International Journal of Information Security (2023) 22:987–1004
https://doi.org/10.1007/s10207-023-00675-1

REGULAR CONTRIBUTION

TENET: a new hybrid network architecture for adversarial defense

Omer Faruk Tuna[1] · Ferhat Ozgur Catak[2] · M. Taner Eskil[3]

Published online: 17 March 2023

Abstract
Deep neural network (DNN) models are widely renowned for their resistance to random perturbations. However, researchers have found out that these models are indeed extremely vulnerable to deliberately crafted and seemingly imperceptible perturbations of the input, referred to as adversarial examples. Adversarial attacks have the potential to substantially compromise the security of DNN-powered systems and posing high risks especially in the areas where security is a top priority. Numerous studies have been conducted in recent years to defend against these attacks and to develop more robust architectures resistant to adversarial threats. In this study, we propose a new architecture and enhance a recently proposed technique by which we can restore adversarial samples back to their original class manifold. We leverage the use of several uncertainty metrics obtained from Monte Carlo dropout (MC Dropout) estimates of the model together with the model's own loss function and combine them with the use of defensive distillation technique to defend against these attacks. We have experimentally evaluated and verified the efficacy of our approach on MNIST (Digit), MNIST (Fashion) and CIFAR10 datasets. In our experiments, we showed that our proposed method reduces the attack's success rate lower than 5% without compromising clean accuracy.

Keywords Adversarial machine learning · Model uncertainty · Robustness · Monte Carlo dropout sampling

1 Introduction

Deep learning algorithms have started to outperform humans in the past few years. For example, in the "ImageNet Large Scale Visual Recognition Challenge (ILSVRC)", a deep learning model called ResNet [1] beat human performance in 2015, and the record was later broken by more advanced architectures. Similarly, Goodfellow et al. [2] created a system that outperforms human operators for the problem of reading addresses from Google Street View imagery and solving CAPTCHAS. In the field of gaming, AlphaGo, an AI program, defeated the global Go champion in 2016. Many advanced systems are now being developed using deep learning models, which have shown to be extremely successful in a variety of domains, including medical diagnosis, autonomous vehicles, game play, and machine translation. However, the main emphasis of the researchers during the rise of deep learning models was the creation of increasingly precise models and the reliability and robustness of those models were paid almost no attention. DNN's do, in fact, necessitate a more thorough examination because they have some inherent vulnerabilities that can be easily exploited by intruders.

Around the end of 2013, researchers discovered that existing DNN models are vulnerable to meticulously crafted attacks. Szegedy et al. [3] were among the very first who noticed the presence of adversarial instances in the domain of image classification. The authors have demonstrated that it is possible to modify an image by a small amount to change the prediction of the deep learning model. It is shown that a very slight and nearly unnoticeable change in input is enough to deceive even the most advanced classifiers and cause incorrect classification. Back then, a vast number of research studies have been undertaken in this new field named "Adversarial Machine Learning" and these studies have not been restricted just to image classification domain. For example, Sato et al. [4] demonstrated in the NLP domain that altering

✉ Omer Faruk Tuna
omer.tuna@ericsson.com

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

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

1 Ericsson Research, Istanbul, Turkey

2 Department of Electrical Engineering and Computer Science, University of Stavanger, Stavanger, Rogaland, Norway

3 Isik University, Istanbul, Turkey

merely one word from an input sentence can deceive a sentiment analyzer trained with textual data. A further example is in the audio domain [5],where the authors built targeted adversarial audio samples in autonomous speech recognition task by introducing very little disturbance to the original waveform. The result of this study shows that the target model may simply be exploited to transcribe the input as any desired phrase.

Adversarial evasion attacks mainly work by modifying the input samples in a way that increases the likelihood of making incorrect decisions, resulting in inaccurate predictions. These attacks can cause the model's prediction performance to deteriorate since the algorithm is unable to correctly predict the real output for the input instances. Attacks that take advantage of DNN's weakness can substantially compromise the security of these machine learning (ML)-based systems, often with disastrous results. In the context of medical applications, a malicious attack could result in an inaccurate disease diagnosis. As a result, it has the potential to impact the patient's health as well as the healthcare industry [6]. Similarly, self-driving cars employ ML to navigate traffic without the need for human involvement. A mistaken decision for the autonomous vehicle based on a adversarial attack could result in a tragic accident [7,8]. Hence, defending against malicious attacks and boosting the robustness of ML models without sacrificing clean accuracy is critical. Presuming that these ML models are to be utilized in crucial areas, we should pay utmost attention to both the performance of ML models and the security problems of these architectures.

In this research work, we concentrate on adversarial defense strategies based on moment-based uncertainty estimates of a distilled model which are obtained from Monte Carlo (MC) Dropout samples. We propose a hybrid approach by using and significantly improving the effectiveness of uncertainty-based reversal technique [9] and combining it with defensive distillation technique to provide more robust models. We name our proposed network architecture `TENET` by inspiring from the famous sci-fi movie *TENET* (Directed by Christopher Nolan, Warner Bros. Pictures and Syncopy Inc., 2020) due to resemblance of main concepts (inversion). We developed two more effective variants of the reversal process based on scibilic uncertainty. Reversal method involves reverting the input sample back to its original data manifold by decreasing its quantified uncertainty before feeding it to the classifier. This technique would be impossible if we used only one metric whose calculation is dependent on a reference information like model loss. However, as the quantification of model uncertainty is independent of any reference information like real label of the input, we can successfully restore the inputs back to their original data manifold by minimizing the quantified uncertainty. Our codes are released on GitHub [1] for scientific use.

To summarize; our key contributions for this work are as follows:

– We enhanced the performance of a recently proposed technique which can successfully restore adversarial samples back to their original class manifold and introduced two more effective variants of it.
– To the best of our knowledge, we are the first in the research community that consider scibilic uncertainty to build robust models.
– We introduce a hybrid architecture which combines defensive distillation technique and uncertainty-based reversal method. We experimentally show that these two approaches can handle complementary situations and they together help to reduce the success rate of different attacks like FGSM, BIM, PGD, DeepFool and CW lower than 5%.

This study is structured as follows: Sect. 2 goes over some of the most well-known attack types and defense techniques in the literature. In Sect. 3, we introduce the concept of uncertainty as well as main types and describe how we can quantify them. The details of our approach are presented in Sect. 4. We provide our experimental findings in Sect. 5 and wrap up our research in Sect. 6.

2 Literature survey

Since the uncovering of DNN's vulnerability to adversarial attacks [3], a lot of work has gone into inventing new adversarial attack algorithms and defending against them by utilizing more robust architectures [10–13]. We discuss some of the noteworthy attack and defense studies separately.

2.1 Adversarial attacks

DNN models have some vulnerabilities that make them challenging to defend in adversarial settings. For example, they are mostly sensitive to slight changes in the input data, leading to unexpected results in the model's predictions. Figure 1 depicts how an adversary could take advantage of such a vulnerability and fool the model using properly crafted perturbation applied to the input.

In general, adversarial strategies can be classified based on different criteria. Considering the final aim of the attacker, attacks can be grouped into two as targeted and untargeted attacks. In the former, the attacker tampers with the input image, causing the model to predict a class other than the

[1] https://github.com/omerfaruktuna/TENET-Adversarial-Defense.

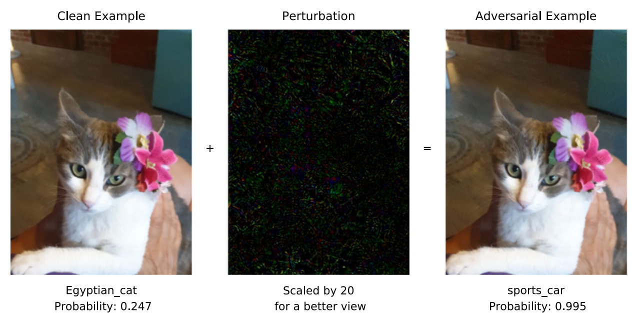


Fig. 1 The figure shows a simple example to adversarial attack. The adversarial perturbation is applied upon the original image. The precisely crafted perturbation manipulates the model in such a way that a "Cat" is wrongly classified as "Sports Car" with very high degree of confidence

genuine class. Whereas in the latter, the attacker perturbs the input image so that a particular target class is predicted by the model. Attacks can also be grouped based the level of knowledge that the attacker has. If the attacker has full knowledge of the model like architecture, weights, hyper-parameters, etc., we call this kind of setting as White-Box Settings. However, if the attacker has no information of the deployed model and defense strategy, we call this kind of setting as Black-Box Settings [14]. In this study, we mainly focus on untargeted attacks in a White-Box setting.

The majority of attack ideas rely on perturbing the input sample in order to maximize the model's loss. In recent years, many different adversarial attack techniques have been suggested in literature. The most widely known and used adversarial attacks are *Fast-Gradient Sign Method, Iterative Gradient Sign Method, Projected Gradient Descent,* `DeepFool` *and Carlini&Wagner*. These five adversarial attack algorithms are briefly explained in Sects. 2.1.1–2.1.4.

2.1.1 Fast-gradient sign method

This approach, sometimes known as FGSM [15], is among the first and most famous adversarial attacks so far. In this attack algorithm, the derivative of the model's loss function with respect to the input sample is used to identify which direction the input image's pixel values should be altered in order to minimize the model's loss function. Once extracted, it alters all pixels in the opposite direction simultaneously to maximize the loss. We may craft adversarial samples for a model with a classification loss function represented as $J(\theta, \mathbf{x}, y)$ by utilizing the formula below, where θ denotes the parameters of the model, $\mathbf{x}$ is the benign input, and y_{true} is the real label of our input.

