USING UNCERTAINTY METRICS IN ADVERSARIAL MACHINE LEARNING AS AN ATTACK AND DEFENSE TOOL

ÖMER FARUK TUNA

IŞIK UNIVERSITY
DECEMBER, 2022

USING UNCERTAINTY METRICS IN ADVERSARIAL MACHINE LEARNING AS AN ATTACK AND DEFENSE TOOL

ÖMER FARUK TUNA
Işık University, Graduate School of Higher Education
Computer Engineering Doctoral Program, 2022

This thesis is submitted to the Graduate School of Higher Education at
Işık University for the degree of Doctor of Philosophy (PhD)
in Computer Engineering

IŞIK UNIVERSITY
DECEMBER, 2022

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

USING UNCERTAINTY METRICS IN ADVERSARIAL MACHINE LEARNING AS AN ATTACK AND DEFENSE TOOL

ÖMER FARUK TUNA

APPROVED BY:

Prof. Dr. M. Taner Eskil (Thesis Supervisor)	Işık University
Prof. Dr. Uluğ Bayazıt	İstanbul Technical University
Assist. Prof. Emine Ekin	Işık University
Prof. Dr. Olcay Taner Yıldız	Özyeğin University
Assist. Prof. Rahim Dehkharghani	Işık University

APPROVAL DATE: 19/12/2022

USING UNCERTAINTY METRICS IN ADVERSARIAL MACHINE LEARNING AS AN ATTACK AND DEFENSE TOOL

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, defined as adversarial samples. 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 thesis study, we leverage the use of various uncertainty metrics obtained from MC-Dropout estimates of the model for developing new attack and defense ideas. On defense side, we propose a new adversarial detection mechanism and an uncertainty-based defense method to increase the robustness of DNN models against adversarial evasion attacks. On the attack side, we use the quantified epistemic uncertainty obtained from the model's final probability outputs, along with the model's own loss function, to generate effective adversarial samples. We've experimentally evaluated and verified the efficacy of our proposed approaches on standard computer vision datasets.

Keywords: Deep Neural Networks, Adversarial Machine Learning, Uncertainty Quantification, Monte-Carlo Dropout Sampling, Epistemic Uncertainty, Aleatoric Uncertainty, Scibilic Uncertainty

BELİRSİZLİK METRİKLERİNİN HASMANE MAKİNE ÖĞRENMESİNDE SALDIRI VE SAVUNMA AMAÇLI KULLANILMASI

ÖZET

Derin Sinir Ağları modelleri, yaygın olarak rastgele bozulmalara karşı dirençleri ile bilinir. Bununla birlikte, araştırmacılar, bu modellerin, karşıt (hasmane) örnekler olarak adlandırılan girdinin kasıtlı olarak hazırlanmış ve görünüşte algılanamaz bozulmalarına karşı gerçekten son derece savunmasız olduğunu keşfettiler. Bu gibi hasmane saldırılar, Derin Sinir Ağları tabanlı yapay zeka sistemlerinin güvenliğini önemli ölçüde tehlikeye atma potansiyeline sahiptir ve özellikle güvenliğin öncelikli olduğu alanlarda yüksek riskler oluşturur. Bu saldırılara karşı savunma yapmak ve hasmane tehditlere karşı daha dayanıklı mimariler geliştirmek için son yıllarda çok sayıda çalışma yapılmıştır.

Bu tez çalışmasında, yeni saldırı ve savunma fikirleri geliştirmek için modelin Monte-Carlo Bırakma Örneklemesinden elde edilen çeşitli belirsizlik metriklerinin kullanımından yararlanıyoruz. Savunma tarafında, hasmane saldırılara karşı yapay sinir ağı modellerinin sağlamlığını artırmak için yeni bir tespit mekanizması ve belirsizliğe dayalı savunma yöntemi öneriyoruz. Saldırı tarafında, etkili hasmane örnekler oluşturmak için modelin kendi kayıp fonksiyonu ile birlikte modelin nihai olasılık çıktılarından elde edilen nicelleştirilmiş epistemik belirsizliği kullanıyoruz. Standart bilgisayarlı görü veri kümeleri üzerinde önerilen yaklaşımlarımızın etkinliğini deneysel olarak değerlendirdik ve doğruladık.

Anahtar Kelimeler: Derin Sinir Ağları, Karşıt Makine Öğrenmesi, Monte-Carlo Bırakma Örneklemesi, Model Belirsizliği, Epistemik Belirsizlik, Rassal Belirsizlik, Bilinebilir Belirsizlik

ACKNOWLEDGEMENTS

In the first place, I would like to offer my heartfelt gratitude to my thesis supervisor Prof. Dr. Taner Eskil for allowing me the opportunity to pursue the research subjects that I am passionate about and for his essential guidance along my thesis journey. He guided and encouraged me in all the time of my academic life. This made it easier to build a strong sense of trust between us and gave me a great degree of independence for my research studies. It was very big pleasure for me working with him.

