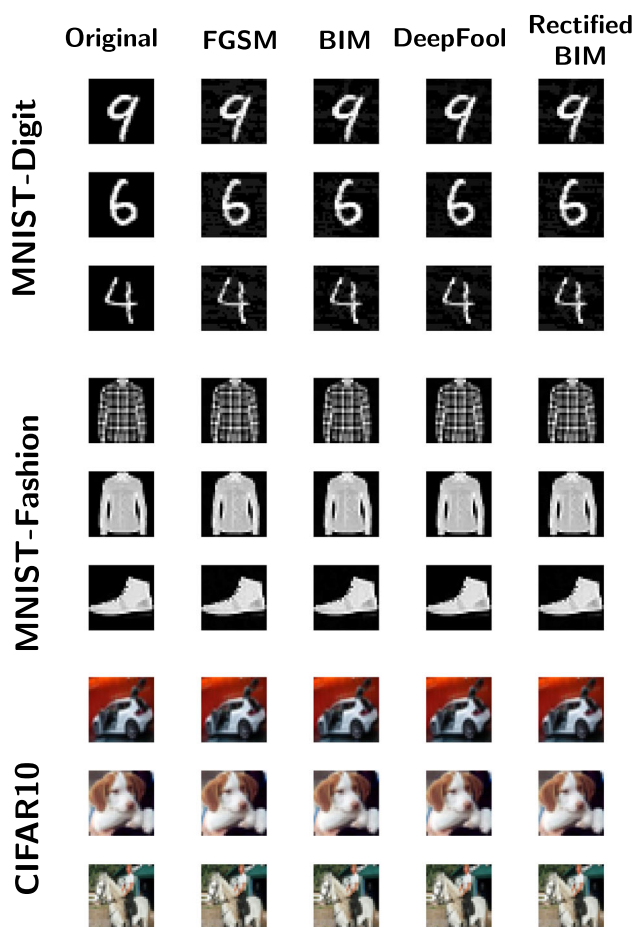


Fig. 7 Some example images from MNIST (Digit), MNIST (Fashion) and CIFAR-10. The original image is shown in the left-most column and adversarial samples crafted based on different methods are on the other columns

PGD attack under  $L_\infty$  norm. Therefore, we shared the results of all possible attack variants together with their original counterparts (BIM, FGSM, PGD) in the experiment section.

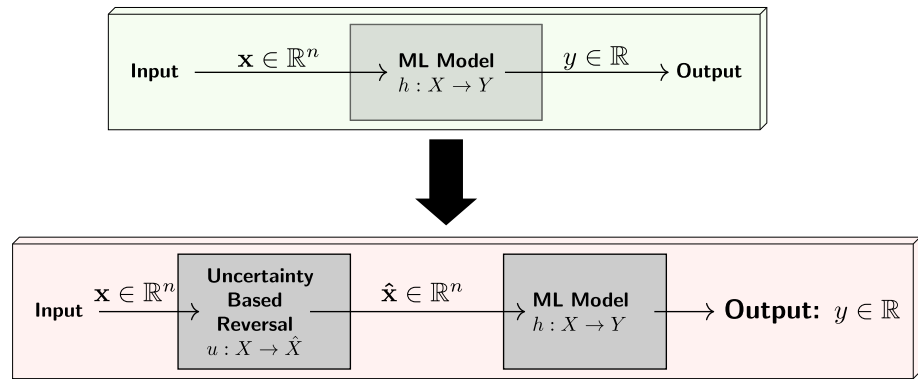
### Uncertainty-based reversal operation

In this section, we first provide the pseudo-code for our uncertainty-based reversal operation as in Algorithm 3. As in the case of our attack algorithm, our proposed reversal method is also designed under  $L_\infty$  norm.

At each iteration of the uncertainty-based reversal algorithm, we calculate  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{\text{pred}}))$  and  $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$ . Then, we simply perturb the input image using the sub-directions of uncertainty’s gradient which are not shared by the loss’ gradient. This idea can be understood better by looking at Fig. 6c.

In a typical production environment where an ML model is used for any classification problem, the input is fed to the model, and the final decision is observed after the input is

**Fig. 8** Deployment options of the ML model—with and without uncertainty-based reversal step



**Algorithm 3:** Algorithm for uncertainty-based reversal operation:  $\mathbf{x}$  is the input image,  $y_{pred}$  is the initial predicted label for  $\mathbf{x}$ ,  $h$  is the hypothesis function learnt during training,  $p$  is the dropout ratio used in dropout layers,  $T$  is the number of MC dropout samples in model training mode,  $N$  is the number of iterations,  $\epsilon$  is the maximum allowed perturbation,  $\alpha$  is the step size,

---

**Input:**  $\mathbf{x} \in \mathbb{R}^m, h, p, T, N, \epsilon, \alpha$   
**Output:**  $\mathbf{x}_{t+1}$

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2  $condition \leftarrow False$ 
3 while  $n < N$  do
4   Compute  $\nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{pred}))$  while  $h$  in evaluation mode
5   Compute  $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$  while  $h$  in training mode
6   if  $\arg \max(h(\mathbf{x}_{t+1})) \neq y_{pred}$  then
7      $condition = True$ 
8     break
9   if  $condition = False$  then
10    Update all elements of  $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$  to 0 where
11     $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T) == \nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{pred}))$ 
12    /* update  $\mathbf{x}$  using below formula */
13     $\mathbf{x}_{(t+1)} = clip_{\mathbf{x}, \epsilon}(\mathbf{x}_t - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)))$ 
14  return  $\mathbf{x}_{t+1}$ 

```

---

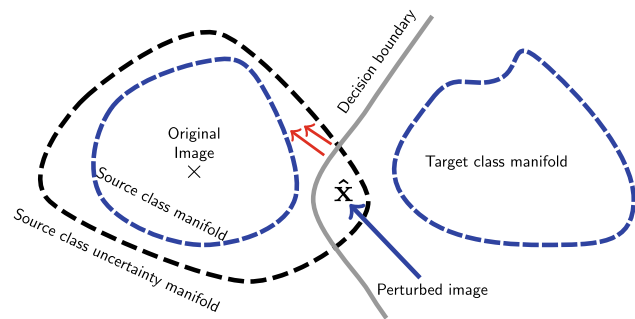
processed and mapped to an output, as shown in the upper example of Fig. 8. During this prediction time, the model does not have any information about the actual label for the input, yet it certainly has an opinion about the predicted label. And this predicted label can be used to quantify the loss. Suppose the prediction of the ML model is correct. In that case, the derivative of the loss against the predicted label gives us an idea about the possible direction by which we can decrease the predicted loss. However, if the initial prediction is wrong, in that case, the model's prediction will be even more erroneous.