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \epsilon \cdot \text{sign}\left(\nabla_x J(\theta, \mathbf{x}, y_{\text{true}})\right) \tag{1}$$

Another important aspect of FGSM is that it is not intended to be optimum, but rather fast. It is not designed to output the minimum required amount of perturbation. Furthermore, when compared to other attack types, the success ratio of FGSM is relatively low when applied with small ϵ values

2.1.2 Iterative gradient sign method

Kurakin et al. [16] proposed a minor but significant enhancement to the FGSM. Instead of taking one large step ϵ in the direction of the gradient sign, we take numerous smaller steps α and utilize the supplied value ϵ to clip the output in this method. This method is also known as the Basic Iterative Method (BIM), and it is simply FGSM applied to an input sample iteratively. Equation 2 describes how to generate perturbed images under the l_{inf} norm for a BIM attack.

$$\begin{aligned} \mathbf{x}_t^* &= \mathbf{x} \\ \mathbf{x}_{t+1}^* &= \text{clip}_{x,\epsilon}\{\mathbf{x}_t + \alpha \cdot \text{sign}\left(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}_t^*, y_{\text{true}})\right)\} \end{aligned} \tag{2}$$

where $\mathbf{x}$ is the clean sample input to the model, $\mathbf{x}^*$ is the output adversarial sample at ith iteration, J is the loss function of the model, θ denotes model parameters, y_{true} is the true label for the input, ϵ is a configurable parameter that limits maximum perturbation amount in given l_{inf} norm, and α is the step size.

The BIM attack has a better success rate than the FGSM [17]. The attacker can manage how far an adversarial sample is pushed further away from the decision boundary by configuring the ϵ parameter.

2.1.3 Projected gradient descent

This attack type, commonly known as PGD, has been proposed by Madry et al. [18]. It perturbs an input image $\mathbf{x}$ for a number of i iterations in the direction of the model's loss function gradient with a tiny step size. It projects the generated adversarial sample back onto the ϵ-ball of the input after each perturbation step depending on the chosen distance norm. In addition, rather than starting from the original point (ϵ = 0, in all the dimensions), PGD employs random start, which can be defined as:

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

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

2.1.4 DeepFool attack

This attack method has been introduced by Moosavi-Dezfooli et al. [19] and it is one of the strongest untargeted attack algorithms in literature. It is made to work with several distance norm metrics, including l_{inf} and l_2 norms.

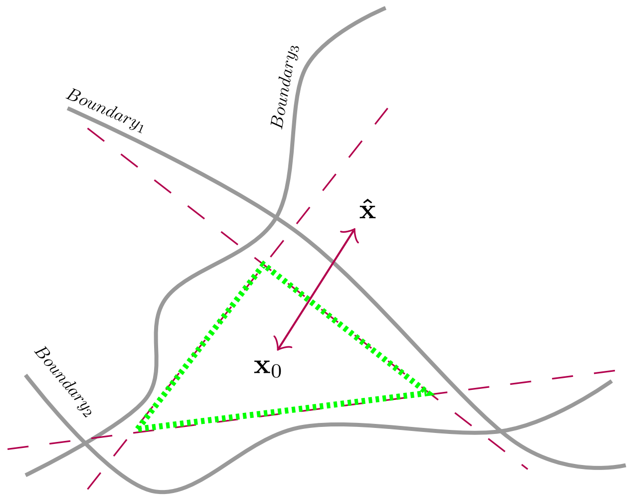


Fig. 2 Illustration of DeepFool attack algorithm

The DeepFool attack is formulated on the idea that neural network models act like linear classifiers with classes separated by a hyperplane. Starting with the initial input point $\mathbf{x_t}$, the algorithm determines the closest hyperplane and the smallest perturbation amount, which is the orthogonal projection to the hyperplane, at each iteration. The algorithm then computes $\mathbf{x}_{t+1}$ by adding the smallest perturbation to the $\mathbf{x}_t$ and checks for misclassification. The illustration of this attack algorithm is provided in Fig. 2. This attack can break defensive distillation method and achieves higher success rates than previously mentioned iterative attack approaches. But the downside of this attack algorithm is that the produced adversarial sample generally lies close to the decision boundary of the model.

2.1.5 Carlini and Wagner attack

The attack proposed by Carlini and Wagner [20] is one of the strongest attack algorithms so far. As a result, it is commonly used as a benchmark for the adversarial defense research groups, which tries to develop more robust DNN architectures that can withstand adversarial attacks. It is shown that, for the most well-known datasets, the CW attack has a greater success rate than the other attack types on normally trained models. Like DeepFool, it can also deceive defensively distilled models, which other attack types struggle to create adversarial examples for.

In order to generate more effective and strong adversarial samples under multiple l_p norms, the authors reformulate the attack as an optimization problem which may be solved using gradient descent. A $confidence$ parameter in the algorithm can used to change the level of prediction score for the created adversarial sample. For a normally trained model, application of CW attack with default setting (confidence set to 0) would generally yield to adversarial samples close to

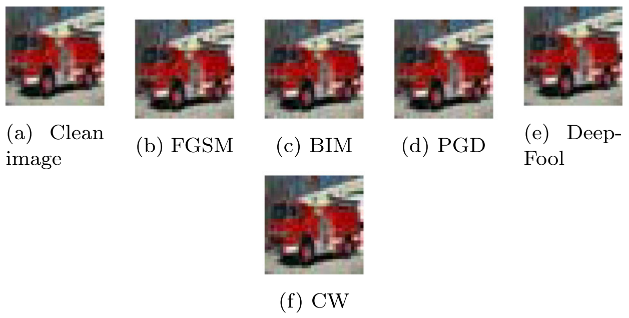


Fig. 3 An example image from CIFAR10 dataset and some of the adversarial samples crafted by using previously mentioned attack types

decision boundary. And high-confidence adversaries generally located further away from decision boundary.

Adversarial machine learning is a burgeoning field of research, and we see a lot of new adversarial attack algorithms being proposed. Some of the recent remarkable ones are as follows: (i) Square Attack [21] which is a query efficient black-box attack that is not based on model's gradient and can break defenses that utilize gradient masking, (ii) HopSkipJumpAttack [22] which is a decision-based attack algorithm based on an estimation of model's gradient direction and binary-search procedure for approaching the decision boundary, (iii) Prior Convictions [23] which utilizes two kinds of gradient estimation (time and data dependent priors) and propose a bandit optimization-based framework for adversarial sample generation under loss-only access black-box setting and (iv) Uncertainty-Based Attack [24] which utilizes both the model's loss function and quantified epistemic uncertainty to generate more powerful attacks. Figure 3 shows adversarial samples generated by attack algorithms discussed earlier.

2.2 Adversarial defense

In this section, we review some of the most notable adversarial defense methods proposed over the last few years.

2.2.1 Defensive distillation

Although the idea of knowledge distillation was previously introduced by Hinton et al. [25] to compress a large model into a smaller one, the utilization of this technique for adversarial defense purposes was first suggested by Papernot et al. [26]. The algorithm starts with training a *teacher model* on training data by employing a high temperature (T) value in the softmax function as in Eq. 4, where p_i is the probability of ith class and z_i's are the logits.

$$p_i = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_i}{T})} \tag{4}$$

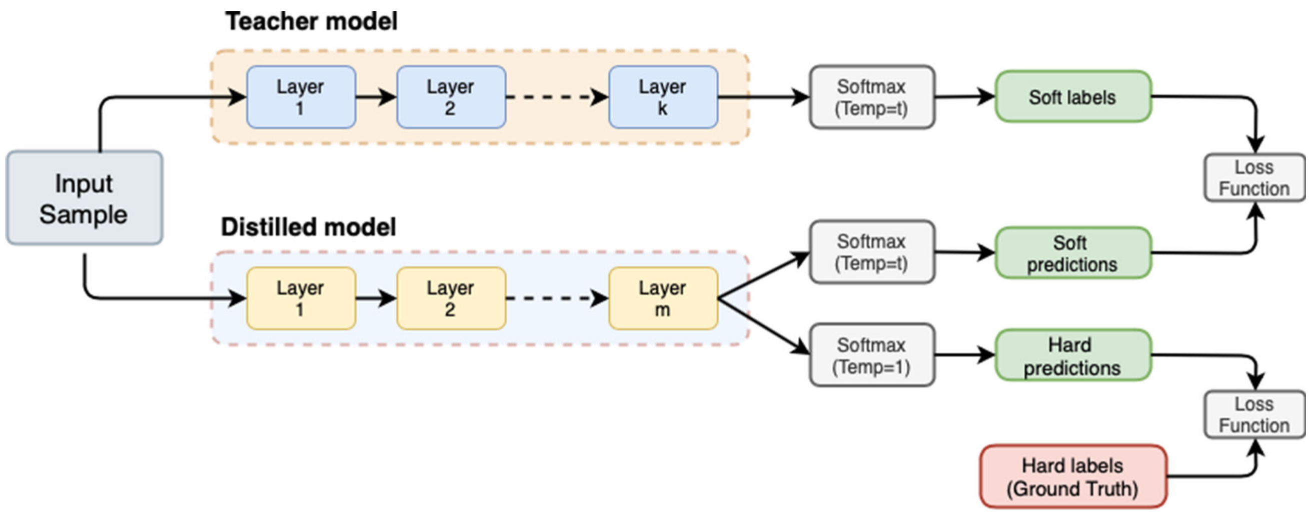


Fig. 4 Defensive distillation

Then, using the previously trained teacher model, each of the samples in the training data is labeled with soft labels calculated with temperature (T) in prediction time. The *distilled model* is then trained with the soft labels acquired from the teacher model, again with a high temperature (T) value in the softmax. When the training of the student model is over, we use temperature value as 1 during prediction time. Figure 4 shows the overall steps for this technique.