I'd want to express my sincere thanks to the members of my thesis monitoring committee: Prof. Dr. Uluğ Bayazıt and Assist. Prof. İlknur Karadeniz for their valuable suggestions and comments. I would like to thank Assist. Prof. Emine Ekin for giving me a chance of being an academician training program scholar in our department. I would like to thank Prof. Dr. Olcay Taner Yıldız for helping me become acquainted with Machine Learning and Artificial Intelligence, which I now find to be fascinating. Also, many thanks to my thesis jury members for their interest and support. Finally, I'd like to thank Assoc. Prof. Ferhat Özgür Çatak for his advices and assistances throughout my research.

Last but not least, I'd like to express my deepest appreciation to my parents Sevgi Yalgın, Nazım Tuna and my lovely wife Selin Tuna. Thanks a lot for their unconditional support and love.

Ömer Faruk TUNA

This thesis study is gratefully dedicated to my dear parents, who have served as an inspiration to me and who consistently give their mental, emotional, and financial assistance

And I am grateful for the unending encouragement of my beautiful wife.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AI:	Artificial Intelligence
BIM:	Basic Iterative Method
CNN:	Convolutional Neural Network
CW:	Carlini Wagner
DNN:	Deep Neural Network
FGSM:	Fast Gradient Sign Method
JSMA:	Jacobian-based Salience Map Attack
KDE:	Kernel Density Estimation
KL:	Kullback-Leibler
ML:	Machine Learning
MAP:	Maximum Aposterior Estimate
MC:	Monte Carlo
MCMC:	Monte Carlo Markov Chain
MLE:	Maximum Likelihood Estimation
MLP:	Multi Layer Perceptron
MSE:	Mean Square Error
NLP:	Natural Language Processing
NN:	Neural Network
PGD:	Projected Gradient Descent
ResNet:	Residual Neural Network
UAP:	Universal Adverserial Perturbation
VGG:	Visual Geometry Group

CHAPTER 1

INTRODUCTION

Deep learning is a subset of machine learning (ML) that uses artificial neural networks (ANN) to enable multi-layered data processing models to learn from data at various levels of abstraction. Deep learning models obtain the data representations for every individual layer from the preceding layer's representation using the backpropagation technique and find subtle features in massive amount of data (LeCun, Bengio, & Hinton, 2015). Deep learning has emerged to be remarkably powerful in solving issues for which traditional machine learning methods were ineffective. Deep learning has achieved great success in a broad range of tasks (Ciodaro, Deva, de Seixas, & Damazio, 2012; Ma, Sheridan, Liaw, Dahl, & Svetnik, 2015) thanks to the evolution of deep neural network (DNN) models, the accessibility of vast amounts of data as well as the availability of high-performance hardware to train complicated models.

Deep learning models have started to outperform humans in the past few years. To give an example, in the ”ImageNet Large Scale Visual Recognition Challenge (ILSVRC)”, a DNN model named ResNet (He, Zhang, Ren, & Sun, 2016) beat human performance in 2015, and the record was later broken by more advanced architectures. Similarly, Goodfellow et al. (Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014) 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 (Chouard, 2016). Many advanced systems are now being developed using DNN models, which have proven to be extremely successful in a variety of domains, including medical diagnosis (Causey et al., 2018; Greenspan, van Ginneken, & Summers, 2016; Gulshan et al., 2016; Shen et al., 2019), autonomous vehicles (Janai, Güney, Behl, & Geiger, 2020;

Redmon, Divvala, Girshick, & Farhadi, 2016; Ren, He, Girshick, & Sun, 2015), game play (Justesen, Bontrager, Togelius, & Risi, 2020), and machine translation (Bahdanau, Cho, & Bengio, 2015; Luong, Pham, & Manning, 2015). However, the main emphasis of the researchers during the rise of DNN models was the creation of increasingly precise models and the robustness and reliability 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.

1.1 Vulnerabilities of AI-driven systems

Within last decade, researchers discovered that existing DNN models are vulnerable to meticulously crafted attacks. Szegedy et al. (Szegedy et al., 2014) were among the very first who noticed the effectiveness 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 to 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. (Sato, Suzuki, Shindo, & Matsumoto, 2018) demonstrated in the NLP domain that altering merely one word from an input sentence can deceive a sentiment analyser trained with textual data. A further example is in the audio domain (Carlini & Wagner, 2018), 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 (Finlayson et al., 2019). Similarly, self-driving cars employ ML to navigate traffic without the need for human involvement. A mislead decision of the autonomous vehicle based on an adversarial attack could result in a tragic accident (Morgulis, Kreines, Mendelowitz, & Weisglass, 2019; Sitawarin, Bhagoji, Mosenia, Chiang, & Mittal, 2018). 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.

1.2 Importance of Uncertainty for AI-driven systems

If we put the security related threats against AI models aside, there are also other important concerns regarding the reliability and robustness of AI-driven systems. Today's AI driven systems are forced to make predictions even in the cases where the model is not confident in it's predictions or the model has not encountered the concept while training. We do need to know how probable the predictions of AI models are right, which could only be accomplished by including uncertainty estimates into the model. However, it is known that model uncertainty is not captured by conventional DNN-based models by default.