This fact is related to the inherent relation of the loss function to the label information. Inevitably, the calculation of loss requires a reference information. One of the important benefits of using uncertainty information is that the quantified uncertainty calculation does not require a reference informa-

tion. Coming back to the ideal case of using an ML model in a production environment, this time, we will be using both the loss and uncertainty's gradient. This time we will only take the sub-directions from the uncertainty's gradient which are not shared by the loss' gradient. Since, for the perturbed images that are pushed away from the decision boundary, the gradient of the loss based on the predicted label will point to the target class data manifold. Therefore, these sub-directions should not be taken into account if we wish to revert the input sample back to its own data manifold. After discarding some portion of the sub-directions, the resulting sub-directions left in the uncertainty's gradient can be safely used to revert the input back to its original data manifold.

Our proposed uncertainty-based reversal algorithm can be implemented as a *preprocessing* module prior to any classification model in production. For an input  $\mathbf{x}$  that is planned to be fed to the ML model, it is first processed within this module as depicted in the bottom part of Fig. 8. This preprocessing aims to carefully perturb the input image in a direction that minimizes the quantified epistemic uncertainty. Then, the resulting reversed image  $\hat{\mathbf{x}}$  will be used as an input to the ML model. The uncertainty-based reversal operation is depicted in Fig. 9. The key point for a successful reverse operation is that the point where the input image resides should not be far away (on the wrong side) from the model's decision boundary. The attack types that we used to show our approach's effectiveness are chosen based on this fact.

It was shown that no matter what kind of defensive approaches are utilized like adversarial training or defensive distillation technique, strong attack types, such as Carlini and Wagner (CW), or Deepfool attacks, can break these defensive methods and successfully fool the ML model [27]. However, when we analyze the softmax scores of the perturbed samples which are crafted using Deepfool attack and CW attack with default setting (confidence parameter set to 0), we see that the resulting confidence level for the wrong class is only slightly larger than the original class. That means the perturbed image is actually not pushed far away from the decision boundary. Consequently, by applying our uncertainty-based reversal operation, one can actually revert the perturbed sample back



**Fig. 9** Reverting perturbed image back to its original class data manifold

to the original class data manifold. We have picked random samples from the MNIST Digit dataset to illustrate this phenomenon, then applied CW and Deepfool attacks on them to craft adversarial samples and finally applied our uncertainty-based reversal operation on these perturbed samples. For each of the original, perturbed and reversed samples, we have also depicted the output softmax scores of the ML model used as in Fig. 10. We observe that although these attacks are successful, the wrong classes' output softmax scores are not large enough in favor of the wrong class, and the difference between the softmax scores of the correct and wrong classes is considerably small. As a result, by applying our uncertainty-based reversal operation on these perturbed samples, we could successfully revert them back to their original class manifold so that the ML model will predict these reversed samples correctly, and therefore will not be fooled. And as a natural consequence of a successful reversal operation, the adversarial inputs can be detected as well. Line 6 of Algorithm 3 checks whether the predicted label for the input has changed from the initial prediction. If this is the case, one can use this as a possible sign of adversarial detection. Therefore, our approach can be used for both detecting adversarial samples and defending against them.

Obviously, a key point that should be considered for any kind of preprocessing operation on the input samples of an ML model is that the preprocessing operation should not negatively affect the “clean data” performance of the model. Any kind of intervention to the ML model's deployment which will decrease the prediction performance higher than an acceptable level, simply can not be tolerated no matter how much robustness it provides. We have performed a test to measure the effect of our uncertainty-based reversal operation on the clean data performance of the model and verified that the accuracy rate does not decrease as presented in the experiments section. The results prove that one can safely use our approach to increase the robustness against malicious attempts to the deployed ML model.

## Adversarial assumptions

In this study, we presume that the attacker's primary motivation is to harvest the desired behavior for an ML model, and the success criteria for the adversary is directly related to any labeling error. In the literature, this type of attack strategy is categorized as *untargeted* attack in which the adversary is counted as successful if, for instance, a dog image is predicted as anything other than a dog. We assumed that the adversary has complete knowledge about the model architecture and model parameters for the attack part as in the case *whitebox* setting. And the adversary uses the gradients obtained from the model to craft adversarial samples. Another important assumption is related to the limitations of the adversary. Obviously, for an attack to be indistinguishable to the human eye, the adversary should be restricted to apply a perturbation with  $L_p$  norm up to some  $\epsilon$ . To make this manipulation quasi-imperceptible to the human eye, the attacker must search for an approximate solution to a complex optimization problem and decides which regions in the input data should be changed. Using any of the available attack methods [22,23,25,60], the attacker tries to reduce the classification performance of the target model as much as possible. Throughout the study, to limit the maximum allowed perturbation for the attacker, we used  $L_\infty$  norm, which is the maximum pixel difference limit between the original and adversarial image.