This technique was found to significantly reduce the ability of traditional gradient-based untargeted attacks to build adversarial samples. Because defense distillation has an effect of diminishing the gradients down to zero and the usage of standard objective function is not effective anymore. To illustrate this fact, we made a simple experiment using a test sample from MNIST (Digit) dataset and draw the loss surface of the normal and distilled models against two different directions (one for loss gradient direction and one for a random direction). As depicted in Fig. 5, the gradient of the distilled model diminishes to zero and thus loss-based attacks have difficulty in crafting adversarial samples for defensively distilled models. However, it was later demonstrated that more successful attack types, such as the CW and DeepFool attacks, could defeat the defensive distillation strategy. The reason why we opt to employ this technique in our approach is that, one can easily craft high confident examples near the decision boundary of a defensively distilled model. And also, due to gradient vanishing property, it is effective in defending against loss gradient-based untargeted attack types.

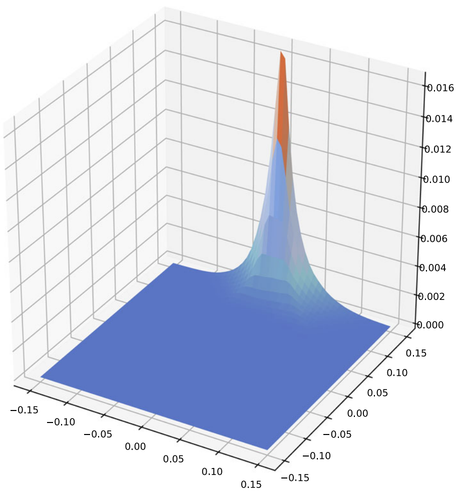
2.2.2 Adversarial training

Adversarial training is considered as an intuitive way of defensive strategy in which the robustness of the deep learner is strengthened by training it with adversarial samples. This strategy can be represented mathematically as a Minimax game, as shown in Eq. 5:

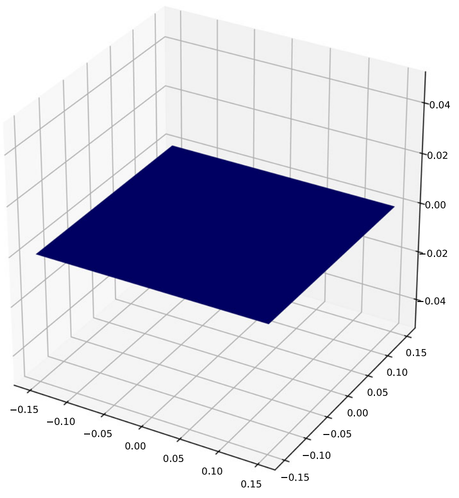
$$\min_{\theta} \max_{\|\delta\|\leq\epsilon} J(h_\theta(x+\delta), y) \tag{5}$$

where h denotes the model, J denotes the model's loss function, θ represents model's weights and y is the actual label. δ is the amount of perturbation amount added to input x and it is constrained by given ϵ value. The inner objective is maximized by employing the most powerful attack possible, which is often approximated by various adversarial attack types. In order to reduce the loss resulting from the inner maximization step, the outside minimization objective is used to train the model. This whole process produces a model that is expected to be resistant to adversarial attacks used during the training of the model. For adversarial training, Goodfellow et al. [15] used adversarial samples crafted by the FGSM attack. And Madry et al. used the PGD attack to build more robust models, but at the expense of consuming more computational resources. Despite the fact that adversarial training is often regarded as one of the most effective defenses against adversarial attacks, adversarially trained models are nevertheless vulnerable to attacks like CW.

Adversarial ML is a very active field of research, and new adversarial defense approaches are constantly being presented. Among the most notable are as follows: (i) High-Level Representation Guided Denoiser (HGD) [27] which avoids the error amplification effect of a traditional denoiser by utilizing the error in the upper layers of a DNN model as loss function and manages the training of a more efficient image denoiser, (ii) APE-GAN [28] which uses a Generative Adversarial Network (GAN) trained with adversarial samples to eliminate any adversarial perturbation of an input image, (iii) Certified Defense [29] which proposes a new



(a) normal model



(b) distilled model

Fig. 5 Loss surfaces of "normally trained" and "distilled" models

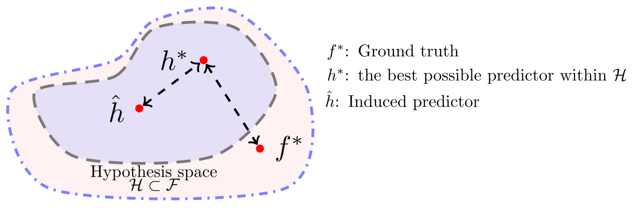

f^*: Ground truth
h^*: the best possible predictor within $\mathcal{H}$
$\hat{h}$: Induced predictor
h^*
$\hat{h}$
f^*
Hypothesis space
$\mathcal{H} \subset \mathcal{F}$

Fig. 6 Different types of epistemic uncertainty

differentiable upper bound yielding a model certificate ensuring that no attack can cause the error to exceed a specific value and (iv) [30] which uses several uncertainty metrics for detecting adversarial samples.

3 Preliminaries

Predictive models have traditionally been required to make decisions even in ambiguous cases where the model is unsure about its prediction. And this fact often leads to low-quality predictions. Assuming that the prediction of the model is always correct without considering the model's uncertainty can have disastrous consequences. This led the researcher study developing different methods for uncertainty quantification in an attempt to improve model reliability.

We begin this part by discussing the main types of uncertainty in ML. Then, we go over how different uncertainty metrics can be quantified.

3.1 Uncertainty in machine learning

In ML, there are two main kinds of uncertainty: aleatoric and epistemic uncertainty [31–33]. And recently, apart from these main types, a new uncertainty metric named scibilic uncertainty has been introduced.

3.1.1 Epistemic uncertainty

Uncertainty due to an inadequate knowledge and limited data required for a perfect predictor is referred to as Epistemic uncertainty [34]. As shown in Fig. 6, it can be classified as: *approximation uncertainty* and *model uncertainty*.
Approximation Uncertainty

In a traditional ML task, the learner is provided with data points from a dataset that is independent and identically distributed. Then, the learner attempts to induce a hypothesis $\hat{h}$ from hypothesis space $\mathcal{H}$ by selecting an appropriate learning method with its associated hyper-parameters and minimizing the expected loss (risk) with a chosen loss function, ℓ. Nevertheless, what the learner actually does is to try to keep *empirical risk* R_{emp} as low as possible, which is an estimation of real risk $R(h)$. The induced $\hat{h}$ represents approximation to the h^* which is the the real risk minimizer and best possible hypothesis within $\mathcal{H}$. This leads to an approximation uncertainty. As a result, the quality of the induced hypothesis is not ideal, and the trained model will be prone to errors.
Model Uncertainty

Assume that the perfect predictor is not included in the hypothesis space H. In that situation, the learner has no possibility of developing a hypothesis function that can effectively map all potential inputs to outputs. This results in a discrepancy between the ground truth f^* and the best possible function h^* within $\mathcal{H}$, which is referred to as model uncertainty.

The Universal Approximation Theorem, on the other hand, showed us that any target function f can be approxi-

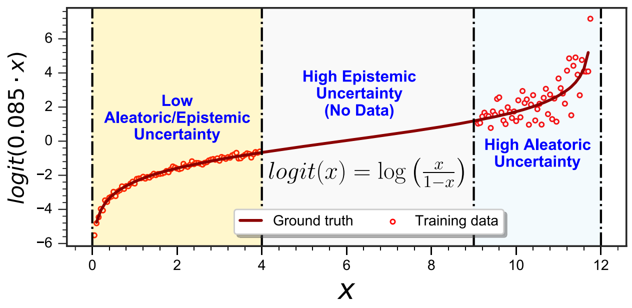


Fig. 7 Illustration of the Epistemic and Aleatoric uncertainty

mated by a neural network [35,36]. For deep neural networks, the hypothesis space $\mathcal{H}$ can be extremely large. Hence, it is reasonable to presume that $h^* = f^*$. The model uncertainty can be neglected in deep neural networks, leaving only the approximation uncertainty to be considered. As a result, the actual source of epistemic uncertainty in deep learning tasks is related with approximation uncertainty. Epistemic uncertainty is referred to the confidence a model has about its prediction [37]. The fundamental cause is the uncertainty regarding the model's parameters. This form of uncertainty is visible in areas where we have inadequate training data and the model weights are not properly tuned.

3.1.2 Aleatoric uncertainty

Aleatoric uncertainty relates to the variation in an experiment's outcome caused by inherent random effects [38]. Despite having adequate training examples, this form of uncertainty cannot be reduced [39]. The noise observed in a sensor's measurement data is an excellent example of this phenomena.

