



Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples

Omer Faruk Tuna¹ · Ferhat Ozgur Catak² · M. Taner Eskil¹

Received: 14 May 2021 / Revised: 10 August 2021 / Accepted: 3 January 2022 /

Published online: 18 February 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Deep neural network (DNN) architectures are considered to be robust to random perturbations. Nevertheless, it was shown that they could be severely vulnerable to slight but carefully crafted perturbations of the input, termed as adversarial samples. In recent years, numerous studies have been conducted in this new area called “Adversarial Machine Learning” to devise new adversarial attacks and to defend against these attacks with more robust DNN architectures. However, most of the current research has concentrated on utilising model loss function to craft adversarial examples or to create robust models. This study explores the usage of quantified epistemic uncertainty obtained from Monte-Carlo Dropout Sampling for adversarial attack purposes by which we perturb the input to the shifted-domain regions where the model has not been trained on. We proposed new attack ideas by exploiting the difficulty of the target model to discriminate between samples drawn from original and shifted versions of the training data distribution by utilizing epistemic uncertainty of the model. Our results show that our proposed hybrid attack approach increases the attack success rates from 82.59% to 85.14%, 82.96% to 90.13% and 89.44% to 91.06% on MNIST Digit, MNIST Fashion and CIFAR-10 datasets, respectively.

Keywords Multimedia security · Uncertainty · Adversarial machine learning · Deep learning · Loss maximization

1 Introduction

In recent years, deep learning models began to exceed human-level performances. In 2015, a deep learning model called ResNet [23] beat the human performance in ImageNet Large

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

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

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

¹ Isik University Istanbul, Turkey

² Ferhat Ozgur Catak, University of Stavanger Fornebu, Norway

Scale Visual Recognition Challenge (ILSVRC) and the record was hit by more advanced architectures later on. Goodfellow et al. [18] proposed a system that outperforms human operators in the task of reading address information from Google Street View imagery or solving CAPTCHAS. In the game playing domain, an AI software named AlphaGo defeated the world Go champion in 2016 [11]. Today, we observe that many advanced systems built upon deep learning models offer a very high degree of success in different domains. As a result of this success, DNN models are being used in various fields, ranging from medical diagnosis and autonomous vehicles to game playing and machine translation. However, during the rise of the DNN models, the researchers' main focus was to build more accurate models, and nearly no attention has been paid to the reliability and robustness of these models. Deep learning models indeed require a more elaborate evaluation since these models do not provide uncertainty estimates and are mostly prone to over-confidence or under-confidence predictions [5, 21]. Besides, they have some intrinsic vulnerabilities that let intruders to easily exploit them [50].

Numerous researchers have been striving to comprehend and quantify the uncertainty in DNNs in an attempt to make more reliable decisions. As a result, many research fields, such as robotics and medical diagnosis, have begun to rely on uncertainty-based reasoning rather than a most likely point estimate. For instance, in scenarios like lesion identification [43] and glioma segmentation [13], researchers have employed predictive uncertainty estimates in evaluating medical images. And recently, Ghoshal et al. [17] employed uncertainty quantification for Coronavirus (COVID-19) detection task in X-ray images. Using uncertainty-based reasoning, the authors significantly improved the diagnostic performance of the human-machine alliance using a COVID-19 imaging dataset.

Similar to the efforts for improving the reliability of model predictions, we have witnessed numerous studies aiming to improve the robustness of DNN models. The history of these efforts goes back to the end of 2013, when the researchers have discovered that existing deep neural networks are vulnerable to attacks. Szegedy et al. [53] first noticed the presence of adversarial examples in the context of image classification. The authors have shown that it is possible to perturb an image by a small amount and change how it is classified. Minimal and nearly imperceptible perturbations of the data samples are sufficient to fool the most advanced classifiers into incorrect classification. As in the case of the steganography technique, where researchers aim to efficiently hide secret data within a digital content [35–37], the applied perturbation for an adversarial sample should be quasi-imperceptible.

Adversarial machine learning attacks are based on perturbation of the input instances in a direction that maximizes the chance of wrong decision making and results in false predictions. These attacks can lead to a loss of the model's prediction performance as the algorithm cannot correctly predict the input instances. 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 can lead to an incorrect diagnosis of a disease. Consequently, it can cause severe harm to the patient's health and also damage the healthcare economy [14]. Another example is autonomous navigation; automobiles that use ML to drive through traffic without human intervention while avoiding accidents. A wrong decision through an adversarial attack could cause a fatal accident [42, 52]. Hence, defending against adversarial attempts and increasing the robustness of ML models without compromising their accuracy is of crucial importance. Assuming that these ML models will be used in critical tasks, we should focus our efforts not only on the performance of the models, but also on the security of these systems.

Adversarial ML is a burgeoning area of research, and scientists around the world are struggling to develop new attack algorithms, which in turn will help to develop more robust architectures resistant to malicious attack threats. In this study, we focus on adversarial attack strategies based on epistemic uncertainty maximization instead of traditional loss maximization based attacks. By looking at the problem from a different perspective and proposing alternative ways of crafting adversarial samples, we hope to contribute to future studies that aim to develop more effective and uncertainty-aware defense approaches. The common approach of adversarial machine learning attacks is to create craftily-designed inputs based on model loss maximization. Unlike the previous researches in literature, we follow a slightly modified strategy to craft adversarial samples by exploiting the vulnerability of the model using its quantified epistemic uncertainty. We show that perturbing the input image in a direction that maximizes the model's uncertainty amplifies model loss and results in wrong predictions. The new approach combines the strengths of adversarial approaches to produce more destructive attacks by uncertainty and loss maximization. We have released our source code on GitHub.¹

To sum up; our main contributions with this paper are:

- We utilize a new metric (epistemic uncertainty of the model) which can be exploited to craft adversarial examples.
- We show that the performance of pure uncertainty based attacks is indeed as high as the attacks based on the model loss.
- We demonstrate that crafting adversarial examples using both the model loss and uncertainty yields better performance in adversarial attacks.
- We empirically show that the maximum value of quantified uncertainty for a sample can be located beyond the decision boundary of the model, where the loss is at its maximum value.

This study is organized as follows. Section 2 introduces some of the known attack types in the literature. In Section 3, we will introduce the concept of uncertainty together with the main types and discuss how we can quantify epistemic uncertainty. Section 4 will give the details of our approach. We will present our experimental results in Section 5 and conclude our work in Section 6.

2 Related work

Since the discovery of DNN's vulnerability to adversarial attacks [53], a vast amount of research has been conducted in both devising new adversarial attacks and defending against these attacks with more robust DNN models [9, 26, 47, 48].

Deep learning models contain many vulnerabilities and weaknesses, making them difficult to defend against adversarial attacks. For instance, they are often sensitive to small changes in the input data, resulting in unexpected results in the final output of the model. Figure 1 shows how an adversary would exploit such a vulnerability and manipulate the model through a carefully crafted perturbation of the input data. The malicious input is produced by slightly perturbing the original image so that a “West Highland White Terrier (Dog)” is misclassified as “Paper Towel” with very high confidence.