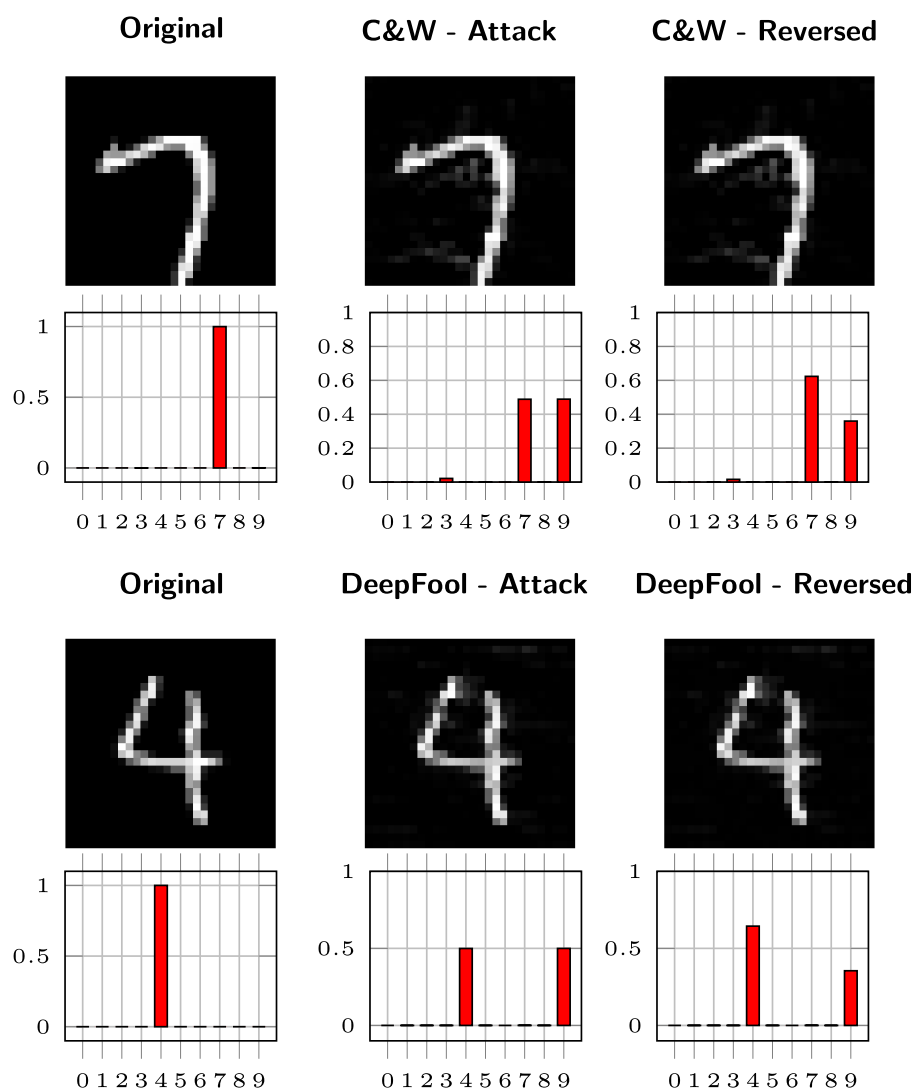
For our proposed defense method, we employed a proactive strategy. We anticipate the potential threat of adversarial activity throughout the deployment life cycle of the ML model regardless of whether an attack attempt exists or not. The same preprocessing operation is being applied to every input that is planned to be fed to the ML model regardless of being an adversarial or benign sample. The amount of perturbation applied during the uncertainty-based reversal step is configurable. If the primary concern is to revert more potentially perturbed samples, the model owner can opt to use a relatively larger  $\epsilon$  value despite the risk of slightly lowering clean accuracy. Our defense method is evaluated in terms of the error rate across a maliciously perturbed version of the chosen test set. This error rate metric is proposed by Goodfellow et al. [22] and is still suggested by Carlini et al. [61].

## Results

### Experimental setup

We trained our CNN models for the MNIST (Digit) [62], MNIST (Fashion) [63] and CIFAR10 [64] datasets, and we achieved accuracy rates of 99.26%, 92.63% and 83.91% respectively. The model architectures are given in Table 1 and the hyperparameters selected in Table 2. For the CIFAR10

**Fig. 10** Effect of adversarial attack and reversal operation on model prediction



dataset we applied a normalization procedure by normalizing all the pixels with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. The adversarial settings that have been used throughout of our experiments is provided in Table 3. Finally, we used  $T = 50$  as the number of MC dropout samples when quantifying epistemic uncertainty.

### Experimental results

We only perturbed the test samples during our experiments on testing the performances of adversarial attack methods, which were already correctly classified by our models in their original states. Obviously, an intruder would have no reason to perturb samples that are already classified wrongly.

The results in Table 5 show that our Rectified-BIM algorithm and other two attack variants (Rectified-FGSM and Rectified-PGD) which leverage both the quantified epistemic uncertainty and the model's loss value result into better performances than their originals. Once we verified the success

of our attack variants on our comparably small models with different epsilon ( $\epsilon$ ) values, we tried to test their performances on larger models. For this purpose, we first trained VGG-19 [65] (with batch normalization) and ResNet (with custom dropout layers) models on CIFAR-10 dataset and achieved accuracy rates of % 85.79 and % 86.24. Then, we have applied all our attack variants and compared the attack success rates with their originals. The results available in Table 6 reveal once again the effectiveness of our proposed attack method. After verifying the efficacy of our approach on various model architectures, we conducted a separate experiment to observe the effect of chosen *number of iterations* parameter on the performance of our proposed iterative attack variant. The results available in Table 7 show that our PGD attack variant outperforms its original counterpart in each attempt (PGD5, PGD10, PGD20).