For AI-driven systems, it is crucial for us as the users to know what a model doesn't actually know (Ghahramani, 2015). Uncertainty is vital to enhance the reliability and security of AI-driven technologies (McAllister et al., 2017). DNNs are nowadays capable of developing potent representations that can map high dimensional data to a variety of outputs. Nevertheless, these mappings are frequently taken for granted and thought to be always correct, which may not actually be the case (Kendall, 2018). This fact has led to catastrophic results in two recent situations. The first death from an aided driving system occurred in May 2016, when the perception system misinterpreted the white side of a trailer under a bright sun light (Technical Report, Jan 2017). In another recent case, an image tagging algorithm offered by Google (Google Photos) misidentified two African Americans as gorillas as shown in Figure 1.1, raising worries about systemic racism (Guynn, 2015). However, both of those algorithms could have been able to make better predictions and probably prevent disaster if they had been able to associate a significant degree of uncertainty along with their inaccurate predictions. These examples show the importance of using uncertainty information for assessing the quality of AI model's predictions and therefore prevent us from relying on erroneous decisions.

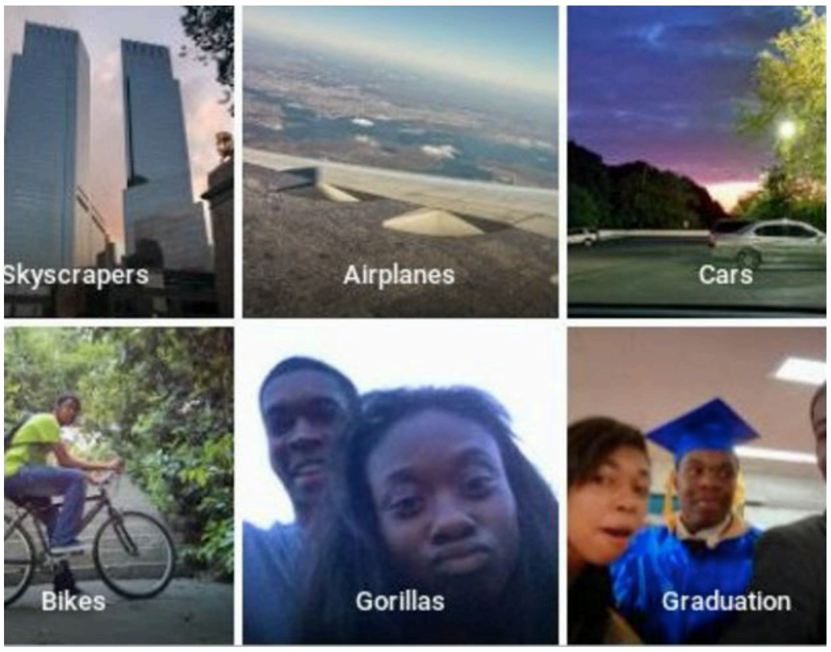


Figure 1.1 Google Photos erroneously labelled a black couple as being gorillas (BBC news, July 2015)

1.3 Problem Statement

The aim of adversarial attacks is to craft a perturbation δ under given constraint on magnitude ($\delta < \varepsilon$) which results into a wrong prediction as $h(x+\delta,w) = y^{adv}$ that is different from a prediction on a clean sample $h(x,w) = y$. In this definition, $h(x,w)$ is the target model with weights denoted as w and x is the input sample. The success criterion of the attack may vary depending on the type of task. In the case of a classification problem, the attack might be considered successful if the model predicts a class other than the actual class. From the perspective of the attacker, the perturbation δ added to the input sample x should be as small as possible and yet sufficient to deceive the target model. On the other hand, from the perspective of the model owner or defender, the model $h(x,w)$ should be as robust as possible and keep predicting accurately even in the case of a powerful adversarial attack threat. It is indeed preferable that the model could distinguish a benign input sample from malicious input sample which is specifically crafted to fool the model. All these aforementioned subjects, each of which constitute a very active research area in its own, have evolved as a consequence of the arms race among both attackers and defenders. These are the core problems to be addressed as part of this thesis.

1.4 Motivation for Using Uncertainty Information

Previous research (Shridhar, Laumann, & Liwicki, 2018) has found a correlation between validation accuracy and uncertainty; as validation accuracy increases, quantified uncertainty reduces, and vice versa. This discovery is based on the premise that the more accurate labels our model predicts, the more confident it is in these predictions. Based on this fact, we would like to investigate whether the uncertainty metrics can be used for adversarial machine learning purposes. The reasoning behind this is that the whole purpose of a malicious adversarial sample is to somehow deceive the model into making an incorrect prediction.

To support our standpoint about why we think uncertainty information could be a useful tool in adversarial ML, we have made some preliminary experiments. We have trained different classifiers to predict all the samples of MNIST (Digit), MNIST (Fashion) and CIFAR10 test datasets. Then, we have analyzed the values of various uncertainty metrics for all the correct and wrong predictions. We observed that for the samples which were predicted correctly; quantified uncertainty metrics are low, and model confidences are high. Furthermore, for the wrong predictions, quantified uncertainty metrics are high, and model confidences are low. The results are summarized in Tables 1.1,1.2 and 1.3.