¹<https://github.com/omerfaruktuna/exploiting-epistemic-uncertainty>

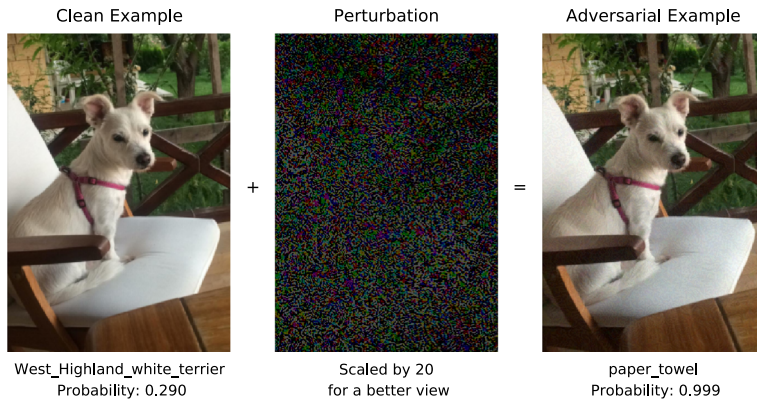


Fig. 1 An adversarial example

Traditionally, attack strategies are mainly based on perturbing the input instance to maximise the model's loss. A great deal of adversarial attack algorithms have been proposed in recent years. In this section, we will briefly describe some of the well-known adversarial attack algorithms.

2.1 Fast-gradient sign method

This method, also known as FGSM [19], is one of the earliest and most popular adversarial attacks to date. FGSM utilizes the gradient of the model's loss function to adjudge in which direction the pixel values of the source image should be altered to minimize the model's loss function. Then it changes all pixels simultaneously in the opposite direction to maximize the loss. For a model with classification loss function described as $L(\theta, \mathbf{x}, y)$ where θ represents model parameters, \mathbf{x} is the input to the model (sample input image in our case), y_{true} is the label of our input, we can generate adversarial samples using below formula:

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

One last key point about FGSM is that it is designed to be fast but not optimal, i.e. it is not intended to devise the minimum required adversarial perturbation. Besides, the success ratio of this method is relatively low in small ϵ values compared to other attack types.

2.2 Iterative gradient sign method

Kurakin et al. [30] proposed a small but effective improvement to the FGSM. In their approach, rather than taking only one step of size ϵ in the direction of the gradient sign, they take several but smaller steps α , and use the given ϵ value to clip the result. This attack type is referred to as Basic Iterative Method (BIM), and it is just FGSM applied to an input image iteratively. Generating perturbed images under L_{inf} norm for BIM attack is given by (2).

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

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_{true} is the actual label for input sample, ϵ is a tunable parameter, limiting maximum level of perturbation in given l_{inf} norm, and α is the step size.

The success ratio of BIM attack is higher than the FGSM [31]. This is mainly due to the fact that the attacker can decide how far past the decision boundary an adversarial sample will be pushed by simply adjusting the ϵ parameter.

We 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 to the attack till the end of provided number of iterations so that we push the input beyond the decision boundary.

2.3 Projected gradient descent

Projected Gradient Descent (PGD) was proposed by Madry et al. [40]. PGD perturbs a clean image \mathbf{x} for several number of iterations with a small step size in the direction of the gradient of the loss function. Unlike BIM, after each perturbation step, it projects the resulting adversarial sample back onto the ϵ -ball of the 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:

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

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

2.4 Other popular attack strategies

Since the main focus of this study is to propose an alternative and effective strategy to traditional loss based attacks, we will only mention some of the other well-known attack types without discussing them in detail. One of the most popular attack algorithms to date is proposed by Carlini and Wagner [8]. Their attack strategy is based on redefining the attacks as optimization problems which can be solved by using gradient descent to craft more powerful and effective adversarial samples. To give another example, Moosavi-Dezfooli et al. [41] proposed an attack named Deepfool based on the assumption that the neural network models behave as linear classifiers and the classes are separated by a hyperplane. Their 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 $\mathbf{x}_t + \mathbf{1}$ by adding the minimal perturbation to the \mathbf{x}_{t+1} and checks whether misclassification has occurred.

Adversarial machine learning is a trending research field, and new attack algorithms are being proposed immensely. Some of the latest studies are Bandit [27], Square Attack [3], JSMA [46] and HSJA [10]. Moreover, there are some recent studies that utilize MC Dropout sampling and uncertainty information to craft adversarial samples. Liu et al. [38] introduced Universal Adversarial Perturbation (UAP) approach that utilizes a metric named virtual uncertainty, extracted from the model's structural activation. Nonetheless, estimating the model's uncertainty involves aggregating all the neurons' virtual uncertainties, which is computationally costly and unattractive. In this study, unlike traditional loss based attacks, we focus on developing an efficient and effective approach by utilizing epistemic uncertainty estimates of the model derived from final softmax score outputs.

3 Preliminaries

Traditionally, predictive models are used to be forced to provide a decision even in ambiguous cases where the model is not sure about its prediction, and the quality of its predictions is expected to be low. Assuming the model's prediction is always correct without any reasoning based on its uncertainty may result in catastrophic results. This fact led the researchers to propose various ways of quantifying uncertainty and to suggest abstaining models based on certain conditions like when the model's uncertainty is high, thus improving the reliability [33, 54].

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

3.1 Uncertainty in machine learning

There are two different types of uncertainty in machine learning: epistemic uncertainty and aleatoric uncertainty [2, 24, 56].

3.1.1 Epistemic uncertainty

Epistemic uncertainty refers to uncertainty caused by a lack of knowledge and limited data needed for a perfect predictor [4]. It can be categorized under 2 groups as *approximation uncertainty* and *model uncertainty* as depicted in Fig. 2.

Approximation uncertainty In a standard 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 choosing an appropriate learning method with its related hyperparameters minimizing the expected loss (risk) with a selected loss function, ℓ . However, what the learner does is to try to minimize the *empirical risk* R_{emp} which is an estimation of the 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.

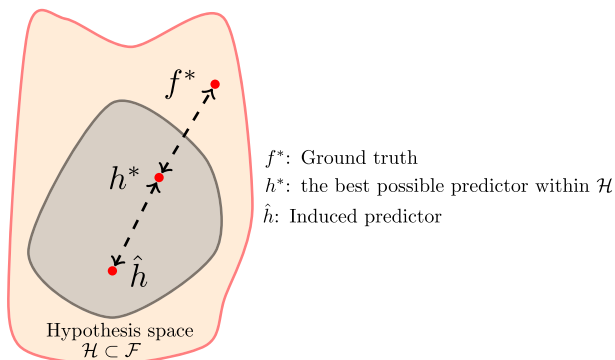


Fig. 2 Epistemic uncertainty

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 aim of finding a hypothesis function that can successfully map all possible inputs to outputs. This leads to a discrepancy 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 , there exists a neural network that can approximate f [12, 57]. The hypothesis space \mathcal{H} is huge for deep neural networks. Thus, it will not be wrong to assume that $h^* = f^*$. We can ignore the model uncertainty for deep neural networks, and only care about the approximation uncertainty. Consequently, in deep learning tasks, the actual source of epistemic uncertainty is related to approximation uncertainty.