Unlike known loss-based attacks, our proposed attack variants perturb less number of pixels at each iteration and still can reach a higher degree of adversarial attack success



**Table 1** CNN model architectures

Dataset	Layer type	Layer information
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Max pooling	$2 \times 2$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Max pooling	$2 \times 2$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Dropout	p : 0.2
	Fully connected + ReLU	$1568 \times 100$
	Fully connected + ReLU	$100 \times 10$
MNIST (Fashion)	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Max pooling	$2 \times 2$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Max pooling	$2 \times 2$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.2
	Fully connected + ReLU	$3136 \times 600$
	Fully connected + ReLU	$600 \times 120$
CIFAR10	Fully connected + ReLU	$120 \times 10$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Max pooling (Stride 2)	$2 \times 2$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 128$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 128$
	Max pooling (Stride 2)	$2 \times 2$
	Dropout	p : 0.1
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Max pooling (Stride 2)	$2 \times 2$
	Fully connected + ReLU	$4096 \times 512$
	Dropout	p : 0.5
	Fully connected + ReLU	$512 \times 512$
Dropout	p : 0.5	
Fully connected + ReLU	$512 \times 10$	

**Table 2** CNN model parameters

Parameter	MNIST (Digit)	MNIST (Fashion)	CIFAR10
Optimizer	SGD	SGD	SGD
Learning rate	$0.01 \times (\frac{1}{2}^{\text{epoch}/10})$	$0.01 \times (\frac{1}{2}^{\text{epoch}/10})$	$0.005 \times (\frac{1}{2}^{\text{epoch}/20})$
Momentum	0.9	0.9	0.9
Batch size	64	64	64
Dropout ratio	0.2	0.2	0.5
Number of epochs	30	50	50

**Table 3** Adversarial settings of our experiments:  $\alpha$ ,  $i$  respectively represent the step-size and the number of attack steps for a perturbation budget  $\epsilon$ ,  $r$  denotes the number of restarts

Attack	Parameters	$l_p$ norm
FGSM	$i = 1$	$l_\infty$
BIM	$\alpha = \epsilon \cdot 0.2, i = 10$	$l_\infty$
PGD	$\alpha = \epsilon \cdot 0.2, i = 10, r = 5$	$l_\infty$

**Table 4** Parameters that are used in uncertainty reversal operation:  $\alpha$ ,  $i$  respectively represent the step-size and the number of reversal steps for a perturbation budget  $\epsilon$ 

Dataset	Parameters	$l_p$ norm
MNIST digit	$\epsilon = 0.02, \alpha = \epsilon \cdot 0.2, i = 10$	$l_\infty$
MNIST fashion	$\epsilon = 0.001, \alpha = \epsilon \cdot 0.2, i = 10$	$l_\infty$
CIFAR10	$\epsilon = 0.1/255, \alpha = \epsilon \cdot 0.2, i = 10$	$l_\infty$

rate. To verify this, we have conducted an additional experiment to compare the resulting perturbation amounts of our attack variants with their originals (FGSM, BIM, PGD). We have used a batch of input from each dataset and computed

the resulting perturbation amounts based on both  $L_1$  and  $L_2$  norms. The results that are shown in Table 8 support our claim. We know that any trained ML model is not the perfect predictor and is just an approximation to the oracle function. The model itself has an inevitable inherent approximation uncertainty which sometimes induce to sub-optimal solutions. Consequently, any method which only relies on the trained model will result in less effective performance. By double-checking the sub-directions pointed by the derivative of the model's loss with the ones available in the quantified epistemic uncertainty derivative, we could discard the sub-directions, which are suspected to be unreliable. The attack success rates are inline with this fact.

As a last experiment for the attack part, we have measured the time spent by each attack method for a batch of input of size 64 from the MNIST (Digit) Dataset. The results available in Table 9 show the measured execution time of our attack variants together with their originals (FGSM, BIM, PGD). As expected, the execution time of our attack variants is longer than their originals due to additional uncertainty quantification steps. However, this can be tolerated thanks to the higher success rate and smaller perturbation need of our attack variants.

**Table 5** Attack success rates on different datasets

	MNIST (Digit)		MNIST (Fashion)		CIFAR10	
	$\epsilon = 0.12$ (%)	$\epsilon = 0.16$ (%)	$\epsilon = 0.04$ (%)	$\epsilon = 0.06$ (%)	$\epsilon = 2/255$ (%)	$\epsilon = 3/255$ (%)
FGSM	23.76	37.79	53.86	61.56	41.71	45.98
Rectified-FGSM	24.64	40.33	55.86	66.62	45.11	52.88
BIM	53.40	81.40	76.24	89.65	59.42	70.21
Rectified-BIM	56.93	85.97	80.86	96.09	69.14	83.13
PGD	54.26	84.01	76.69	91.23	59.38	70.62
Rectified-PGD	55.41	86.58	79.71	95.22	67.14	81.63
Deepfool	40.05	72.10	75.67	92.38	64.09	81.81

**Table 6** Attack success rates under  $l_\infty$  norm against VGG-19 and ResNet models

CIFAR10 dataset	VGG19bn (%)	ResNet (%)
FGSM ( $\epsilon = 3/255$ )	56.78	74.33
Rectified-FGSM ( $\epsilon = 3/255$ )	59.18	78.11
BIM ( $\epsilon = 3/255, \alpha = \epsilon \cdot 0.4, i = 5$ )	73.45	94.99
Rectified-BIM ( $\epsilon = 3/255, \alpha = \epsilon \cdot 0.4, i = 5$ )	84.67	96.42
PGD ( $\epsilon = 3/255, \alpha = \epsilon \cdot 0.4, i, r = 5$ )	74.95	95.06
Rectified-PGD ( $\epsilon = 3/255, \alpha = \epsilon \cdot 0.4, i, r = 5$ )	83.97	96.17

**Table 7** Results under PGD attack with various number of iterations

VGG19bn on CIFAR10 dataset	PGD (%)	Rectified-PGD (%)