Our results show that uncertainty indicators are indeed useful tools to evaluate the quality of the model predictions. Furthermore, there are recent studies in literature which investigate the use of various uncertainty metrics such as mutual information and predictive entropy as possible indicators for an adversarial attempt (Feinman, Curtin, Shintre, & Gardner, 2017; Smith & Gal, 2018). There are also more recent studies (Stutz, 2022; Stutz, Hein, & Schiele, 2020) which utilize uncertainty information for improving the robustness of deep neural networks. These findings have deeply motivated us for investigating the role of uncertainty in adversarial machine learning.

Table 1.1 Uncertainty values of the model for the MNIST (digit) data.

Expected value	Wrong Predictions	Correct Predictions
Epistemic Uncertainty	0.0218	0.0014
Aleatoric Uncertainty	0.0298	0.0028
Scibilic Uncertainty	0.7936	0.1798
Entropy	0.9722	0.10124

Table 1.2 Uncertainty values of the model for the MNIST (Fashion) data.

Expected value	Wrong Predictions	Correct Predictions
Epistemic Uncertainty	0.0065	0.0011
Aleatoric Uncertainty	0.0394	0.0083
Scibilic Uncertainty	0.1836	0.05281
Entropy	0.8234	0.1804

Table 1.3 Uncertainty values of the model for the CIFAR10 data.

Expected value	Wrong Predictions	Correct Predictions
Epistemic Uncertainty	0.0282	0.0058
Aleatoric Uncertainty	0.0404	0.0091
Scibilic Uncertainty	0.6671	0.2861
Entropy	0.9699	0.3039

1.5 Main Contributions of the Thesis Dissertation

This study is based on utilizing uncertainty estimates derived from the DNN-based classifiers for both defense and attack purposes. We seek to find effective ways of using uncertainty information for detecting any malicious adversarial attempts and improving robustness for DNN-based classifiers. We also focus on using quantified uncertainty estimates for crafting efficient adversarial perturbations. Figure 1.2 depicts the overall content of this thesis.

Our main contributions in this research are:

- We investigated the use of various uncertainty metrics (entropy, epistemic uncertainty, aleatoric uncertainty, scibilic uncertainty) to detect adversarial samples.
- We proposed a novel method for quantifying the closeness of an input sample's representation to its predicted class data distribution in the subspace of last hidden layer activations.

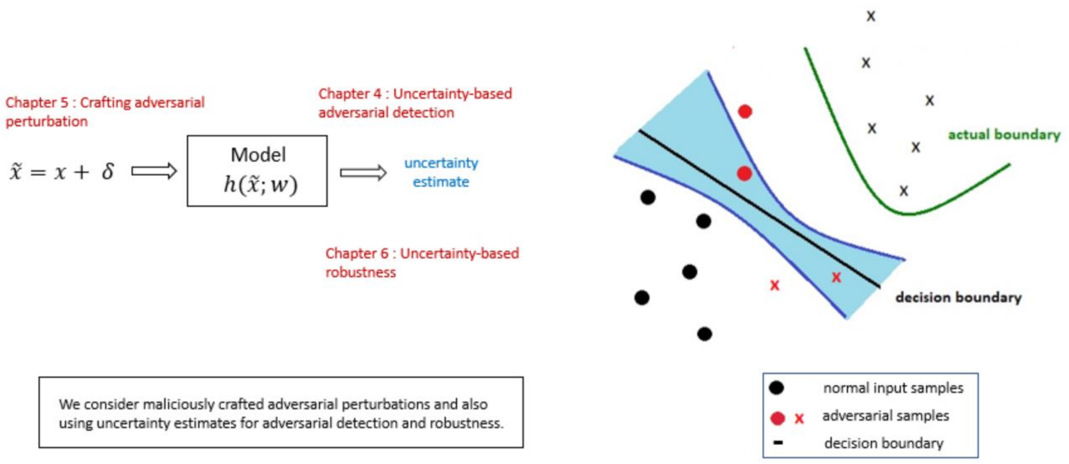

Chapter 5 : Crafting adversarial perturbation
$\tilde{x} = x + \delta$
Model
$h(\tilde{x}; w)$
Chapter 4 : Uncertainty-based adversarial detection
uncertainty estimate
Chapter 6 : Uncertainty-based robustness
We consider maliciously crafted adversarial perturbations and also using uncertainty estimates for adversarial detection and robustness.
actual boundary
decision boundary
normal input samples
adversarial samples
decision boundary

Figure 1.2 Illustration of thesis content using a 2-dimensional and 2-class classification task

- We showed experimentally that no single measure performs well in all circumstances and that an ensemble technique utilizing a number of metrics should be used for adversarial sample detection.
- We utilized a new metric (epistemic uncertainty of the model) which can be exploited to craft adversarial examples.
- We showed that the performance of a pure uncertainty-based attack is indeed as high as the attacks based on the model loss.
- We demonstrated that crafting adversarial examples using both the model loss and uncertainty yields better performance in adversarial attacks.
- We introduced a novel and effective attack method by utilizing the model's epistemic uncertainty, which yields more powerful adversarial impact with less amount of perturbation at each step.
- We introduce a new adversarial defense technique which provides high degree of robustness against some of the strongest attacks like Deepfool attack and Carlini-Wagner attack (under default setting).
- We enhanced the performance of this uncertainty-based reversal 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 research community to use scibilic uncertainty for the aim of building more robust models.
- We introduced a hybrid architecture which combines different known defense methods like adversarial training and defensive distillation technique with our uncertainty based reversal method. We experimentally show that these two approaches can handle complementary situations and they together provide very high degree of robustness against different attack types.