Epistemic uncertainty refers to the confidence a model has about its prediction [39]. The underlying cause is the uncertainty about the parameters of the model. This type of uncertainty is apparent in areas with limited training data, and the model weights are not optimized correctly. High epistemic uncertainty is observed when the model is expected to predict a sample derived from a shifted version of the training data distribution or for an out-of-domain sample [16].

3.1.2 Aleatoric uncertainty

Aleatoric uncertainty refers to the variability in an experiment’s outcome, which is due to the inherent random effects [22]. This type of uncertainty can not be reduced even if we have enough training samples [49]. An excellent example of this phenomenon is the noise observed in the measurements of a sensor.

Figure 3 depicts a simple nonlinear function ($\sin(0.3 \times x)$ 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. As an example, these points may represent measurements of a faulty sensor; 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 areas of high epistemic uncertainty. Because, there are not enough training samples for our model to describe the data best. Moreover, we can assume that the area of high epistemic uncertainty area corresponds to the area with low prediction accuracy.

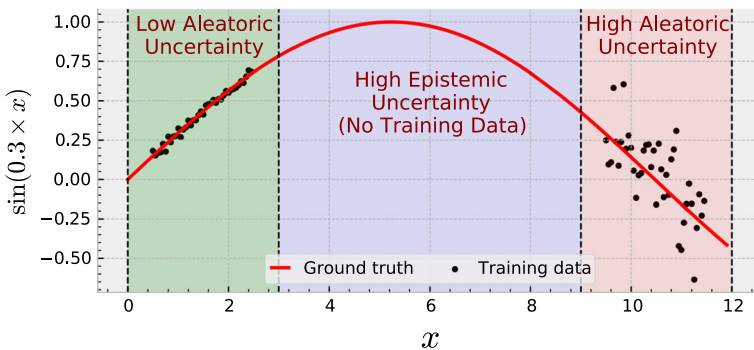


Fig. 3 Illustration of epistemic and aleatoric uncertainty

3.2 Quantifying epistemic uncertainty in deep neural networks

Using techniques that help us to quantify the uncertainty of the model is necessary for robust decision making. Assuming that we, as humanity, will use deep learning models in areas where safety and reliability are critical concerns as in the case of autonomous driving and 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 model predictions.

In recent years, a significant number of researches 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 [44]. However, Bayesian NN's come with an additional computational cost and inference issues. Therefore, several approximations to Bayesian methods have been developed which make use of variational inference [7, 20, 25, 45]. On the other hand, Lakshminarayanan et al. [32] 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. [15]. 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 (MC) dropout sampling during the prediction time.

Inference time dropout acts as an ensemble approach. In each single ensemble model, the system needs to drop out different neurons in network layers according to the dropout ratio in the prediction time. The predictive mean is the average of the predictions over dropout iterations, T , and the predictive mean is used as the final inference \hat{y} , for the input instance $\hat{\mathbf{x}}$ in the dataset. The overall prediction uncertainty is approximated by finding the variance of the probabilistic feed-forward Monte Carlo (MC) dropout sampling during prediction time. The final prediction is defined as follows:

$$p(\hat{y} = c | \hat{\mathbf{x}}, \mathcal{D}) \approx \hat{\mu}_{pred} = \frac{1}{T} \sum_{y \in T} p(\hat{y} | \theta, \mathcal{D}) \quad (4)$$

where θ is the model weights, \mathcal{D} is the input dataset, T is the number of predictions of the MC dropouts, and \mathbf{x} is the input sample. The label of input sample \mathbf{x} can be estimated with the mean value of Monte-Carlo dropout predictions $p(\hat{y} | \theta, \mathcal{D})$, which will be done T times.

Figure 4 shows 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 probability 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 class label of the input instance and quantify uncertainty of the overall prediction. For each testing of an input sample \mathbf{x} , the predicted label is assigned with the highest predictive mean. And the variance of the $p(\hat{y})$ is used as a measure of epistemic uncertainty of the model.

We chose the MC dropout method due to its simplicity and efficiency. The approach needs only a single trained model to measure the uncertainty, while different techniques such as Deep Ensemble need multiple models. Secondly, one can take the backward derivative of the computed variance term for each input sample and use it to craft adversarial samples to evade the model.

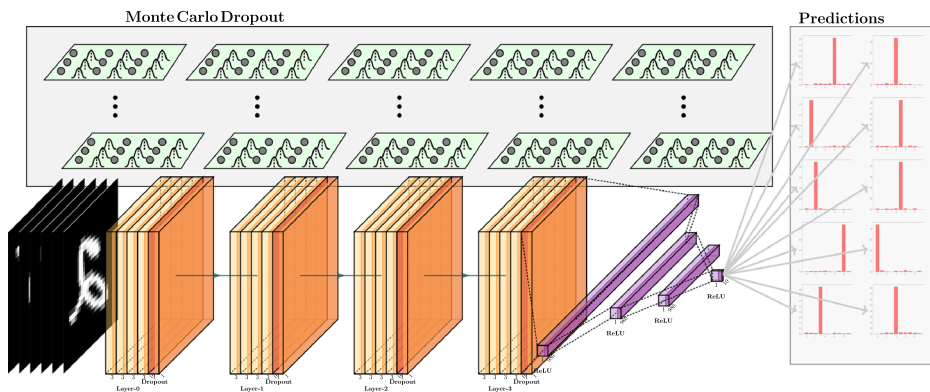


Fig. 4 Illustration of the Monte Carlo dropout based Bayesian prediction

4 Approach

The uncertainty of the model is higher in areas with limited number of training points. Due to this ignorance about ground truth, we cannot achieve a perfect model that can predict accurately every possible test data. Figure 5 shows the prediction of a regression model trained on a limited number of data points constrained on some interval. In this simple example, we trained a single hidden layer NN with ten neurons to learn a linear function $y = -x + 1$. As can be seen from the graph, in the areas where we do not have enough training points, the uncertainty values obtained from MC dropout estimates of our model is high, which can be interpreted as the quality of the prediction is low, and our model is having difficulties in deciding the correct output values. Harmoniously, we also observe high error in these areas. For this reason, we can conclude that the high epistemic uncertainty area coincides with the low prediction accuracy area. Accordingly, we claim that pushing