Number of iterations = 5, $\epsilon = 3/255$	73.25	83.01
Number of iterations = 10, $\epsilon = 3/255$	82.13	91.30
Number of iterations = 20, $\epsilon = 3/255$	83.60	92.24

**Table 8** Mean perturbation amount comparison

	MNIST Digit ( $\epsilon = 0.16$ )		MNIST Fashion ( $\epsilon = 0.06$ )		CIFAR10 ( $\epsilon = 3/255$ )	
	$L_2$ norm	$L_1$ norm	$L_2$ norm	$L_1$ norm	$L_2$ norm	$L_1$ norm
FGSM	3.36	71.57	1.44	35.19	0.64	35.90
Rectified-FGSM	3.06	59.53	1.29	28.56	0.56	27.48
BIM	2.75	53.41	1.21	27.40	0.55	28.27
Rectified-BIM	2.69	52.09	1.16	25.99	0.54	27.70
PGD	2.79	54.79	1.22	27.96	0.55	28.39
Rectified-PGD	2.74	53.70	1.19	26.92	0.54	28.02

**Table 9** Execution time comparison (in seconds)

FGSM	0.0104
Rectified-FGSM	0.4613
BIM	0.0444
Rectified-BIM	6.4161
PGD	0.4695
Rectified-PGD	31.3377

For the defense part, the results in Table 10 show that our proposed uncertainty-based reversal method does not decrease the classification performance of the deployed models on clean data. While some minor portion of data samples were wrongly classified even if they were previously classified correctly, there is an almost similar amount of samples that happened to be predicted correctly after the reversal process. Therefore, we can conclude that on average, there is no negative effect of deploying our uncertainty-based reversal method as a preprocessing module prior to feeding any input to an ML model in production.

We then tested our proposed defense technique against two powerful attack types, namely Deepfool attack and Carlini and Wagner attack (with confidence = 0). For the implementations of these attacks, we used a Python toolbox called Foolbox [66]. The results in Table 11 prove our uncertainty-based reversal method's effectiveness. For the MNIST datasets, our proposed defense method provides per-

fect robustness and almost totally revert all the perturbed samples crafted by these strong attacks. And for the CIFAR10 dataset, we achieved reversal success rates of around % 96 for Deepfool attack and almost % 98 for CW attack. The settings used in the uncertainty-based reversal operation are given in Table 4. Thanks to our defense method, we could lower the final attack success rates of these attacks to very low levels as shown in Table 12.

## Discussions and further results

We have tested and verified our approach's effectiveness in three different datasets, which are heavily used by the adversarial research community. Experimental results show that our proposed methods generalize well across datasets. Our defense method is not concerned with the internal dynamics of the attack. Instead, it is concerned with the eventual result of the attack, which is the perturbed sample. The underlying factor for our approach's success can be understood better by analyzing the resulting final probability scores of the perturbed samples. Since the gap between the predicted probability scores of original and target classes is not high for Deepfool attack and CW attack (when confidence parameter is set to 0), we could successfully revert almost all perturbed images with a reasonably small perturbation because the perturbed images are not pushed far away from the decision boundary of the model for these attacks. Other known attacks

**Table 10** Performance of uncertainty-based reversal operation on clean data

	Number of successful reversal	Number of unsuccessful reversal
MNIST (Digit)		
On corrects	9901	25
On wrongs	14	60
MNIST (Fashion)		
On corrects	9185	78
On wrongs	62	675
CIFAR10		
On corrects	8080	241
On wrongs	214	1465

**Table 11** Performance of uncertainty-based reversal operation on different attack types

	Successful reversal	Unsuccessful reversal	Reversal success rate (%)
MNIST (Digit)			
Deepfool ( $L_\infty$ , $\epsilon = 0.1$ )	2505	0	100
Deepfool ( $L_2$ , $\epsilon = 1.35$ )	2935	27	99.09
CW ( $L_2$ , $\epsilon = 1.35$ )	5161	0	100
MNIST (Fashion)			
Deepfool ( $L_\infty$ , $\epsilon = 0.05$ )	7940	19	99.76
Deepfool ( $L_2$ , $\epsilon = 0.67$ )	7268	25	99.66
CW ( $L_2$ , $\epsilon = 0.67$ )	7898	19	99.76
CIFAR10			
Deepfool ( $L_\infty$ , $\epsilon = 2/255$ )	5078	255	95.22
Deepfool ( $L_2$ , $\epsilon = 53/255$ )	4081	117	97.21
CW ( $L_2$ , $\epsilon = 53/255$ )	4904	102	97.96

**Table 12** Effect of uncertainty-based reversal on attack success rates

	Attack success rates	
	Without reversal (%)	With (%)
MNIST (Digit)		
Deepfool ( $L_\infty$ , $\epsilon = 0.1$ )	25.24	0.00
Deepfool ( $L_2$ , $\epsilon = 1.35$ )	29.84	0.27
CW ( $L_2$ , $\epsilon = 1.35$ )	51.99	0.00
MNIST (Fashion)		
Deepfool ( $L_\infty$ , $\epsilon = 0.05$ )	85.92	0.21
Deepfool ( $L_2$ , $\epsilon = 0.67$ )	78.73	0.27
CW ( $L_2$ , $\epsilon = 0.67$ )	85.47	0.21
CIFAR10		
Deepfool ( $L_\infty$ , $\epsilon = 2/255$ )	63.56	3.04
Deepfool ( $L_2$ , $\epsilon = 53/255$ )	50.03	1.39
CW ( $L_2$ , $\epsilon = 53/255$ )	59.66	1.22

like FGSM, BIM or PGD might not result in such a small gap in the final softmax scores. This is also valid if one applies CW attack with a high confidence value as a parameter as well. Thus, our reversal process's success rate will not be so high as in the case of the previously mentioned attack implementations. However, we have empirically verified tools like adversarial training to defend against those attacks (FGSM, BIM, PGD), and the success rate of those attacks can be lowered to an acceptable level (Table 15).