A simple nonlinear function (logit$(0.085 \times x)$ in the interval $x \in [0, 12]$) is presented in Fig. 7. Noisy samples are illustrated in the region at right where $9 < x < 12$, and those samples lead to high aleatoric uncertainty. These points, for example, could reflect an erroneous sensor measurement; one can deduce that the sensor generates errors around $x = 10.5$ for some unknown inherent reason. We can also argue that the figure's central regions represent areas of high epistemic uncertainty. Because our model does not have enough training examples to accurately represent the data.

3.1.3 Scibilic uncertainty

Reinhold et al. [40] proposed a new sort of uncertainty named scibilic uncertainty by combining epistemic and aleatoric uncertainty. This new uncertainty metric was employed in an image segmentation challenge to identify areas in an input image that the model could resolve how to predict if it was given enough data to train with. After quantifying epistemic and aleatoric uncertainty, we can compute scibilic uncertainty by dividing the former by the latter. The intuition behind scilibilic uncertainty is as follows: For a suspicious input, a DNN model trained on naturally occurring data may result in high epistemic uncertainty. Nevertheless, due to some intrinsic property of the data, the model can lead to significant aleatoric uncertainty for that same input, making it difficult to make a reliable prediction. The division procedure allows us to keep epistemic uncertainty that is not caused by the model's difficulty for that particular input.

3.2 Quantifying uncertainty in deep neural networks

Numerous research studies have been conducted in recent years to quantify uncertainty in DNN models. The majority of these studies relied on Bayesian NNs, which quantify predictive uncertainty by learning the posterior distribution over the weights. However, Bayesian NNs have an extra computing overhead and an inference problem. As a result, a number of approximations to Bayesian approaches have been proposed which employ variational inference [41–44]. Lakshminarayanan et al. [45], on the other hand, adopted deep ensemble approach for uncertainty quantification as an alternative to Bayesian Neural Networks. However, this method involves training of many models, which may be impractical in practice. Gal et al. [46] proposed a more elegant and efficient technique and demonstrated that an NN model with inference time dropout corresponds to a Bayesian approximation of the Gaussian process. Their approach functions as an ensemble model in training mode (during prediction time) of the model and dropout is therefore enabled. In each individual ensemble model, the system drops out part of the neurons in each layer of the network based on the dropout ratio. The variance of the MC dropout sampling output throughout prediction time is used to approximate the overall epistemic uncertainty. Later, Kendall and Gal [47] presented a technique in which both epistemic and aleatoric uncertainties are captured in a single model. They employed a CNN Model f (Bayesian NN) with weights represented by $\hat{\omega}$ that maps an input x to $\hat{y}$ and σ^2. In their approach, the model output is divided into two parts as predictive mean $(\hat{y})$ and predicted variance $\hat{\sigma}^2$ terms. Consequently, the two types of uncertainty are quantified as follows:

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T}\operatorname{diag}(\hat{\sigma}^2)}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}-\bar{y})^{\otimes 2}}_{\text{epistemic}} \tag{6}$$

the number of MC Dropout samples in prediction time when the model is in training mode, $\bar{y} = \sum_{t=1}^{T}\hat{y}_t/T$ and $y^{\otimes 2} = yy^T$

The method described above is delicate and has been demonstrated to be effective in computer vision applications such as image segmentation. Unfortunately, since the output of the model is divided into two parts for predicting mean and variance terms, it was inconvenient to employ in adversarial machine learning trials for us. We had to look for other options, since the attack algorithms are developed to function with model architectures with only prediction output term (no variance).

The method we employed in this study is proposed by Kwon et al. [48] as an alternate approach for quantifying both epistemic and aleatoric uncertainty in classification models. In the author's method, the variance of the prediction is comprised of two parts that represent aleatoric and epistemic uncertainty. Let $\hat{\omega}$ be the trained weights used in the neural network, K denotes the number of output classes and $p(y^*|x^*, \hat{\omega})$ denotes the prediction y^* of a model for any test sample x^* given the weights of the model where $y^* \in \mathbb{R}^k$, then the following is the formula for their method:

$$\mathrm{Var}_{p(y^*|x^*,\omega)}(y^*) = \mathbb{E}_{p(y^*|x^*,\omega)}(y^{*\otimes 2}) - \mathbb{E}_{p(y^*|x^*,\omega)}(y^*)^{\otimes 2} \tag{7}$$

$$= \underbrace{\frac{1}{T}\sum_{t=1}^{T}[\mathrm{diag}\{p(y^*|x^*,\hat{\omega}_t)\} - p(y^*|x^*,\hat{\omega}_t)^{\otimes 2}]}_{\text{aleatoric}} \tag{8}$$

$$+ \underbrace{\frac{1}{T}\sum_{t=1}^{T}\{p(y^*|x^*,\hat{\omega}_t)\} - \hat{p}(y^*|x^*,\hat{\omega}_t)^{\otimes 2}}_{\text{epistemic}} \tag{9}$$

where $\hat{p}(y^*|x^*, \hat{\omega}_t) = \sum_{t=1}^{T}\{p(y^*|x^*, \hat{\omega}_t)\}$

Both Eqs. (8, 9) produce a $k \times k$ matrix with diagonal elements representing the variance of each output class.

After we calculate epistemic and aleatoric uncertainty, we may simply compute scibilic uncertainty as follows:

$$\text{Scibilic} = \frac{\text{Epistemic}}{\text{Aleatoric}} \tag{10}$$

Eventually, for a given input $\mathbf{x}^*$, we have three different column vectors of shape $K \times 1$ as $EP \in \mathbb{R}^k$, $AL \in \mathbb{R}^k$, $SC \in \mathbb{R}^k$, whose elements represent epistemic, aleatoric and scibilic uncertainty for each class respectively.

4 Approach

In regions with a low number of training samples, model uncertainty is larger. We cannot obtain a model that perfectly predict all testing data. This can be explained due to the absence of ground truth in these areas. Figure 8 displays the prediction outputs of a regression model trained on a small amount of data that are bound by some interval. For this toy example, we trained a neural network with single hidden layer and ten neurons to learn a linear function $y = 2 \times x + 3$. As can be observed in the figure, the model's uncertainty values (epistemic) derived from MC dropout estimates are high in places where we do not have training data, indicating that the quality of the prediction is low and the model is having difficulty deciding the accurate output values. Consistently, high loss values are observed in those regions. As a result, we can argue that the regions with high epistemic uncertainty corresponds to the regions of low prediction accuracy. Therefore, testing the model in severe settings with input that it has never encountered before will lead to model prediction failure [24]. Similarly, restoring the input samples to the regions where the model was trained on (low uncertainty regions) would yield more accurate predictions. In this study, we employed this idea. However, we paid attention to one key point, that is, while trying to minimize the quantified uncertainty for any input sample, we made sure that the restoration operation has minimal effect to model loss.

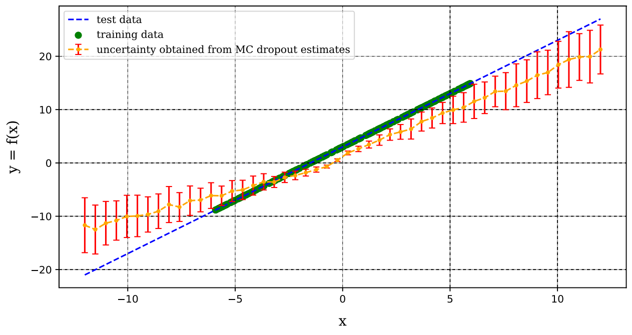


Fig. 8 Uncertainty values obtained from a regression model

4.1 Uncertainty-based reversal operation

We begin this section by presenting the pseudo-code for uncertainty-based reversal procedure, as described in Algorithm 1. This reversal method is designed under L_∞ norm.

We compute $\nabla_{\mathbf{x}}\ell(h(\mathbf{x}_t, y_{pred}))$ and $\nabla_{\mathbf{x}}U(\mathbf{x}_t, h, p, T)$ for each iteration of our uncertainty-based reversal procedure. Then, we restore the input sample by minimizing its quantified uncertainty and utilize the sub-directions of uncertainty's gradient that are not shared by the loss' gradient with respect to predicted class. A better understanding of this idea can be obtained by glancing at Fig. 9.