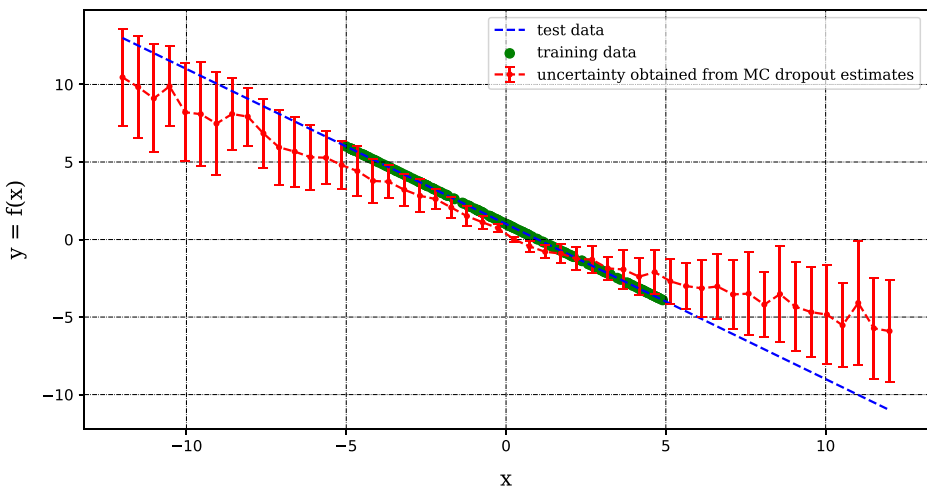


Fig. 5 Uncertainty estimates obtained from a regression model

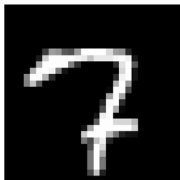
the model's limits by testing it in extreme conditions with input outside from training data distribution (input from a shifted-domain) may cause model prediction failure.

The adversarial attacks aim to find the necessary perturbation amount (δ) constrained to some interval (ϵ), resulting in maximum loss, thus fooling the classifier. We can express this mathematically in the below equation, where $F_\theta(x)$ is our neural network.

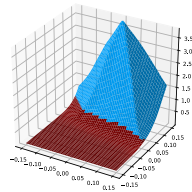
$$\arg \max_{\|\delta\| \leq \epsilon} \ell(F_\theta(x + \delta), y) \quad (5)$$

Like most of the attack types in literature, the attackers can perturb the input image in a direction that maximizes the loss, and this direction is found using the gradient of the loss function. However, we showed that instead of using the loss function, another effective approach is to use the model's epistemic uncertainty. Our alternative method uses the model's epistemic uncertainty as a tool for creating successfully manipulated adversarial input instances. In contrast to the loss based adversarial machine learning attacks, this method can provide an alternative strategy in which the attacker can make an effective attack by exploiting uncertainty due to the difficulty of the model to interpret the shifted-domain sample based on the data observed during training.

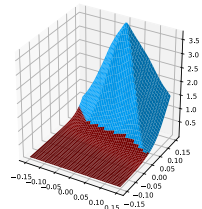
To verify that our intuition holds true, we have done a simple experiment and depicted the loss surfaces of a trained CNN model (digit classifier) within a constrained epsilon neighbourhood of the original input data points, as shown in Fig. 6. Figure 6b shows the model's loss values in the direction of the model's loss gradient and a random direction. We see that the maximum loss value observed is 3.783. Then, as shown in Fig. 6c, we depicted the model loss surface in the direction of the model's epistemic uncertainty's gradient and the same random direction we used in the previous try. This time, the maximum loss value achieved is 3.713, which is close to the previous one. We observed that out of 784 sub



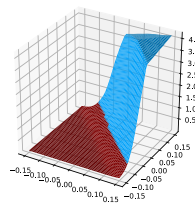
(a) MNIST Digit test image number 37



(b) Loss gradient direction



(c) Uncertainty gradient direction



(d) Hybrid gradient direction

Fig. 6 Loss surfaces in different directions. The maximum loss values are 3.783, 3.713, 4.167 for b, c and d respectively

directions, 693 were the same and 91 were different according to the directions of loss and uncertainty gradients. The model loss can be maximized by perturbing the input image in a slightly different direction than we used to do before. Lastly, we depicted the model loss surface in the direction of loss and uncertainty’s gradient directions as in Fig. 6d. We reached a loss value of 4.167, which is bigger than the previous two attempts. In Fig. 6b, c and d, the points where there is a difference in color on the loss surfaces indicate that the model prediction has changed from the correct class 7 to wrong class 2. Therefore, we can conclude that perturbing the image in both directions will lead to misclassification for the model.

The loss surfaces of DNN models are well-known to be highly non-linear, with many local minimums and maximums in high-dimensional space. Numerical solution to finding global extrema points is an NP-hard problem [6, 28]. No optimization approach can reach these global extrema points by utilizing a naive method like gradient descent on the model’s loss function. Eventually, the optimizer will be stuck to local extrema points. However, the above experiment shows that slightly changing the direction in each gradient descent step by leveraging the model uncertainty can increase the proposed attacks’ performance.

We conducted the same experiment on a different sample from the MNIST test dataset. Figure 7 shows that the maximum loss value in uncertainty’s gradient direction is far greater than the maximum loss value in model loss’ gradient direction. And the maximum loss value in the hybrid direction is larger than the ones in both model loss’ and uncertainty’s directions. Besides, we observe that there is no possibility of misclassification in the loss

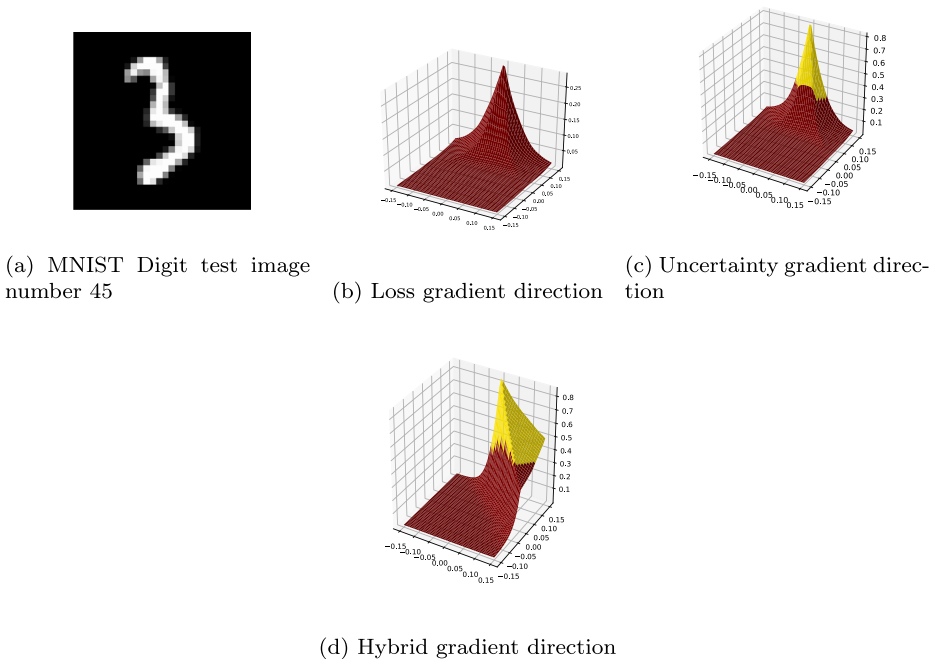


Fig. 7 Loss surfaces in different directions. The maximum loss values are 0.285, 0.811, 0.845 for b, c and d respectively

gradient direction, as there is no visible colour change in the surface plot of Fig. 7b, whereas in Fig. 7c we observe that there are yellow regions where the model misclassifies the input image in the uncertainty's gradient direction. Again, when we analyzed the directions of loss' and uncertainty's gradients, we saw that out of 784 directions, only 639 of them were the same and 145 of them were different, which is much larger than the first experiment.