To verify this, we have used two other models, which are naturally and adversarially trained for the MNIST Digit classification task. We used a slightly different architecture this time and applied the dropout in the first convolutional layer as in Table 13. We then checked the attack success rates of FGSM, BIM, Deepfool and CW attacks against these two trained models. As expected, the success rates for FGSM and BIM were significantly lowered via adversarial training. Additional implementation of our uncertainty-based reversal

approach decreased the attack success rates even further, as shown in Table 14.

Regarding the most powerful attack in our experiments, our defense method provides excellent robustness if the attacker applies CW attack with confidence set to 0. The attacker can try to avoid our defense using a high confidence value as a parameter. However, trying to craft high confident adversaries will eventually decrease the attack success rate. And if our method is used in front of an adversarially trained model, the CW attack's success rate will decrease even further.

Finally, when we check if our uncertainty-based reversal operation negatively affect the model's performance for legitimate inputs as shown in Table 15, we once again confirm that the accuracy of the model does not decrease considerably at all. Hence, our approach can be safely used in production when there is no security threat. Suppose that the model owner is worried about any suspicious activity and thinks that



**Table 13** CNN model architecture for adversarially trained MNIST model

Dataset	Layer type	Layer information
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Max pooling	$2 \times 2$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Max pooling	$2 \times 2$
	Fully connected + ReLU	$1568 \times 100$
Fully connected + ReLU	$100 \times 10$	

**Table 14** Effect of adversarial training and uncertainty-based reversal on attack success rates

	Attack success rates	
	Without reversal	With reversal
Normal model		
FGSM ( $\epsilon = 0.1$ )	% 11.82	% 9.01
BIM ( $L_\infty, \epsilon = 0.1$ )	%32.02	% 22.13
Deepfool ( $L_\infty, \epsilon = 0.1$ )	% 17.28	% 0.02
Deepfool ( $L_2, \epsilon = 1.35$ )	% 22.79	% 0.30
CW ( $L_2, \epsilon = 1.35, \text{confidence} = 0$ )	% 50.42	% 0
CW ( $L_2, \epsilon = 1.35, \text{confidence} = 10$ )	% 40.42	% 30.38
Robust model		
FGSM (eps = 0.1)	% 1.32	% 1.02
BIM ( $L_\infty, \epsilon = 0.1$ )	% 1.43	% 1.11
Deepfool ( $L_\infty, \epsilon = 0.1$ )	% 1.34	% 0.01
Deepfool ( $L_2, \epsilon = 1.35$ )	% 4.14	% 0.20
CW ( $L_2, \epsilon = 1.35, \text{confidence} = 0$ )	% 7.78	% 0.03
CW ( $L_2, \epsilon = 1.35, \text{confidence} = 10$ )	% 4.82	% 4.04

**Table 15** Effect of uncertainty-based reversal on clean performance for normal and adversarially trained model

	# of successful reversal	# of unsuccessful reversal
Normal model		
On corrects	9916	17
On wrongs	19	48
Robust model		
On corrects	9921	13
On wrongs	4	62

the model is under attack. In that case, he/she can tune the  $\epsilon$  parameter for the reversal operation to increase the power of defense at the cost of slightly lowering clean data performance.

## Conclusion

This study proposed new attack and defense ideas based on the combined usage of the model's epistemic uncertainty

and the model's loss function. We experimentally showed that our rectified attack algorithms outperform some of the well known adversarial attack methods which rely solely on the model loss function. On the adversarial defense side, we showed that our uncertainty-based reversal operation could be used as a preprocessing module prior to feeding any input to the model in production. Our approach provides a very high degree of adversarial robustness without compromising clean data performance.

As future work, we plan to investigate the possible usage of our uncertainty-based reversal operation to build dynamic models that train themselves during production by updating model parameters after the detection of any adversarial input at the reversal process. We believe that enabling such a self-training in the production life cycle of a model will help to improve adversarial robustness as in the case of any adversarial training approach. Another advantage of this implementation is that model's owner will not have to be involved in crafting those adversarial samples beforehand and turn a negative situation of being under attack into a benefit. Another future study is to use the uncertainty-based reversal approach to craft Universal Robustness Perturbation instead of the previous studies of Universal Adversarial Perturbation. We encourage the research community to investigate further opportunities which our uncertainty-based reversal approach presents.