In our proposed method, we used both loss and uncertainty information and exclusively use the sub-directions from the uncertainty's gradient that are not shared by the gradient of the loss. The intuition behind this approach is as follows: In a conventional production setting where an ML model is

Algorithm 1: $\mathbf{x}$ is the input image, y_{pred} is the predicted label for $\mathbf{x}$, h is the learnt hypothesis function, p is the dropout ratio of the model used in dropout layers, T is the number of MC dropout samples at prediction time in model training mode, N is the number of iterations, ϵ is the maximum perturbation amount allowed, α is 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
           $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T) == \nabla_{\mathbf{x}} \ell(h(\mathbf{x}_t, y_{pred}))$
           /* update X by using below formula */
11         $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\epsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T))$
12 return $\mathbf{x}_{t+1}$
```

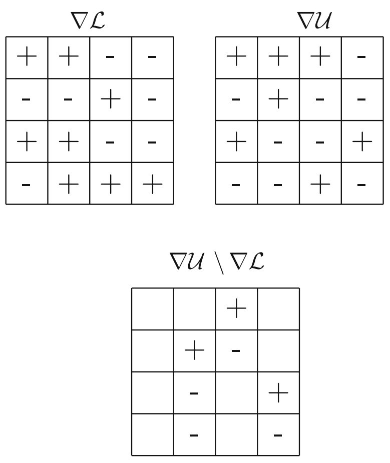


Fig. 9 Sub-directions used in reversal procedure

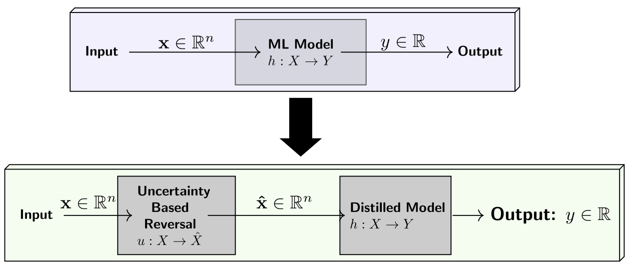


Fig. 10 Options for ML model deployment

employed for a classification problem, the input is supplied to the model, and the final prediction is observed after the input sample is processed and mapped to an output, as illustrated in upper part of Fig. 10. For any input, the gradient of the loss against predicted label gives us an idea about the possible direction where we can minimize the loss. However, if the prediction of the ML model is wrong, the model will be more confident in its wrong prediction and final prediction will be much more inaccurate when we perturb the image in loss gradient direction. Therefore, when trying to minimize the quantified uncertainty of the input sample, we needed to get rid of the common sub-directions shared by loss gradient. After rejecting part of the sub-directions, the remaining sub-directions in the uncertainty's gradient can be utilized to safely return the input to its original data manifold.

In this study, we have used both the standard version which is based on epistemic uncertainty and developed 2 additional variants of above procedure which are different in terms of the type of the uncertainty metric employed and the way of using the output uncertainty vector. We started our experiments by using epistemic uncertainty obtained from Eq. 9. We used the expected value(mean) of the epistemic uncertainty (EP) for the uncertainty quantification as in the case of [9]. Then, we tried scibilic uncertainty (SC) via Eq. 10 and used mean of the SC. Lastly, instead of using the average scibilic uncertainty measure of all classes, we used the uncertainty value of the predicted class only. In this way, we used the following three equations for uncertainty quantification.

$$U(\mathbf{x}_t, h, p, T) = \frac{1}{K} \sum_{k=1}^{K} EP[k] \tag{11}$$

$$U(\mathbf{x}_t, h, p, T) = \frac{1}{K} \sum_{k=1}^{K} SC[k] \tag{12}$$

$$U(\mathbf{x}_t, h, p, T) = SC[pred] \tag{13}$$

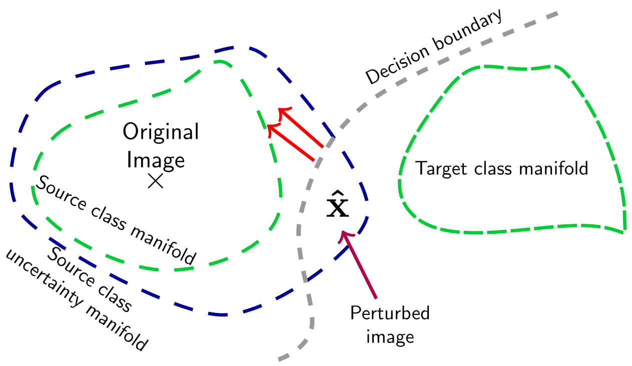


Fig. 11 Restoring perturbed image back to its original class data manifold

4.2 Analysis of the uncertainty-based reversal method

This technique can be applied as a reverse-perturbation operation before feeding any input into a classification model. As seen in the bottom section of Fig. 10, an input X that is intended to be presented to the ML model is first processed by uncertainty-based reversal procedure. The goal of this reverse-perturbation operation is to judiciously perturb the input image in a way that reduces its quantified uncertainty. This "slightly reversed" image $\hat{X}$ will then be fed into the ML model. Figure 11 illustrates the uncertainty-based reversal process. The crucial thing is that the location of the input sample should not be too far away (on the incorrect side) from the decision boundary of the model to ensure a successful reversal operation. And, this is the major drawback of the standard uncertainty-based reversal operation. For DeepFool attack and CW attack with confidence parameter set to 0, the perturbed samples generally resides close to the decision boundary. However, if one applies CW attack by setting the confidence parameter to a high value, the attack algorithm will generally craft high confident adversarial samples far away from decision boundary. That is why, for a "normally trained" model, the success rate of standard reversal operation will be lower.

However, defense distillation technique can help us to overcome this problem. Because, during the training of a distilled network, what we actually do is to force the model to learn making high-confident predictions. And therefore, during prediction time, we see that the distilled models mostly make high-confident predictions in favor of the predicted class no matter where the input resides in its own data manifold [20]. And this is valid even if the test sample lies near the vicinity of model's decision boundary (whether it is in the correct or wrong side). This way, even if the attacker sets a high value for the confidence parameter for CW attack, the algorithm can easily craft adversarial sample near decision boundaries. Thanks to this, reversal procedure can successfully restore the input back to its original data manifold. To demonstrate this phenomenon, we made an experiment by using two different models as normal and defensively distilled (student) model. For each of these models: we used the same random sample from CIFAR10 dataset and applied DeepFool attack on it for generating adversarial samples. Then, we applied uncertainty-based reversal procedure on these perturbed samples and get the restored images. For each of the input, perturbed and restored samples, we have also shown the softmax output scores of the normal and student models used as illustrated in Fig. 12. The attack algorithm and our reversal procedure variants are successful on both of the normal and distilled models. When we check the softmax output scores of the normal model for perturbed sample in the first scenario, we see that there is not much difference between the prediction scores of the correct and wrong class. However, we observe that the distilled model makes its prediction in favor of the predicted class with a very high confidence. We know that DeepFool attack results in adversarial samples close to decision boundaries, this experiment verifies our intuition of using a defensively distilled model together with uncertainty-based reversal procedure to force most of the successful adversarial samples to reside near the decision boundary.

Of course, for any kind of procedure that is planned to be applied on the input samples of a deployed model, a significant issue to consider is that this process should not have highly negative impact on the model's performance on clean data. Any modification to the model's functioning that reduces prediction accuracy below an acceptable level

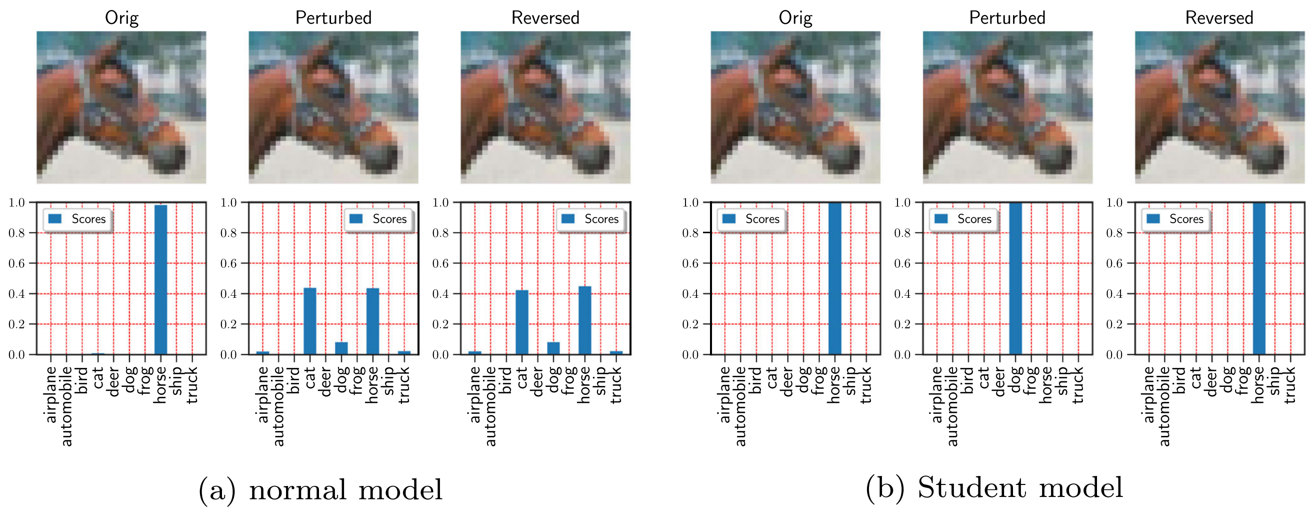


Fig. 12 The effect of uncertainty-based reversal procedure on the predictions of normally trained and distilled models

can not be permitted, regardless of how much robustness it delivers. We conducted comprehensive testing to determine the impact of uncertainty-based reversal procedure on the model's clean data performance and confirmed that the accuracy rate did not decline more than a tolerable level, as shown in the experiments section. The results show that this technique can be used to strengthen the robustness of the deployed ML models against malicious attacks, especially in risky environments where security is an important concern.

4.3 Adversarial assumptions

In this work, we assume that the main objective of the adversary is to obtain the desired behavior for an ML model and the criteria for success for the attacker are tied directly with "any" labeling mistake. This type of attack strategy is classified in the literature as *untargeted-attack*, in which the attacker is considered successful if, for instance, the rifle image is predicted to be anything other than a rifle. Our assumption was that the attacker was fully aware of the architecture and parameters of the target model as in the case of *whitebox* setting. Another crucial assumption concerns the constraints of the attacker. Clearly, the attacker should be limited to applying a perturbation with l_p norm up to certain ϵ value for an attack to be unrecognizable to the human eye. To ensure this modification to be imperceptible, the attacker must find an approximate solution to a difficult constraint optimization problem and identify which areas of the input should be modified. The adversary tries to decrease the classification performance of the target network as much as possible by employing any of the known attack algorithms like [15,16,18,49]. For this study, we used l_∞ and l_2 norm metrics to restrict the maximum perturbation amount that an adversary can apply on the input sample. Finally, the error rate of our proposed defense technique is assessed over the percentage of resulting successful attack samples which is proposed by Goodfellow et al. [15] and recommended by Carlini et al. [50].