1.6 Organization of the Thesis Dissertation

The remainder of this thesis is structured as follows: Chapter 2 will introduce the research field of adversarial machine learning with an emphasis on evasion attacks, discussing some of the known attack types and defense techniques in the literature. In Chapter 3, we will introduce the notion of uncertainty in machine learning together with its main types and discuss how we can efficiently quantify different uncertainty metrics for DNN-based classifiers. Chapter 4 will focus on detection of adversarial samples using uncertainty metrics. In Chapter 5, we will present our uncertainty-based adversarial attack methods. Chapter 6 will present our work on defending adversarial attacks using various uncertainty metrics. Each of these chapters will include detailed methodology and experimental results sections. And in Chapter 7, we will wrap up our thesis work with conclusions.

CHAPTER 2

ADVERSARIAL MACHINE LEARNING

Since the discovery of DNN's vulnerability to adversarial attacks (Szegedy et al., 2014), a vast amount of research has been conducted in both devising new adversarial attacks and defending against these attacks with more robust DNN models (Huang et al., 2020). After giving some preliminary information for adversarial attacks in Section 2.1, we will treat the attack and defense studies separately and review some of the notable ones in Section 2.2-2.3.

2.1 Adversarial Attacks

Deep learning models contain many vulnerabilities and weaknesses which make them difficult to defend in the context of adversarial machine learning. For instance, they are often sensitive to small changes in the input data, resulting in unexpected results in the model's final output. Figure 2.1 shows how an adversary would exploit such a vulnerability and manipulate the model through the use of carefully crafted perturbation applied to the input data.

The malicious perturbation is applied upon the original image and it manipulates the model in such a way that a ”Chihuahua (Dog)” is wrongly classified as ”Sports Car” with high degree of confidence.

We will continue by giving the formal definition of what an adversarial sample is and give some preliminary information about the basic criteria which are used to classify adversarial attacks.

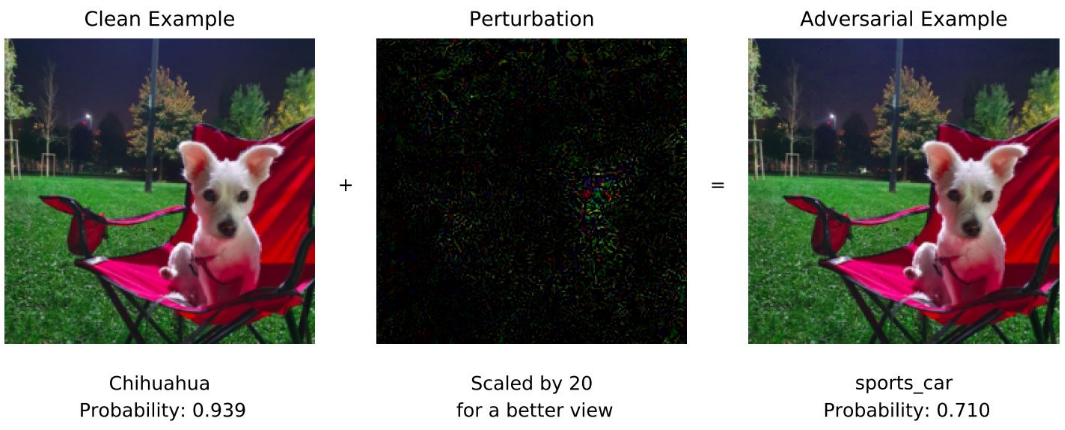


Figure 2.1 An example of adversarial attack.

2.1.1 Formal Definition of Adversarial Sample

Let k represent the total number of output classes, where $\mathscr{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ denotes our dataset, and $x_i \in \mathbb{R}^d$ and y_i are the i^{th} input and output respectively and N is size of our dataset. Based on this notation, we can define a neural network represented as $\mathbf{h}(.)$ which successfully maps the input x to output y via $\mathbf{h}(\mathbf{x})$. We can consider an input sample of $x^{'}$ to be an adversarial equivalent of x, when x and $x^{'}$ are close to each other under a particular distance metric and also the model output for $x^{'}$ is different from model output for x. In formal terms, we can formulate an adversarial sample of x as shown below:

$$x^{'} : D(x, x^{'}) < \varepsilon,\ h(x^{'}) \neq y$$

where $D(.,.)$ is the distance metric and ε is a constraint representing the maximum amount of allowed perturbation.

2.1.2 Distance Metrics

According to the formal definition above, an adversarial example x^{adv} shall be near to a benign sample x under a particular distance norm. This is especially an important requirement which makes the attack unnoticeable to the human eye. To formulate the maximum allowed perturbation amount, L_p distance is generally being used. The L_p distance between an adversarial sample x^{adv} and a normal input sample x can be represented as $\left\| x - x^{adv} \right\|_p$, where $\|.\|_p$ can be denoted as :

$$\|x\|_p = (|x_1|^p + |x_2|^p + |x_3|^p + \ldots + |x_d|^p)^{1/p}$$

where d represents the dimension of the vector x.