## Declarations

**Conflict of interest** On behalf of all authors, the corresponding author states that they do not have any conflicts of interest to declare.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. [arXiv:1512.03385](https://arxiv.org/abs/1512.03385)
- Goodfellow IJ, Bulatov Y, Ibarz J, Arnaud S, Shet V (2014) Multi-digit number recognition from street view imagery using deep convolutional neural networks. [arXiv:1312.6082](https://arxiv.org/abs/1312.6082)
- Chouard T (2016) The go files: Ai computer wraps up 4-1 victory against human champion. *Nature*. <https://doi.org/10.1038/nature.2016.19575>
- Shen L, Margolies LR, Rothstein JH, Fluder E, McBride R, Sieh W (2019) Deep learning to improve breast cancer detection on screening mammography. *Sci Rep* 9(1):12495. <https://doi.org/10.1038/s41598-019-48995-4>
- Causey JL, Zhang J, Ma S, Jiang B, Qualls JA, Politte DG, Prior F, Zhang S, Huang X (2018) Highly accurate model for prediction of lung nodule malignancy with ct scans. *Sci Rep* 8(1):9286. <https://doi.org/10.1038/s41598-018-27569-w>
- Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) *Advances in neural information processing systems*, vol 28. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf>. Accessed 20 Dec 2021
- Justesen N, Bontrager P, Togelius J, Risi S (2019) Deep learning for video game playing (2019). [arXiv:1708.07902](https://arxiv.org/abs/1708.07902)
- Bahdanau D, Cho K, Bengio Y (2016) Neural machine translation by jointly learning to align and translate. [arXiv:1409.0473](https://arxiv.org/abs/1409.0473)
- Luong M-T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. [arXiv:1508.04025](https://arxiv.org/abs/1508.04025)
- Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. [arXiv:1312.6199](https://arxiv.org/abs/1312.6199)
- Sato M, Suzuki J, Shindo H, Matsumoto Y (2018) Interpretable adversarial perturbation in input embedding space for text. [arXiv:1805.02917](https://arxiv.org/abs/1805.02917)
- Carlini N, Wagner D (2018) Audio adversarial examples: targeted attacks on speech-to-text. [arXiv:1801.01944](https://arxiv.org/abs/1801.01944)
- Finlayson SG, Chung HW, Kohane IS, Beam AL (2019) Adversarial attacks against medical deep learning systems. [arXiv:1804.05296](https://arxiv.org/abs/1804.05296)
- Sitawarin C, Bhagoji AN, Mosenia A, Chiang M, Mittal P (2018) Darts: deceiving autonomous cars with toxic signs. [arXiv:1802.06430](https://arxiv.org/abs/1802.06430)
- Morgulis N, Kreines A, Mendelowitz S, Weisglass Y (2019) Fooling a real car with adversarial traffic signs. [arXiv:1907.00374](https://arxiv.org/abs/1907.00374)
- Tuna OF, Catak FO, Eskil MT (2022) Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples. *Multimedia Tools App* year. <https://doi.org/10.1007/s11042-022-12132-7> ISBN: 1573–7721
- Huang X, Kroening D, Ruan W, Sharp J, Sun Y, Thamo E, Wu M, Yi X (2020) A survey of safety and trustworthiness of deep neural networks: verification, testing, adversarial attack and defence, and interpretability. *Comput Sci Rev* 37:100270. <https://doi.org/10.1016/j.cosrev.2020.100270>
- Catak FO, Sivaslioglu S, Sahinbas K (2020) A generative model based adversarial security of deep learning and linear classifier models. [arXiv:2010.08546](https://arxiv.org/abs/2010.08546)
- Qayyum A, Usama M, Qadir J, Al-Fuqaha A (2020) Securing connected autonomous vehicles: challenges posed by adversarial machine learning and the way forward. *IEEE Commun Surv Tutor* 22(2):998–1026. <https://doi.org/10.1109/COMST.2020.2975048>
- Sadeghi K, Banerjee A, Gupta SKS (2020) A system-driven taxonomy of attacks and defenses in adversarial machine learning. *IEEE Trans Emerg Top Comput Intell* 4(4):450–467. <https://doi.org/10.1109/TETCI.2020.2968933>
- Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. [arXiv:1412.6572](https://arxiv.org/abs/1412.6572)
- Kurakin A, Goodfellow I, Bengio S (2017) Adversarial examples in the physical world. [arXiv:1607.02533](https://arxiv.org/abs/1607.02533)
- Kurakin A, Goodfellow IJ, Bengio S (2017) Adversarial machine learning at scale. *CoRR* abs/1611.01236. [arXiv:1611.01236](https://arxiv.org/abs/1611.01236)
- Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2019) Towards deep learning models resistant to adversarial attacks. [arXiv:1706.06083](https://arxiv.org/abs/1706.06083)
- Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning. [arXiv:1602.02697](https://arxiv.org/abs/1602.02697)
- Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. [arXiv:1608.04644](https://arxiv.org/abs/1608.04644)

28. Moosavi-Dezfooli S-M, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks. [arXiv:1511.04599](https://arxiv.org/abs/1511.04599)
29. Liu H, Ji R, Li J, Zhang B, Gao Y, Wu Y, Huang F (2019) Universal adversarial perturbation via prior driven uncertainty approximation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 2941–2949. <https://doi.org/10.1109/ICCV.2019.00303>
30. Qu X, Sun Z, Ong YS, Gupta A, Wei P (2020) Minimalistic attacks: How little it takes to fool deep reinforcement learning policies. *IEEE Trans Cogn Dev Syst*. <https://doi.org/10.1109/TCDS.2020.2974509>
31. Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, vol 70 Proceedings of machine learning research, PMLR, pp 1321–1330. <https://proceedings.mlr.press/v70/guo17a.html>. Accessed 20 Dec 2021
32. Gawlikowski J, Tassi CRN, Ali M, Lee J, Humt M, Feng J, Kruspe A, Triebel R, Jung P, Roscher R, Shahzad M, Yang W, Bamler R, Zhu XX (2021) A survey of uncertainty in deep neural networks. [arXiv:2107.03342](https://arxiv.org/abs/2107.03342)
33. Papernot N, McDaniel P, Wu X, Jha S, Swami A (2016) Distillation as a defense to adversarial perturbations against deep neural networks. [arXiv:1511.04508](https://arxiv.org/abs/1511.04508)