5 Results

5.1 Experimental setup

For our experiments, we used two sets of models as normal and distilled(student) by using same architectures and trained our CNN models using MNIST (Digit) [51], MNIST (Fashion) [52] and CIFAR-10 [53] datasets. In the first group, our *normally trained* models attained accuracy rates of 99.11%, 92.61%, and 79.38%, whereas, in the second group, our *distilled models* attained accuracy rates of 99.41%, 92.62%, and 80.47%. The architectures of our CNN models and the hyper-parameters used in model training are listed in Table 1 and 2. Lastly, for quantifying uncertainty metrics, we set $T = 50$ as the number of MC dropout samples.

Table 1 CNN architectures for normal and distilled models

Dataset	Layer type	Layer information
M.—Digit	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 64$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 64$
	Max Pooling	2×2
	Fully Connected + ReLU	3136×200
	Dropout	p: 0.5
	Fully Connected + ReLU	200×200
	Dropout	p: 0.5
	Fully Connected + ReLU	200×10
M.—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: 0.25
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p: 0.25
	Fully Connected + ReLU	3136×600
	Dropout	p: 0.25
	Fully Connected + ReLU	600×120
	Fully Connected + ReLU	120×10
CIFAR10	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 32$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Max Pooling (Stride 2)	2×2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 128$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 128$
	Max Pooling (Stride 2)	2×2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Dropout	p: 0.5
	Max Pooling (Stride 2)	2×2
	Fully Connected + ReLU	4096×1024
	Dropout	p: 0.5
	Fully Connected + ReLU	1024×256
	Dropout	p: 0.5
	Fully Connected + ReLU	256×10

5.2 Experimental results

Throughout our experiments, we applied attack on the test samples only if they were previously classified correctly by our models. Because, an attacker would obviously have no

Table 2 CNN model parameters

Parameters	MNIST (Digit)		MNIST (Fashion)		CIFAR-10	
	Normal	Distilled	Normal	Distilled	Normal	Distilled
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
Batch size	128	128	128	128	128	128
Dropout ratio	0.5	0.5	0.25	0.25	0.5	0.5
# of Epochs	10	30	30	50	30	50
Temperature	1	20	1	100	1	100

Table 3 Parameters that are used in our uncertainty-based reversal process: α denotes the step size and i denotes # of reversal steps for a perturbation budget ϵ

Dataset	Parameters
MNIST Digit	$\epsilon = 0.02$, $\alpha = \epsilon \cdot 0.2$, i = 10
MNIST Fashion	$\epsilon = 0.006$, $\alpha = \epsilon \cdot 0.2$, i = 20
CIFAR10	$\epsilon = 0.2/255$, $\alpha = \epsilon \cdot 0.2$, i = 10

motivation to perturbed samples that have already been misclassified. We utilized an open source Python library called Foolbox [54] to implement the attacks used in this study .[2].

[2] We use Torch 1.13.0 to implement the attacks and the proposed defense method in a computer with processor Intel Core i5-1145G7 2.6 GHz and Windows 10 OS.

We started our experiments by first evaluating contributions of different uncertainty metrics on uncertainty-based reversal procedure performance. To do this, we used normal CNN models which are trained on MNIST (Digit) and CIFAR10 Datasets, and we applied several different attack types on each sample to craft their adversarial counterparts. We then tried to restore those adversarial samples back to their original class manifolds by using each of the three variants of reversal procedure. Table 3 summarizes the values of the parameters that are used in the reversal procedure.

The results of the defense method variants are provided in Table 4 and 5. As can be seen from the final attack success rates, best robustness performance is achieved when we used the Scibilic uncertainty value of the predicted class only. When we check Table 4, we also observe a considerable dif-

Table 4 Attack success rates of normally trained model on MNIST (Digit) dataset with and without uncertainty-based reversal procedure

	Epistemic (mean)		Scibilic (mean)		Scibilic (pred)	
	w/o rev (%)	w rev (%)	w/o rev (%)	w rev (%)	w/o rev (%)	w rev (%)
FGSM (l_∞, ϵ: 0.1)	13.25	9.04	13.25	6.41	13.25	5.82
BIM (l_∞, ϵ: 0.1)	38.94	31.47	38.94	20.48	38.94	18.16
PGD (l_∞, ϵ: 0.1)	35.32	27.34	35.32	18.64	35.32	16.68
DeepFool (l_∞, ϵ: 0.1)	21.60	0.07	21.60	0.01	21.60	0.01
DeepFool (l_2, ϵ: 0.1)	26.78	1.07	26.78	0.20	26.78	0.11
CW (l_2, ϵ: 1.35 c : 0)	67.71	0.00	67.69	0.02	67.63	0.01
CW (l_2, ϵ: 1.35 c : 40)	21.15	16.87	21.21	7.44	21.33	7.16

Table 5 Attack success rates of normally trained model on CIFAR10 dataset with and without uncertainty-based reversal procedure

	Epistemic (mean)		Scibilic (mean)		Scibilic (pred)	
	w/o rev (%)	w rev (%)	w/o rev (%)	w rev (%)	w/o rev (%)	w rev (%)
FGSM (l_∞, $\epsilon = 4/255$)	62.91	60.23	62.91	59.58	62.91	59.29
BIM (l_∞, $\epsilon = 4/255$)	77.92	77.67	77.92	77.64	77.92	77.61
PGD (l_∞, $\epsilon = 4/255$)	76.36	75.98	76.36	75.92	76.36	75.89
DeepFool (l_∞, $\epsilon = 4/255$)	76.23	1.21	76.23	0.65	76.23	0.28
DeepFool (l_2, $\epsilon = 0.42$)	73.73	0.59	73.73	0.31	73.73	0.12
CW (l_2, $\epsilon = 0.42$ c : 0)	78.26	5.50	78.26	4.57	78.26	4.15
CW (l_2, $\epsilon = 0.42$ c : 10)	74.39	73.44	74.39	72.98	74.39	72.50

Table 6 Effect of reversal procedure on clean performance of normally trained model—MNIST (Digit) Dataset

	Without reversal (%)	With reversal (%)
Epistemic Unc. (mean) Based Reversal	99.11	99.00
Scibilic Unc. (mean) Based Reversal	99.11	98.95
Scibilic Unc. (pred class) Based Reversal	99.11	98.92

Table 7 Attack success rates of distilled model on MNIST (Digit) dataset with and without uncertainty-based reversal procedure

	Scibilic Unc. Based Rev. (pred)	
	Without reversal (%)	With reversal (%)
FGSM (l_∞, ϵ: 0.1)	1.01	0.91
BIM (l_∞, ϵ: 0.1)	1.04	1.03
PGD (l_∞, ϵ: 0.1)	1.10	1.09
DeepFool (l_∞, ϵ: 0.1)	15.95	0.00
DeepFool (l_2, ϵ: 0.1)	26.12	0.12
CW (l_2, ϵ: 1.35 conf: 0)	75.57	0.01
CW (l_2, ϵ: 1.35 conf: 40)	75.43	0.76

Table 8 Effect of reversal procedure on clean performance of distilled model—MNIST (Digit) dataset

	Without reversal (%)	With reversal (%)
Scibilic Unc. Based Reversal (pred class)	99.41	99.10

Table 9 Attack success rates on MNIST (Fashion) dataset with and without uncertainty-based reversal procedure

		Attack success	
		w/o rev (%)	with rev (%)
Normal model	FGSM ($\epsilon = 0.03$)	42.77	31.16
	BIM (l_∞, $\epsilon = 0.03$)	71.79	61.29
	PGD (l_∞, $\epsilon = 0.03$)	68.15	58.04
	DeepFool (l_∞, $\epsilon = 0.03$)	59.96	0.02
	DeepFool (l_2, $\epsilon = 0.403$)	57.72	0.00
	CW (l_2, $\epsilon = 0.403$, conf. = 0)	75.36	0.01
	CW (l_2, $\epsilon = 0.403$, conf. = 10)	66.65	49.08
Distilled model	FGSM ($\epsilon = 0.03$)	2.48	2.35
	BIM (l_∞, $\epsilon = 0.03$)	2.57	2.57
	PGD (l_∞, $\epsilon = 0.03$)	3.44	3.44
	DeepFool (l_∞, $\epsilon = 0.03$)	72.45	0.02
	DeepFool (l_2, $\epsilon = 0.403$)	69.47	0.06
	CW (l_2, $\epsilon = 0.403$, conf. = 0)	83.61	0.00
	CW (l_2, $\epsilon = 0.403$, conf. = 10)	83.41	0.13

ference between the reversal performances of Scibilic and Epistemic Uncertainty (standard uncertainty metric used in the initial proposal of reversal procedure). For instance: in the case of BIM attack, attack success rates drop from 31.47 to 20.48% if we switch from Epistemic Uncertainty (Eq. 11) to Scibilic Uncertainty (Eq. 12). And instead of using the mean of Scibilic Uncertainty vector, if we use the uncertainty value of the predicted class only (Eq. 13), we can even lower the final attack success rate to 18.16%. We also observe that the difference between the reversal performances of each uncertainty metric is less clear as the complexity and the dimensions of the used dataset increases.

Table 10 Effect of reversal procedure on clean performance—MNIST (Fashion) dataset

	Without reversal (%)	With reversal (%)
Normal model	92.61	90.18
Distilled model	92.62	90.25

Table 11 Attack success rates on CIFAR-10 dataset with and without uncertainty-based reversal procedure

		Attack success	
		w/o rev (%)	with rev (%)
Normal model	FGSM ($\epsilon = 4/255$)	62.91	59.29
	BIM ($l_\infty, \epsilon = 4/255$)	77.92	77.61
	PGD ($l_\infty, \epsilon = 4/255$)	76.36	75.89
	DeepFool ($l_\infty, \epsilon = 4/255$)	76.23	0.28
	DeepFool ($l_2, \epsilon = 0.42$)	73.73	0.12
	CW ($l_2, \epsilon = 0.42$, conf. = 0)	78.26	4.15
	CW ($l_2, \epsilon = 0.42$, conf. = 10)	74.39	72.50
Distilled model	FGSM ($\epsilon = 4/255$)	4.21	4.16
	BIM ($l_\infty, \epsilon = 4/255$)	4.24	4.24
	PGD ($l_\infty, \epsilon = 4/255$)	4.58	4.58
	DeepFool ($l_\infty, \epsilon = 4/255$)	77.94	0.64
	DeepFool ($l_2, \epsilon = 0.42$)	74.95	0.23
	CW ($l_2, \epsilon = 0.42$, conf. = 0)	79.71	0.16
	CW ($l_2, \epsilon = 0.42$, conf. = 10)	79.73	1.87