Researchers commonly use 3 types of metrics to limit the maximum allowed perturbation amount applied on the input sample. And these metrics are L_2, L_0 and L_∞. L_2 distance is the conventional Euclidean distance. L_∞ distance is the greatest change to any of the input attributes. And L_0 distance is the number of distinct input features.

2.1.3 Attacker Objective

We now introduce different sorts of adversarial attacks. Based on the attacker's objective, adversarial attacks can be categorized under two types as untargeted and targeted attacks. In the untargeted attack scenario, the attacker perturbs the input sample, causing the model to predict a class other than the actual class. In the targeted scenario, the attacker perturbs the input sample so that a particular target class is predicted by the model.

2.1.4 Capability of the Attacker

The last criteria for grouping the adversarial attacks is the threat model. White-box adversarial attacks assumes access of an attacker to the model details like architecture and weights. It is considered a valid security threat (Katzir & Elovici, 2021) and therefore this threat model has been intensively studied in literature. Under this threat model, researchers have mostly proposed gradient based approaches for loss maximization, or constrained optimization based methods to generate adversarial examples. Figure 2.2 illustrates White-box setting where an adversary uses the target AI model to craft an adversarial perturbation.

There is also the Black-box threat model where the attacker does not have access to model details. In this threat model, the attacker can only access the target model's outputs (prediction scores or the final decisions of the model). In this type of setting, adversary interacts with the target AI model to query the predictions for specific inputs as illustrated in Figure 2.3.

Black-box attacks are classified as score-based or decision-based according to whether the attacker has access to the entire probability output scores, or the predicted

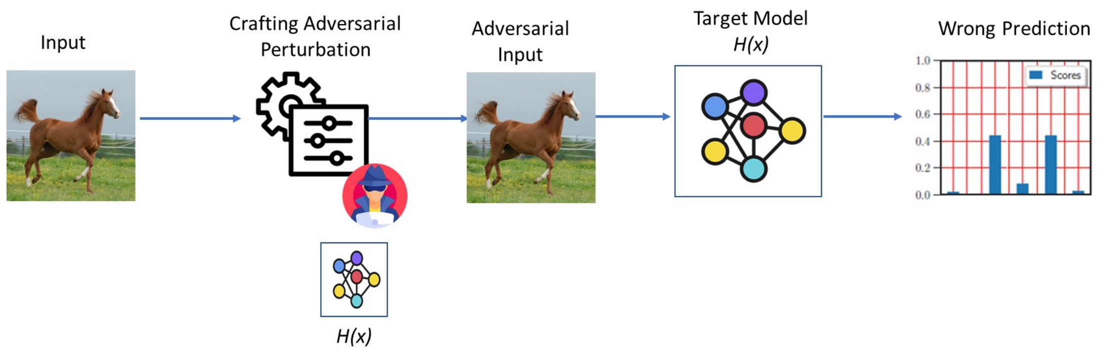


Figure 2.2 Illustration of White-box Setting

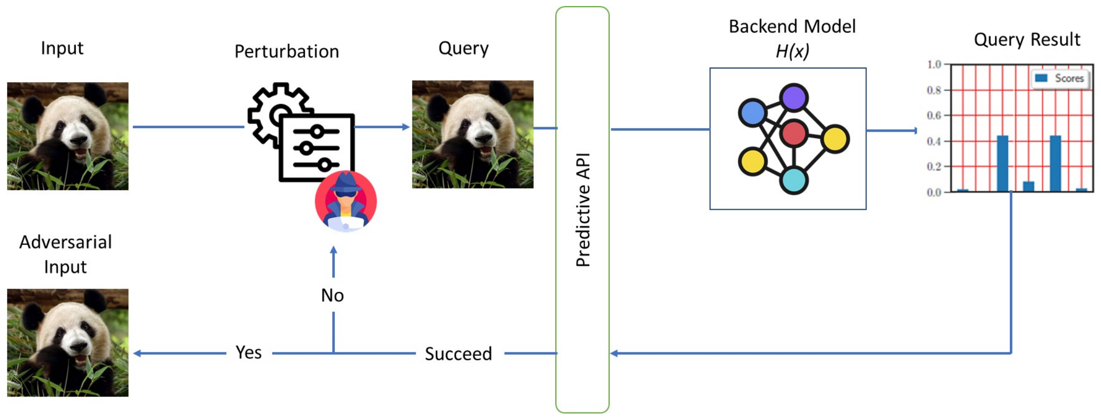


Figure 2.3 Illustration of Black-box Setting

label of a given input. Chen et al. (P.-Y. Chen, Zhang, Sharma, Yi, & Hsieh, 2017) developed score-based approaches for generating adversarial examples using zeroth-order gradient estimation. Chen et al. (J. Chen, Jordan, & Wainwright, 2020) proposed decision-based methods based on an estimation of model's gradient direction and binary search procedure for approaching the decision boundary. One of the drawbacks of these kind of Black-box adversarial attack ideas which makes them unpractical in real world scenarios is that the adversary needs to make very high number of queries to the target model (i.e. more that 25k) and the difference between the query samples of each successive sample is so small (input samples are usually quite similar in input space) which gives the AI model owner high chance of detecting these malicious attempts as in (S. Chen, Carlini, & Wagner, 2020; Li et al., 2022).