34. Xie C, Wu Y, van der Maaten L, Yuille A, He K (2019) Feature denoising for improving adversarial robustness. [arXiv:1812.03411](https://arxiv.org/abs/1812.03411)
35. Carlini N, Katz G, Barrett C, Dill DL (2018) Ground-truth adversarial examples. <https://openreview.net/forum?id=Hki-ZIbA->. Accessed 20 Dec 2021
36. Meng D, Chen H (2017) Magnet: a two-pronged defense against adversarial examples. [arXiv:1705.09064](https://arxiv.org/abs/1705.09064)
37. Carlini N, Wagner D (2017) Magnet and “efficient defenses against adversarial attacks” are not robust to adversarial examples. [arXiv:1711.08478](https://arxiv.org/abs/1711.08478)
38. Liao F, Liang M, Dong Y, Pang T, Hu X, Zhu J (2018) Defense against adversarial attacks using high-level representation guided denoiser. [arXiv:1712.02976](https://arxiv.org/abs/1712.02976)
39. Shen S, Jin G, Gao K, Zhang Y (2017) Ape-gan: adversarial perturbation elimination with gan. [arXiv:1707.05474](https://arxiv.org/abs/1707.05474)
40. Raghunathan A, Steinhardt J, Liang P (2020) Certified defenses against adversarial examples. [arXiv:1801.09344](https://arxiv.org/abs/1801.09344)
41. Laves M-H, Ihler S, Ortmaier T (2019) Uncertainty quantification in computer-aided diagnosis: Make your model say “i don’t know” for ambiguous cases. [arXiv:1908.00792](https://arxiv.org/abs/1908.00792)
42. Tuna OF, Catak FO, Eskil MT (2020) Closeness and uncertainty aware adversarial examples detection in adversarial machine learning. [arXiv:2012.06390](https://arxiv.org/abs/2012.06390)
43. Hüllermeier E, Waegeman W (2020) Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. [arXiv:1910.09457](https://arxiv.org/abs/1910.09457)
44. An D, Liu J, Zhang M, Chen X, Chen M, Sun H (2020) Uncertainty modeling and runtime verification for autonomous vehicles driving control: a machine learning-based approach. *J Syst Softw* 167:110617. <https://doi.org/10.1016/j.jss.2020.110617>
45. Zheng R, Zhang S, Liu L, Luo Y, Sun M (2021) Uncertainty in bayesian deep label distribution learning. *Appl Soft Comput* 101:107046. <https://doi.org/10.1016/j.asoc.2020.107046>
46. Antonelli F, Cortellessa V, Gribaudo M, Pincioli R, Trivedi KS, Trubiani C (2020) Analytical modeling of performance indices under epistemic uncertainty applied to cloud computing systems. *Future Gener Comput Syst* 102:746–761. <https://doi.org/10.1016/j.future.2019.09.006>
47. Zhou D-X (2018) Universality of deep convolutional neural networks. [arXiv:1805.10769](https://arxiv.org/abs/1805.10769)
48. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Math Control Signals Syst (MCSS)* 2(4):303–314. <https://doi.org/10.1007/BF02551274>
49. Loquercio A, Segu M, Scaramuzza D (2020) A general framework for uncertainty estimation in deep learning. *IEEE Robot Autom Lett* 5(2):3153–3160. <https://doi.org/10.1109/LRA.2020.2974682>
50. Gurevich P, Stuke H (2019) Pairing an arbitrary regressor with an artificial neural network estimating aleatoric uncertainty. *Neurocomputing* 350:291–306. <https://doi.org/10.1016/j.neucom.2019.03.031>
51. Senge R, Bösner S, Dembczyński K, Haasenritter J, Hirsch O, Donner-Banzhoff N, Hüllermeier E (2014) Reliable classification: learning classifiers that distinguish aleatoric and epistemic uncertainty. *Inf Sci* 255:16–29. <https://doi.org/10.1016/j.ins.2013.07.030>
52. Hinton GE, Neal R (1995) Bayesian Learning for Neural Networks. University of Toronto, CAN ISBN: 0612026760. <https://dl.acm.org/doi/book/10.5555/922680>
53. Graves A (2011) Practical variational inference for neural networks. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ (eds) Advances in neural information processing systems, vol 24. Curran Associates, Inc., pp 2348–2356. <https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf>. Accessed 20 Dec 2021
54. Paisley J, Blei D, Jordan M (2012) Variational bayesian inference with stochastic search. [arXiv:1206.6430](https://arxiv.org/abs/1206.6430)
55. Hoffman M, Blei DM, Wang C, Paisley J (2013) Stochastic variational inference. [arXiv:1206.7051](https://arxiv.org/abs/1206.7051)
56. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. [arXiv:1505.05424](https://arxiv.org/abs/1505.05424)
57. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. [arXiv:1612.01474](https://arxiv.org/abs/1612.01474)
58. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: representing model uncertainty in deep learning. [arXiv:1506.02142](https://arxiv.org/abs/1506.02142)
59. Combalia M, Hueto F, Puig S, Malveyh J, Vilaplana V (2020) Uncertainty estimation in deep neural networks for dermoscopic image classification. In: 2020 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), pp 3211–3220. <https://doi.org/10.1109/CVPRW50498.2020.00380>
60. Aladag M, Catak FO, Gul E (2019) Preventing data poisoning attacks by using generative models, in: 2019 1st international informatics and software engineering conference (UBMYK), pp 1–5. <https://doi.org/10.1109/UBMYK48245.2019.8965459>
61. Carlini N, Athalye A, Papernot N, Brendel W, Rauber J, Tsipras D, Goodfellow I, Madry A, Kurakin A (2019) On evaluating adversarial robustness. [arXiv:1902.06705](https://arxiv.org/abs/1902.06705)
62. LeCun Y, Cortes C MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>. Accessed 20 Dec 2021
63. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. [arXiv:1708.07747](https://arxiv.org/abs/1708.07747)
64. Krizhevsky A, Nair V, Hinton G Cifar-10 (canadian institute for advanced research). <http://www.cs.toronto.edu/~kriz/cifar.html>. Accessed 20 Dec 2021
65. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
66. Rauber J, Brendel W, Bethge M (2018) Foolbox: a python toolbox to benchmark the robustness of machine learning models. [arXiv:1707.04131](https://arxiv.org/abs/1707.04131)