We then checked the effect of reversal procedure on clean data performance. For this purpose, we applied reversal strategy (the standard one with epistemic uncertainty and our 2 variants separately) directly to each of the test samples of MNIST (Digit) dataset. And we compared the resulting model accuracy values with the ones we obtained without any reversal operation. As shown in Table 6, reversal strategy has only a minimal and tolerable impact on model classification performance. Considering the level of robustness it provides, we can thus infer that the use of uncertainty-based reversal strategy has no detrimental impact on overall. We also do not observe a noticeable difference between the impact of our variants on clean data classification performance. Therefore, we choose the usage of our second variant (Scibilic Unc. with pred. class) as our base metric in our reversal strategy and the rest of the experiments are conducted using this.

Although the reversal procedure is performing very well on certain attack types like DeepFool or CW attack (when confidence parameter set to a low value), for other loss-based attacks like FGSM, BIM or PGD, we still face some problems. The same is valid if we opt to use CW attack by setting confidence parameter to a high value during attack implementation. The reason is that, for those cases, the resulting adversarial samples generally lie far from the decision boundary of the model. To mitigate this problem, we employed another method known as defensive distillation. Distillation technique has an effect of diminishing the gradients of the model down to almost zero and also force the model to make its predictions much more confidently. The former effect of distillation prohibits loss-based untargeted attacks to use gradients efficiently and results in considerably lower attack success rates. And the latter effect of distillation results in high confidence adversarial samples located close to decision

Table 12 Effect of reversal procedure on clean performance—CIFAR10 dataset

	Without reversal (%)	With reversal (%)
Normal Model	79.38	77.84
Distilled Model	80.47	79.45

Table 13 BPDA attack success rates

	Normal model (%)	TENET architecture (%)
PGD $l_\infty, \epsilon = 0.1$	37.06	1.82
BIM $l_\infty, \epsilon = 0.1$	35.34	1.02
FGSM $l_\infty, \epsilon = 0.1$	13.17	0.89

boundary. Therefore, when we combined reversal procedure with defensive distillation, we achieved much better results. The results in Table 7 show that our proposed architecture (TENET) provides perfect robustness to all kinds of used attacks and reduces the attack success rates down to 1% regardless of the attack algorithm with only a negligible effect on clean data classification performance (Table 8).

To evaluate and validate the effectiveness of our proposed architecture, we have conducted additional experiments on different datasets. Table 9 shows the performance of reversal procedure on MNIST (Fashion) dataset for both *normal* and *distilled* models.

And Table 10 shows the effect of our proposed architecture on clean data classification performance.

Finally, we have performed the same set of experiments on CIFAR-10 dataset. The results are available in Table 11 and Table 12. The results of our detailed experiments on all the datasets reveal that our reversal procedure and defensive

Table 14 Attack success rates on a normally trained Fashion MNIST model—Algorithm comparison

	w/o rev (%)	with rev. via Algorithm 1 (primitive) (%)	with rev. via Algorithm 1 (%)
DeepFool (l_∞)	59.96	9.98	0.03
DeepFool (l_2)	57.72	12.90	0.01
CW (l_2, conf.=0)	75.36	8.19	0.01

Table 15 Comparison of attack success rates with TENET and Adversarial Training

	MNIST DIGIT		MNIST FASHION		CIFAR10	
	Adv. Training (%)	TENET (%)	Adv. Training (%)	TENET (%)	Adv. Training (%)	TENET (%)
FGSM (l_∞)	1.19	0.91	4.76	2.35	16.02	4.16
BIM (l_∞)	1.28	1.03	5.68	2.57	18.83	4.24
PGD (l_∞)	1.09	1.09	5.02	3.54	16.82	4.58
DeepFool (l_∞)	1.23	0.00	5.19	0.02	19.74	0.64
DeepFool (l_2)	3.78	0.12	7.03	0.06	13.93	0.23
CW (c=0, l_2)	10.23	0.01	9,30	0,00	26,59	0.16
CW (c=10, l_2)	6.06	0.76	3.49	0.13	11.47	1.87

Table 16 Attack success rates on normally trained VGG19 model and VGG19 with TENET architecture

	VGG19 Normal Model (%)	VGG19 TENET (%)
FGSM (l_∞)	65.56	7.04
BIM (l_∞)	79.62	8.94
PGD (l_∞)	79.26	8.93
DeepFool (l_∞)	89.68	1.34
DeepFool (l_2)	84.77	0.88
CW (c=0, l_2)	97.32	0.11
CW (c=10, l_2)	94.98	0.62

distillation technique can handle complimentary situations and together they provide very high degree of robustness against various kinds of untargeted attacks.

In the last part of our experiments, we wanted to test our defense method against an adaptive attack idea. For this purpose, we tried to compare the robustness of a normal model and our proposed architecture which both have non-differentiable components that are obscuring the gradients from the attacker. In this scenario, for attacking the target models, we used Backward Pass Differentiable Approximation (BPDA) approach via Advertorch Toolbox [55] and replaced the non-differentiable components (bits squeezing, median filter) with identity function in the backward pass as suggested by Athalye et al. [56]. The results that are available in Table 13 show that adaptive attack ideas like BPDA might be successful against a defense approach which obscures the gradients from the attacker. However, if the same gradient masking-based defense approach was applied to our TENET architecture, BPDA attack idea would not be successful. The main reason behind the robustness of our proposed architecture against BPDA is that our method involves a defensive distillation step which forces gradients of the model to zero for any gradient-based untargeted attack [57]. Hence, even if the attacker tries to circumvent the defense by using an approximate function in the backward pass, the computed gradients will still be useless for crafting successful adversarial perturbation as it is a defensively distilled model.

5.3 Discussions and further results

We begin this part by showing the positive effect of getting rid of the common directions which are shared by loss and uncertainty's gradient from uncertainty's gradient. Results available in Table 14 show the attack success rates of DeepFool and CW attacks against normal prediction and our proposed defense method (with and without eliminating the common directions). For this experiment, we call the version of Algorithm-1 which does not discard the common directions as "primitive" (omitting line 10 in Algorithm-1). As can be seen from the results, we can substantially increase the defensive performance once we discard the common directions. Because, for the perturbed input samples that are pushed away from their decision boundaries and thus, which are already classified wrongly, the gradient of the loss with respect to the predicted label will point to the wrong class data manifold. Therefore, these sub-directions have a negative impact on reverting the input sample back to its own data manifold.

We then wanted to compare the performance of our proposed defense method with one of the most effective defense approaches in literature, which is adversarial training. The results available in Table 15 show that our proposed TENET architecture outperforms adversarial training in terms of robustness in all the experiments we conducted with different datasets.

Once we have evaluated the effectiveness of our proposed defense method on our comparably small models, we tried to test its performances on a considerably larger model. To do this, we first trained VGG-19 [58] models (with custom dropout layers) on CIFAR-10 dataset and achieved accuracy rates of 90.46% and 89.47% for normally trained and distilled models. Then, we have applied different attack algorithms and compared the attack success rates of TENET architecture with a standalone normal model. The results available in Table 16 reveal once again the efficacy of our proposed defense approach. To effectively use our technique in transfer learning settings, the transferred model should have already been trained using dropout layers (the layers which are located before the part of the model that is frozen).

As a last experiment, we have measured the time spent by our defense method for a batch of input of size 64 from the MNIST Dataset. In our local machine, it took 1,51 s to make a prediction with our proposed defense method, compared to 4.12 milliseconds of making a prediction directly without any previous operation. As expected, the execution time of our defense method is longer than making a single prediction due to additional uncertainty quantification steps and backward derivative operation.

6 Conclusion

In this study, we proposed a new defense architecture by significantly enhancing uncertainty-based reversal method and combining it with the utilization of defensive distillation technique. We evaluated and validated the effectiveness of our approach on three different datasets that are widely utilized in adversarial research field. The results of our extensive experiments suggest that our proposed architecture generalizes effectively across datasets and offers a very high degree of adversarial robustness without jeopardizing clean data classification performance.

In this research, we focused solely on the image domain and used only CNN models. However, we wonder if uncertainty-based reversal procedure is adaptable to other domains, such as audio or text, where different network models are used. Therefore, we plan to apply and evaluate the efficacy of our approach on other DNN architectures utilized in various domains. And finally, as a future work, we would like to test our method in transfer learning scenarios and work on potential improvements for tackling additional time and computational complexity introduced via our proposed defense method.