There is also the concept of attack transferability in adversarial ML. It has been observed that an adversarial sample crafted using a surrogate model can fool the target AI model as in (Papernot et al., 2017). In this context, transferability refers to the capability of a malicious attack to be successful against another, presumably unknowable model. Since the malicious actor does not have a copy of the target model and instead uses his/her own surrogate model, this scenario is also considered as Black-box attack scenario. However, the need for the adversary to train a surrogate model requires to collect huge amount of input training samples makes it again unpractical in many real world scenarios.

2.2 Adversarial Attack Types

Many different adversarial attack algorithms have been proposed in literature in the past few years. In this section, we will briefly describe some of the notable adversarial machine learning attacks.

2.2.1 Fast-Gradient Sign Method

This approach, sometimes known as FGSM (Goodfellow, Shlens, & Szegedy, 2015), is amongst 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.

$$\mathbf{x}^{adv} = \mathbf{x} + \varepsilon \cdot sign\left(\nabla_x J(\theta, \mathbf{x}, y_{true})\right) \tag{2.1}$$

In (Kurakin, Goodfellow, & Bengio, 2017), the authors presented a targeted variant of FGSM referred to as the Targeted Gradient Sign Method (TGSM). This way, they could change the attack to try to convert the model's prediction to a particular class. To achieve this, instead of maximizing the loss with respect to the true class label, TGSM attempts to minimize the loss with respect to the target class J_{target}.

$$\mathbf{x}^{adv} = \mathbf{x} - \varepsilon \cdot sign\left(\nabla_x J(\theta, \mathbf{x}, y_{target})\right) \tag{2.2}$$

Different from Eq. 2.1, we now subtract the crafted perturbation from the original image as we try to minimize the loss this time. If we want to increase the efficiency of this approach, we can modify above equation as in Eq.2.3. The only difference is that instead of minimizing the loss of the target label, we maximize the loss of the true label and also minimize the loss for the alternative label.

$$\mathbf{x}^{adv} = \mathbf{x} + \varepsilon \cdot sign\left(\nabla_x (J(\theta, \mathbf{x}, y_{true}) - J(\theta, \mathbf{x}, y_{target}))\right) \tag{2.3}$$

Another important aspect of FGSM is that it is not intended to be optimum, but rather fast. It isn't 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.

2.2.2 Iterative Gradient Sign Method

Kurakin et al. (Kurakin et al., 2017) 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 2.4 describes how to generate perturbed images under the l_{inf} norm for a BIM attack.

$$\begin{aligned} \mathbf{x}_t^* &= \mathbf{x} \\ \mathbf{x}_{t+1}^* &= clip_{x,\varepsilon}\{\mathbf{x}_t + \alpha \cdot sign\left(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}_t^*, y_{true})\right)\} \end{aligned} \tag{2.4}$$

where $\mathbf{x}$ is the clean sample input to the model, $\mathbf{x}^*$ is the output adversarial sample at i^{th} 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 (Kurakin, Goodfellow, & Bengio, 2016). The attacker can manage how far an adversarial sample is pushed further away from the decision boundary by configuring the ε parameter.

One can group BIM attacks under two main types, namely BIM-A and BIM-B. In the former type, we stop iterations as soon as we succeed in fooling the model (passing the decision boundary), while in the latter, we continue the attack till the end of the provided number of iterations so that we push the input further away the decision boundary. This phenomenon is illustrated in Figure 2.4.

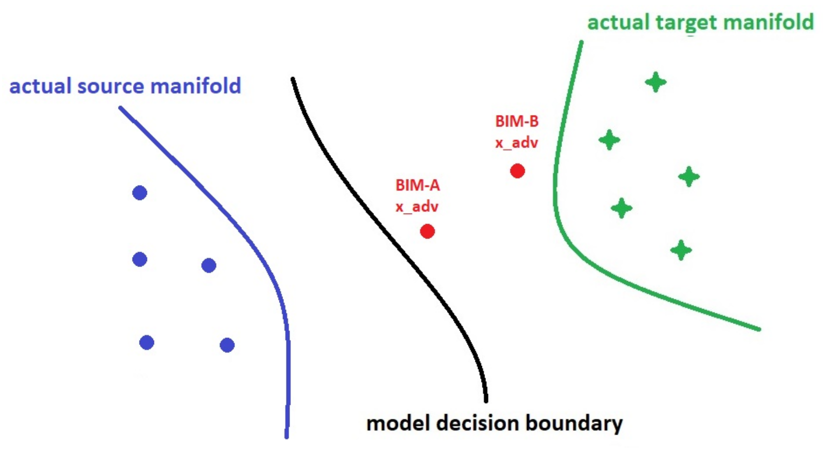


Figure 2.4 BIM-A vs BIM-B

As in the case of TGSM, we can easily modify Eq. 2.4 to produce targeted variant of BIM. At each intermediate step, we can try to minimize the loss with respect to target class while at the same time maximizing the loss with respect to original class.