The epistemic uncertainty yields a better direction for our second experiment because our model (like all the trained ML models) is not the "perfect" predictor and is just an approximation to the oracle function. The model itself has an inherent "approximation uncertainty" which sometimes induce sub-optimal solutions. Consequently, any method which only relies on the trained model (which is not the optimum model) will result in less effective performance.

4.1 Proposed epistemic uncertainty based attacks

Previous attack types in literature have been designed to exploit the model loss and maximize the model loss value within a constrained neighbourhood of the input data points. And we have witnessed quite successful results with this approach. However, one possible drawback for these 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 improves the convergence to global extrema points by yielding a higher loss value. Results shown in Figs. 6 and 7 support our argument. Therefore, we can reformulate existing attacks using model *uncertainty* instead of model *loss*. And even we can benefit from both of them.

4.1.1 Fast gradient sign method (uncertainty-based)

The formulation used in traditional loss-based FGSM attack is given below:

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \ell(\mathbf{x}, y_{true})) \quad (6)$$

where \mathbf{x} is the input (clean) image x^{adv} is the perturbed adversarial image, ℓ is the classification loss function, y_{true} is true label for the input \mathbf{x} .

Our modified FGSM attack (uncertainty-based) is shown as;

$$\mathbf{x}^{adv} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} U(\mathbf{x}, F, p, T)) \quad (7)$$

where \mathbf{x} is the input (clean) image, \mathbf{x}^{adv} is the perturbed adversarial image, U is the uncertainty metric (mean variance) obtained from T different MC dropout estimates, F 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.

And the steps necessary for the calculation of uncertainty metric (mean variance of T predictions) is given as below:

Step: 1 For an input image \mathbf{x} , T different predictions is obtained $p_t(\mathbf{x})$ by Monte-Carlo Dropout sampling where each prediction is a vector of softmax scores for the C classes.

$$p_t(\mathbf{x}) = \mathcal{F}(\mathbf{x}, p, T)$$

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

$$p_T(\mathbf{x}) = \frac{1}{T} \sum_{t \in T} p_t(\mathbf{x})$$

Step 3: 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: Compute the expected value of variance over all classes by taking their average.

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

4.1.2 Basic iterative attack (uncertainty-based)

In this section, we first provide the pseudo-codes for known loss-based BIM attack types as in Algorithm 1 and 2. Then, we provide our proposed uncertainty-based BIM attack variants in Algorithm 3 and 4. All the attack types proposed here are designed under L_∞ norm.

Algorithm 1 Algorithm for BIM A (loss-based) \mathbf{x} is the benign image, y_{true} is the true label for \mathbf{x} , F is the model function learnt during training, N is the number of iterations, ϵ is the maximum allowed perturbation, α is the step size.

Input: $\mathbf{x} \in \mathbb{R}^m, y_{true}, F, N, \epsilon, \alpha$
Output: \mathbf{x}_{t+1}

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2 while  $n < N$  do
    /* update X by using below formula, F in evaluation mode */
3    $\mathbf{x}_{(t+1)} = clip_{\mathbf{x}, \epsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$ 
4   if  $arg\ max(F(\mathbf{x}_{t+1})) \neq y_{true}$  then
5     | end while
6 return  $\mathbf{x}_{t+1}$ 

```

Algorithm 2 Algorithm for BIM B (loss-based) \mathbf{x} is the benign image, y_{true} is the true label for \mathbf{x} , F is the function learned by the network during training, N is the number of iterations, ϵ is the maximum allowed perturbation, α is the step size.

Input: $\mathbf{x} \in \mathbb{R}^m, y_{true}, F, N, \epsilon, \alpha$
Output: \mathbf{x}_{t+1}

```

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

```

Algorithm 3 Algorithm for BIM A (uncertainty-based) \mathbf{x} is the benign image, F is the model 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, ϵ is the maximum allowed perturbation, α is the step size.

Input: $\mathbf{x} \in \mathbb{R}^m$, F , p , T , N , ϵ , α
Output: \mathbf{x}_{t+1}

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2  $initialprediction = arg\ max(F(\mathbf{x}_0))$ 
3 while  $n < N$  do
    /* update X by using below formula, F in training mode */
4    $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\epsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}}U(\mathbf{x}_t, F, p, T)))$ 
5   if  $arg\ max(F(\mathbf{x}_{t+1})) \neq initialprediction$  then
6     └ end while
7 return  $\mathbf{x}_{t+1}$ 

```

Algorithm 4 Algorithm for BIM B (uncertainty-based) \mathbf{x} is the benign image, F is the model 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, ϵ is the maximum allowed perturbation, α is the step size.

Input: $\mathbf{x} \in \mathbb{R}^m$, F , p , T , N , ϵ , α
Output: \mathbf{x}_{t+1}

```

1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2 while  $n < N$  do
3   if  $arg\ max(F(\mathbf{x}_{t+1})) \neq initialprediction$  then
4     └  $condition = True$ 
5   if  $condition = False$  then
6     /* update X by using below formula, F in training mode */
7     └  $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\epsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}}U(\mathbf{x}_t, F, p, T)))$ 
8   else
9     /* update X by using below formula, F in training mode */
10    └  $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\epsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}}U(\mathbf{x}_t, F, p, T)))$ 
11 return  $\mathbf{x}_{t+1}$ 

```

4.1.3 Basic iterative attack (hybrid approach)

Here, we present the pseudo-code for our Hybrid Approach in Algorithm 5. Same as the previous BIM attack variants, our Hybrid Approach is also designed under ℓ_∞ norm. At each iteration, we step into both the model loss' gradient direction and model uncertainty's gradient direction. These two metrics make up for each other and yield to a better result.

Algorithm 5 Algorithm for BIM A (Hybrid Approach) \mathbf{x} is the benign image, F is the model 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, ϵ is the maximum allowed perturbation, α is the step size.

```

Input:  $\mathbf{x}, F, p, T, N, \epsilon, \alpha$ 
Output:  $\mathbf{x}_{t+1}$ 
1  $\mathbf{x}_0 \leftarrow \mathbf{x}$ 
2  $initialprediction = arg\ max(F(\mathbf{x}_0))$  /* F in evaluation mode */
3 while  $n < N$  do
    /* update X by using below formula, F in training mode
       when calculating gradient of model uncertainty, F
       in evaluation mode when calculating gradient of
       model loss */
4  $\mathbf{x}_{(t+1)} = clip_{\mathbf{x}, \epsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}}U(\mathbf{x}_t, F, p, T)) + \alpha \cdot sign(\nabla_{\mathbf{x}}\ell(\mathbf{x}_t, y_{true})))$ 
5 if  $arg\ max(F(\mathbf{x}_{t+1})) \neq initialprediction$  then
6     | end while
7 return  $\mathbf{x}_{t+1}$ 
    
```

4.2 Visualizing gradient path for uncertainty-based attacks

Figure 8 shows a simplified example of the gradient path for our uncertainty-based BIM attack variants. In the example shown in the figures, the low uncertainty regions are shown in blue, while the high uncertainty regions are shown in red. Figure 8a shows an example of successful uncertainty-based BIM attack type-A. But, we would expect the uncertainty-based BIM attack type-B to be unsuccessful for this specific example. Because at the intermediate iteration where we passed the decision boundary from source to target class, we are at the left side of the uncertainty hill. Therefore, when we try to decrease the uncertainty, we will perturb the image back to the original class manifold. However, for Fig. 8b, we would expect both uncertainty-based BIM attack types A and B would be successful. Because this time, at the intermediate iteration where we passed the decision boundary from source to target class, we are at the right side of the uncertainty hill.