Acknowledgements This work was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) through the 1515 Frontier Research and Development Laboratories Support Program under Project 5169902, and has been partly funded by the European Union's Horizon Europe research and innovation programme and Smart Networks and Services Joint Undertaking (SNS JU) under Grant Agreement No: 101096034 (VERGE Project).

Data availability Datasets used in the manuscript can be found at: http://yann.lecun.com/exdb/mnist/, https://github.com/zalandoresearch/fashion-mnist, https://www.cs.toronto.edu/kriz/cifar.html.

Declarations

Conflict of interest The authors have no conflicts of interest to declare. All co-authors have seen and agreed with the contents of the manuscript. We certify that the submission is original work and is not under review at any other publication.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). arXiv:1512.03385
2. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks (2014). arXiv:1312.6082
3. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2014). arXiv:1312.6199
4. Sato, M., Suzuki, J., Shindo, H., Matsumoto, Y.: Interpretable adversarial perturbation in input embedding space for text (2018). arXiv:1805.02917
5. Carlini, N., Wagner, D.: Audio adversarial examples: Targeted attacks on speech-to-text (2018). arXiv:1801.01944
6. Finlayson, S.G., Chung, H.W., Kohane, I.S., Beam, A.L.: Adversarial attacks against medical deep learning systems (2019). arXiv:1804.05296
7. Sitawarin, C., Bhagoji, A.N., Mosenia, A., Chiang, M., Mittal, P.: Darts: Deceiving autonomous cars with toxic signs (2018). arXiv:1802.06430
8. Morgulis, N., Kreines, A., Mendelowitz, S., Weisglass, Y.: Fooling a real car with adversarial traffic signs (2019). arXiv:1907.00374
9. Tuna, O.F., Catak, F.O., Eskil, M.T.: Uncertainty as a swiss army knife: new adversarial attack and defense ideas based on epistemic uncertainty. Complex Intell. Syst. https://doi.org/10.1007/s40747-022-00701-0
10. Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., Yi, X.: A survey of safety and trustworthiness of deep neu-

ral networks: verification, testing, adversarial attack and defence, and interpretability. Comput. Sci. Rev. **37**, 100270 (2020). https://doi.org/10.1016/j.cosrev.2020.100270
11. Catak, F.O., Sivaslioglu, S., Sahinbas, K.: A generative model based adversarial security of deep learning and linear classifier models (2020). arXiv:2010.08546
12. Qayyum, A., Usama, M., Qadir, J., Al-Fuqaha, A.: Securing connected autonomous vehicles: challenges posed by adversarial machine learning and the way forward. IEEE Commun. Surv. Tutor. **22**(2), 998–1026 (2020). https://doi.org/10.1109/COMST.2020.2975048
13. Sadeghi, K., Banerjee, A., Gupta, S.K.S.: A system-driven taxonomy of attacks and defenses in adversarial machine learning. IEEE Trans. Emerg. Top. Comput. Intell. **4**(4), 450–467 (2020). https://doi.org/10.1109/TETCI.2020.2968933
14. Zheng, Z., Hong, P.: Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc. (2018). https://proceedings.neurips.cc/paper/2018/file/e7a425c6ece20cbc9056f98699b53c6f-Paper.pdf
15. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015). arXiv:1412.6572
16. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world (2017). arXiv:1607.02533
17. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. CoRR arXiv:1611.01236
18. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2019). arXiv:1706.06083
19. Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks (2016). arXiv:1511.04599
20. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks (2017). arXiv:1608.04644
21. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search (2020). arXiv:1912.00049
22. Chen, J., Jordan, M.I., Wainwright, M.J.: Hopskipjumpattack: A query-efficient decision-based attack. In: IEEE Symposium on Security and Privacy (SP) 2020, pp. 1277–1294 (2020). https://doi.org/10.1109/SP40000.2020.00045
23. Ilyas, A., Engstrom, L., Madry, A.: Prior convictions: Black-box adversarial attacks with bandits and priors (2019). arXiv:1807.07978
24. Tuna, O.F., Catak, F.O., Eskil, M.T.: Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples. Multimedia Tools Appl. **81**(8), 11479–11500 (2022). https://doi.org/10.1007/s11042-022-12132-7
25. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015). arXiv:1503.02531
26. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks (2016). arXiv:1511.04508
27. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser (2018). arXiv:1712.02976
28. Shen, S., Jin, G., Gao, K., Zhang, Y.: Ape-gan: Adversarial perturbation elimination with gan (2017). arXiv:1707.05474
29. Raghunathan, A., Steinhardt, J., Liang, P.: Certified defenses against adversarial examples (2020). arXiv:1801.09344
30. Tuna, O.F., Catak, F.O., Eskil, M.T.: Closeness and uncertainty aware adversarial examples detection in adversarial machine learning (2020). arXiv:2012.06390
31. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods (2020). arXiv:1910.09457
32. An, D., Liu, J., Zhang, M., Chen, X., Chen, M., Sun, H.: Uncertainty modeling and runtime verification for autonomous vehicles driving control: a machine learning-based approach. J. Syst. Softw. **167**, 110617 (2020)
33. Zheng, R., Zhang, S., Liu, L., Luo, Y., Sun, M.: Uncertainty in bayesian deep label distribution learning. Appl. Soft Comput. **101**, 107046 (2021). https://doi.org/10.1016/j.asoc.2020.107046
34. Antonelli, F., Cortellessa, V., Gribaudo, M., Pinciroli, R., Trivedi, K.S., Trubiani, C.: Analytical modeling of performance indices under epistemic uncertainty applied to cloud computing systems. Future Gen. Comput. Syst. **102**, 746–761 (2020). https://doi.org/10.1016/j.future.2019.09.006
35. Zhou, D.-X.: Universality of deep convolutional neural networks (2018). arXiv:1805.10769
36. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. (MCSS) **2**(4), 303–314 (1989). https://doi.org/10.1007/BF02551274
37. Loquercio, A., Segu, M., Scaramuzza, D.: general framework for uncertainty estimation in deep learning. IEEE Robot. Autom. Lett. **5**(2), 3153–3160 (2020). https://doi.org/10.1109/LRA.2020.2974682
38. Gurevich, P., Stuke, H.: Pairing an arbitrary regressor with an artificial neural network estimating aleatoric uncertainty. Neurocomputing **350**, 291–306 (2019). https://doi.org/10.1016/j.neucom.2019.03.031
39. Senge, R., Bösner, S., Dembczyński, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., Hüllermeier, E.: Reliable classification: learning classifiers that distinguish aleatoric and epistemic uncertainty. Inf. Sci. **255**, 16–29 (2014). https://doi.org/10.1016/j.ins.2013.07.030
40. Reinhold, J.C., He, Y., Han, S., Chen, Y., Gao, D., Lee, J., Prince, J.L., Carass, A.: Finding novelty with uncertainty (2020). arXiv:2002.04626
41. Graves, A.: Practical variational inference for neural networks. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 24, pp. 2348–2356. Curran Associates Inc, London (2011)
42. Paisley, J., Blei, D., Jordan, M.: Variational bayesian inference with stochastic search (2012). arXiv:1206.6430
43. Hoffman, M., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference (2013). arXiv:1206.7051
44. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks (2015). arXiv:1505.05424
45. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles (2017). arXiv:1612.01474
46. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning (2016). arXiv:1506.02142
47. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? (2017). arXiv:1703.04977
48. Kwon, Y., Won, J.-H., Kim, B.J., Paik, M.C.: Uncertainty quantification using Bayesian neural networks in classification: application to biomedical image segmentation. Comput. Stat. Data Anal. **142**, 106816 (2020). https://doi.org/10.1016/j.csda.2019.106816
49. Aladag, M., Catak, F.O., Gul, E.: Preventing data poisoning attacks by using generative models. In: 2019 1st International Informatics and Software Engineering Conference (UBMYK), pp. 1–5 (2019). https://doi.org/10.1109/UBMYK48245.2019.8965459
50. Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., Kurakin, A.: On evaluating adversarial robustness (2019). arXiv:1902.06705

51. LeCun, Y., Cortes, C.: MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/
52. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017). arXiv:1708.07747
53. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/~kriz/cifar.html
54. Rauber, J., Brendel, W., Bethge, M.: Foolbox: A python toolbox to benchmark the robustness of machine learning models (2018). arXiv:1707.04131
55. Ding, G. W., Wang, L., Jin, X.: advertorch v0.1: An adversarial robustness toolbox based on pytorch (2019). arXiv:1902.07623
56. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples (2018). arXiv:1802.00420
57. Katzir, Z., Elovici, Y.: Why blocking targeted adversarial perturbations impairs the ability to learn (2019). arXiv:1907.05718
58. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015). arXiv:1409.1556