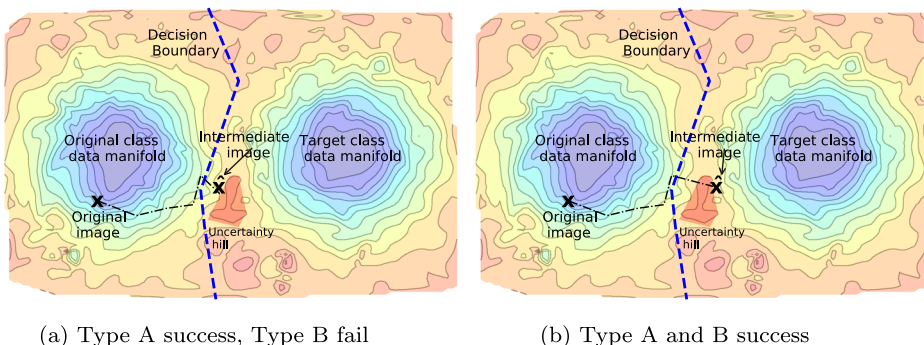


Fig. 8 Uncertainty gradient path

4.3 Capability of the attacker

We assumed that the attacker's primary purpose is to evade the model by applying a carefully crafted perturbation to the input data. In a real-world scenario, the white-box setting is the most desired choice for an adversary that does not take the risks of being caught in a trap. The problem is that it requires the attacker to access the model from outside to generate adversarial examples. After capturing model information, the attacker can exploit the model's vulnerabilities in the same manner as an adversary's sandbox environment.

However, the attacker must solve an optimization problem to decide which regions in the input data must be changed to prevent this manipulation from being easily noticed by the human eye. By solving this optimization problem using one of the available attack methods [1, 19, 30, 40], the attacker aims to reduce the classification performance of the model on the adversarial data as much as possible. In this study, to limit the maximum allowed perturbation for the attacker, we used ℓ_∞ norm, which is the maximum pixel difference limit between original and adversarial images.

5 Results

5.1 Experimental Setup

We trained our CNN models for the MNIST (Digit) [34] and MNIST (Fashion) [55] datasets, and we achieved accuracy rates of 99.05 % and 91.15 % respectively. The model architectures are given in Table 1 and the hyperparameters selected in Table 2. For the CIFAR10 dataset [29], we used a pretrained VGG-A (11 weight layers) model [51] and then apply transfer learning by freezing the convolution layers, changing the number of neurons in the output layer from 1000 to 10 and updating the weights of the dense layers only for 10 epoch. In this way, we achieved an accuracy rate of 89.07 % on test data. Since the used pretrained VGG model was trained on IMAGENET dataset, we had to rescale the CIFAR10 images from 32×32 to 224×224 . We also applied the same 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 our experiments are provided in Table 3. Finally, we used $T = 50$ as the number of MC dropout samples when quantifying uncertainty.

5.2 Visualizing uncertainty under different attack variants

Figure 9 shows the change in our quantified uncertainty values of the model during different BIM attack variants. In this experiment, we applied all of the attack variants to the 23th test sample from MNIST (Digit) dataset. The original label of the input image was 6. For type A and B of loss and uncertainty based attacks, we observed that at 11th and 13th iterations, respectively, the attack is successful, and the input image was misclassified as 4. In Fig. 9a, we stop the iteration as soon as we succeeded in fooling the model, whereas in Fig. 9b, we continue to perturb the image, but this time in a direction which minimize the uncertainty. After the last iteration, the predicted label was still 4, and the uncertainty level decreased compared to the time of misclassification. For this sample, our uncertainty-based BIM attack type-B was successful, because, when we pass the decision boundary as we try to maximize model uncertainty, we also go beyond the point where there is the maximum uncertainty. One last important point to mention is that, when we apply the hybrid approach

Table 1 CNN model architectures

Dataset	Layer Type	Layer Information
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 10$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 10$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 20$
	Dropout	p : 0.25
	Convolution (padding:1) + ReLU	$3 \times 3 \times 20$
	Dropout	p : 0.25
	Fully Connected + ReLU	980
	Fully Connected + ReLU	100
Output Layer	10	
MNIST (Fashion)	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Max Pooling	2×2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Max Pooling	2×2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.5
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.5
	Fully Connected + ReLU	3136
	Fully Connected + ReLU	600
Fully Connected + ReLU	120	
Output Layer	10	

Table 2 Model training parameters

Parameter	CNN Model for MNIST (Digit)	CNN Model for MNIST (Fashion)
Optimizer	SGD	Adam
Learning rate	0.01	0.001
Batch Size	100	64
Dropout Ratio	0.25	0.25
Epochs	7	10

Table 3 Adversarial settings of our experiments: α , i respectively denote the step-size and the number of attack steps for a perturbation budget ϵ

Attack	Parameters	l_p norm
FGSM	$i = 1$	l_∞
BIM	$\alpha = \epsilon \cdot 0.2, i = 10$ (for MNIST)	l_∞
BIM	$\alpha = \epsilon \cdot 0.3, i = 5$ (for CIFAR-10)	l_∞

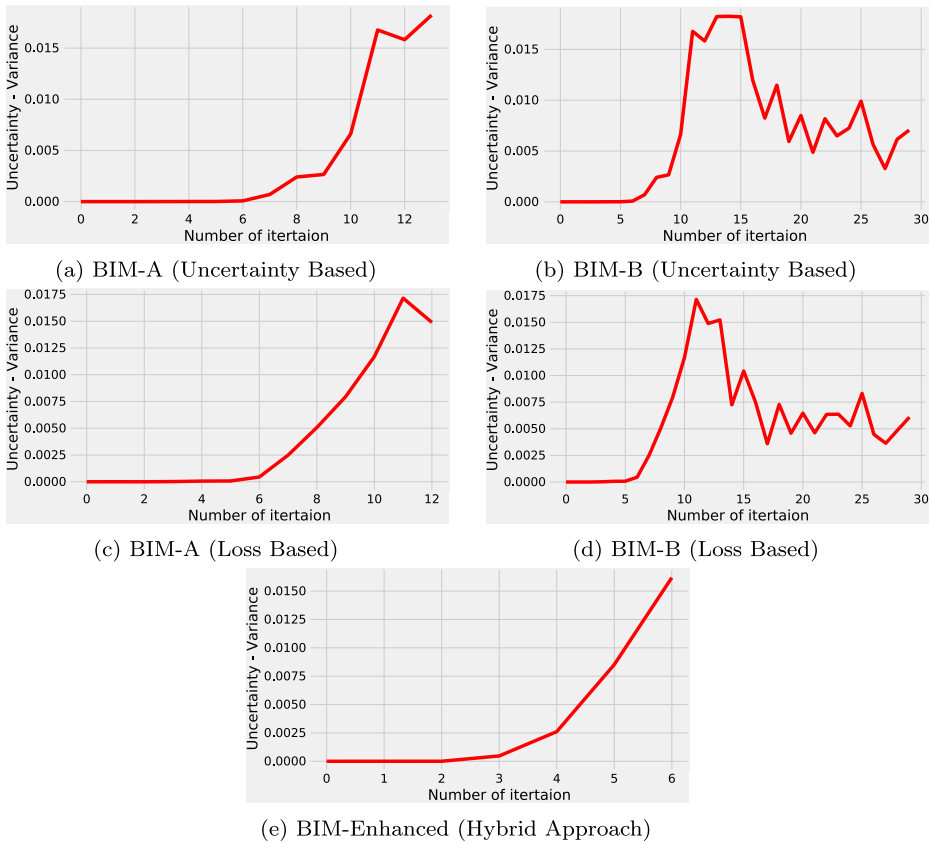


Fig. 9 Change of uncertainty values during different BIM attack variants

where we utilize both loss and uncertainty, we could successfully fool the model after 6th iteration, which is much faster. This also proves our assumption that the hybrid approach is more effective than the others.

5.3 Experimental results

During our experiments, we only perturbed the test samples which were correctly classified by our models in their original states. Because, an intruder would have no reason to perturb samples that are already classified wrongly.

The results show that our Hybrid Approach of using both model's loss and uncertainty results into the best performance. Success rates of pure loss-based and pure uncertainty-based attacks are similar to each other. We also observe that the success rates for uncertainty-based attack types A and B are different. We argue that the point of global maximum for uncertainty metric for any class is not on the model's decision boundary as in the case of model loss. Instead, the point where the uncertainty is maximum can be beyond the decision boundary. Therefore, during the gradient-based search, it may be possible for us to pass the decision boundary but still not reaching the peak value for uncertainty. And when we start to decrease the uncertainty after passing the decision boundary (fooling the

model), it may be possible to go back to the original class. However, this is not the case in loss-based approaches. Since we are trying to maximize the loss based on a reference class, we always see an increasing trend during the gradient descent approach of the loss

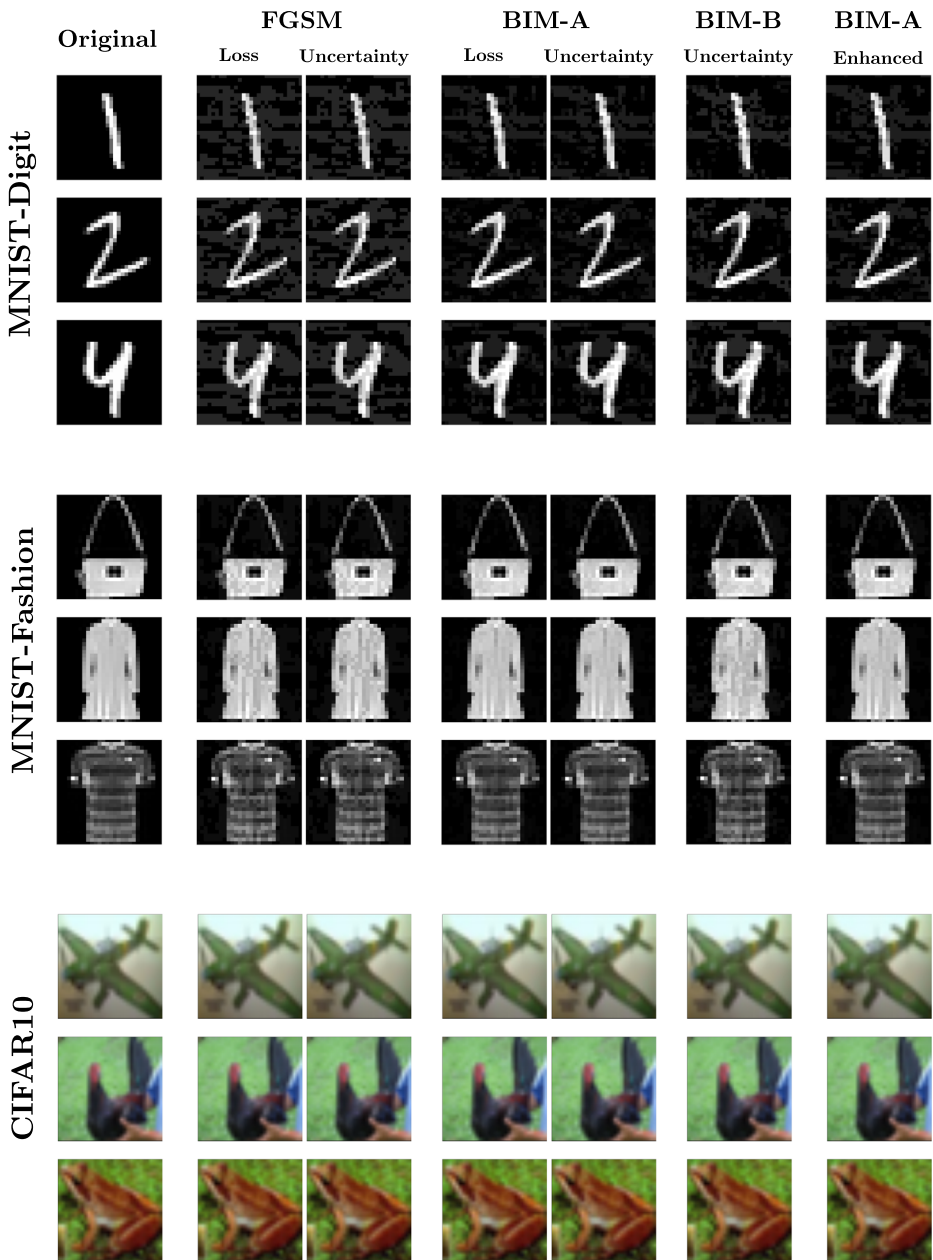


Fig. 10 Some example images from MNIST(Digit), MNIST(Fashion) and CIFAR-10. The original image is shown in the left-most column and adversarial samples crafted based on different methods are on the other columns

Table 4 Attack success rates on different datasets

	MNIST (Digit) $\epsilon = 0.15$	MNIST (Fashion) $\epsilon = 0.05$	CIFAR-10 $\epsilon = 0.2/255$
FGSM (loss-based)	% 46.81	%59.28	%72.42
FGSM (uncertainty-based)	% 47.29	%62.89	%71.52
BIM-A (loss-based)	% 82.56	% 82.96	%89.44
BIM-A (uncertainty-based)	% 76.09	% 84.83	%86.77
BIM-B (uncertainty-based)	% 65.43	% 71.71	%82.98
BIM-A (Hybrid Approach)	% 85.14	% 90.13	%91.06

maximization journey. Figure 10 shows some examples of adversarial samples crafted using different methods mentioned in this study. Table 4 shows the results of our experiments.

6 Conclusion

In this study, we proposed new attack algorithms by perturbing the input in a direction that maximizes the model's epistemic uncertainty instead of its loss. We observed almost similar performances compared to loss based approaches. We also introduced a new concept for finding better points resulting in higher loss values within a specified ℓ_p norm interval to craft adversarial samples. For this, we used a hybrid approach and stepped into gradient directions of both loss and uncertainty in each gradient descent step. We showed that the attack success rates are higher when we utilize this approach.

The aim of this study was not to propose the most powerful attack to date. Instead, we aimed to show that there exist other powerful metrics, different from model's loss, that can be exploited to craft adversarial examples. Besides, we empirically demonstrated that relying just on the trained model may not always be the greatest option as it is just an approximation to the best predictor whereas epistemic uncertainty information can be very advantageous in cases where the model is misleading. We also showed that the combined usage of uncertainty and loss yields better performance in attacks.

In future work, we will be investigating the possible usage of uncertainty metrics for defense purposes. We believe that accurate efforts for minimizing the quantified uncertainty of an input sample can help to reduce any possible adversarial effect and thereby assist to restore the input sample back to its original data manifold.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. 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>
2. 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>

3. Andriushchenko M, Croce F, Flammarion N, Hein M (2020) Square attack: a query-efficient black-box adversarial attack via random search
4. Cortellessa V, Gribaudo M, Pincirolri R, Trivedi KS, Trubiani C (2020) Analytical modeling of performance indices under epistemic uncertainty applied to cloud computing systems, vol 102. <https://doi.org/10.1016/j.future.2019.09.006>
5. Ayhan M, Berens P (2018) Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks
6. Blum AL, Rivest RL (1992) Training a 3-node neural network is np-complete, vol 5. [https://doi.org/10.1016/S0893-6080\(05\)80010-3](https://doi.org/10.1016/S0893-6080(05)80010-3)
7. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks
8. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks
9. Catak FO, Sivaslioglu S, Sahinbas K (2020) A generative model based adversarial security of deep learning and linear classifier models
10. Chen J, Jordan MI, Wainwright MJ (2020) Hopskipjumpattack: a query-efficient decision-based attack. In: 2020 IEEE Symposium on security and privacy (SP), pp. 1277–1294. <https://doi.org/10.1109/SP40000.2020.00045>
11. 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>
12. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)* 2(4):303–314. <https://doi.org/10.1007/BF02551274>
13. Eaton-Rosen Z, Bragman F, Bisdas S, Ourselin S, Cardoso MJ (2018) Towards safe deep learning: accurately quantifying biomarker uncertainty in neural network predictions
14. Finlayson SG, Chung HW, Kohane IS, Beam AL (2019) Adversarial attacks against medical deep learning systems
15. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: representing model uncertainty in deep learning
16. 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
17. Ghoshal B, Tucker A (2020) Estimating uncertainty and interpretability in deep learning for coronavirus (covid-19) detection
18. Goodfellow I, Bulatov Y, Ibarz J, Arnold S, Shet V (2014) Multi-digit number recognition from street view imagery using deep convolutional neural networks
19. Goodfellow I, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples
20. 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, pp 2348–2356. Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf>
21. Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks
22. 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>
23. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition
24. Hüllermeier E, Waegeman W (2020) Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods
25. Hoffman M, Blei D, Wang C, Paisley J (2013) Stochastic variational inference
26. 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. *Computer Science Review* 37:100270. <https://doi.org/10.1016/j.cosrev.2020.100270>
27. Ilyas A, Engstrom L, Madry A (2019) Prior convictions: black-box adversarial attacks with bandits and priors
28. Judd JS (1990) *Neural network design and the complexity of learning*. MIT press, cambridge
29. Krizhevsky A, Nair V, Hinton G Cifar-10 (canadian institute for advanced research). <http://www.cs.toronto.edu/kriz/cifar.html>
30. Kurakin A, Goodfellow I, Bengio S (2017) Adversarial examples in the physical world
31. Kurakin A, Goodfellow I, Bengio S (2016) Adversarial machine learning at scale. arXiv:1611.01236
32. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles
33. Laves MH, Ihler S, Ortmaier T, Ortmaier T (2019) Uncertainty quantification in computer-aided diagnosis: make your model say “i don’t know” for ambiguous cases
34. LeCun Y, Cortes C (2010) MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>

35. Liao X, Li K, Yin J (2017) Separable data hiding in encrypted image based on compressive sensing and discrete fourier transform. *Multimedia Tools and Applications* 76(20):20739–20753. 10.1007/s11042-016-3971-4
36. Liao X, Qin Z, Ding L (2017) Data embedding in digital images using critical functions. *Signal Process Image Commun* 58:146–156. <https://doi.org/10.1016/j.image.2017.07.006>
37. Liao X, Yin J, Chen M, Qin Z (2020) Adaptive payload distribution in multiple images steganography based on image texture features. *IEEE Transactions on Dependable and Secure Computing*, pp 1–1. <https://doi.org/10.1109/TDSC.2020.3004708>
38. 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>
39. Loquercio A, Segu M, Scaramuzza D (2020) A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* 5(2):3153–3160. <https://doi.org/10.1109/ra.2020.2974682>
40. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2019) Towards deep learning models resistant to adversarial attacks
41. Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: a simple and accurate method to fool deep neural networks
42. Morgulis N, Kreines A, Mendelowitz S, Weisglass Y (2019) Fooling a real car with adversarial traffic signs
43. Nair T, Precup D, Arnold DL, Arbel T (2018) Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation
44. Neal RM (1996) Bayesian learning for neural networks. Springer, Berlin
45. Paisley J, Blei D, Jordan M (2012) Variational bayesian inference with stochastic search
46. Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A (2017) Practical black-box attacks against machine learning
47. 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 Communications Surveys Tutorials* 22(2):998–1026. <https://doi.org/10.1109/COMST.2020.2975048>
48. Sadeghi K, Banerjee A, Gupta SKS (2020) A system-driven taxonomy of attacks and defenses in adversarial machine learning. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4(4):450–467. 10.1109/TETCI.2020.2968933
49. Senge R, Börsner 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>
50. Serban AC, Poll E, Visser J (2019) Adversarial examples - a complete characterisation of the phenomenon
51. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition
52. Sitawarin C, Bhagoji AN, Mosenia A, Chiang M, Mittal P (2018) Darts: deceiving autonomous cars with toxic signs
53. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks
54. Tuna OF, Catak FO, Eskil MT (2020) Closeness and uncertainty aware adversarial examples detection in adversarial machine learning
55. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms
56. 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>
57. Zhou DX (2018) Universality of deep convolutional neural networks

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.