2.2.3 Projected Gradient Descent

This attack type, commonly known as PGD, has been proposed by Madry et al. (Madry, Makelov, Schmidt, Tsipras, & Vladu, 2018). 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 ($\varepsilon = 0$, in all the dimensions), PGD employs random start, which can be defined as:

$$\mathbf{x}_0 = \mathbf{x} + P(-\varepsilon, +\varepsilon) \tag{2.5}$$

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

2.2.4 Jacobian-based Saliency Map Attack (JSMA)

This method, also known as JSMA, has been proposed by Papernot et al. (Papernot et al., 2017). It is designed to be used under L_0 distance norm which takes total number of altered pixels into count, thereby restricting the attacker. It is a greedy algorithm which selects two pixels at a time. The algorithm utilizes the gradient $\nabla Z(x)_l$ to compute a saliency map, which shows each pixel's impact on the classification of each class. And the aim is to enhance the possibility of the target class while diminishing the possibility of other classes by selecting and updating two pixels at a time based on the saliency map. The attack is continued until either a predefined number of pixels is modified or the model is successfully fooled.

2.2.5 Carlini&Wagner Attack

Proposed by Carlini and Wagner (Carlini & Wagner, 2017b), and it is one of the strongest attack algorithms so far. As a result, it's 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 adversarially trained and 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 be 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 decision boundary. And high confident adversaries generally located further away from decision boundary.

2.2.6 Deepfool Attack

This attack method has been introduced by Moosavi-Dezfooli et al. (Moosavi-Dezfooli, Fawzi, & Frossard, 2016) and it is one of the strongest untargeted attack algorithms in literature. It's made to work with several distance norm metrics, including l_{inf} and l_2 norms.

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 Figure 2.5. This attack can break defensive distillation method and achieves higher success rates than previously mentioned iterative attack approaches. But the downside is, produced adversarial sample generally lies close to the decision boundary of the model.

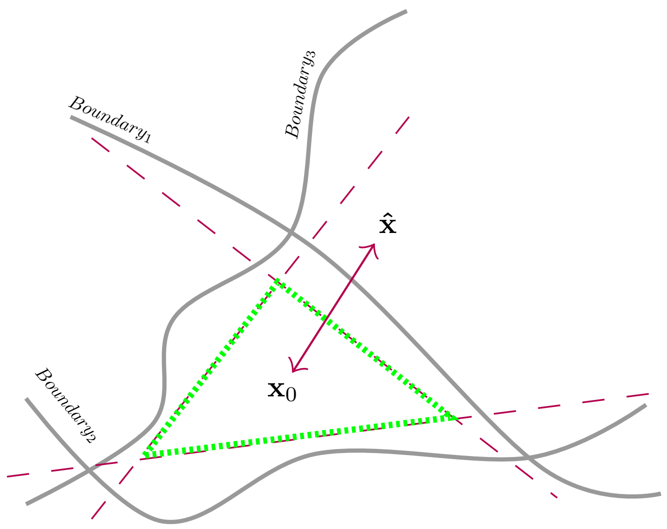


Figure 2.5 Illustration of Deepfool attack algorithm

2.2.7 Hopskipjump Attack

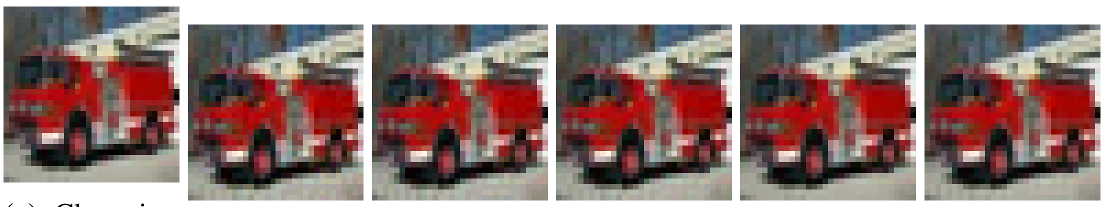
HopSkipJumpAttack (HSJA) (J. Chen et al., 2020) is a decision-based black-box attack algorithm in which the adversary does not have any information about the model architecture and weights. In this algorithm, the attacker makes a large number of queries to the target model and observe the decisions of the model. The attack can be thought of as an iterative method where the whole process is based on a binary-search procedure for approaching the decision boundary and an estimation of model's gradient direction. The important thing to note about this attack as all the other black-box attacks is that the resulting adversarial samples lie very close to the decision boundary of the target model and therefore the final prediction probability scores of the predicted wrong class is very close to the probability score of the actual class.

2.2.8 Universal Adversarial Attack

The carefully created adversarial perturbations were unique to benign samples in each of the aforementioned attacks. This limitation has led the researchers to determine if there exists a universal perturbation that can successfully deceive the target model on vast majority of benign samples. Instead of crafting perturbations to deceive a target model on a single input, Moosavi-Dezfooli et al. (Moosavi-Dezfooli, Fawzi, Fawzi, & Frossard, 2017) suggested a method to craft universal adversarial perturbations that can deceive a model on any input with high confidence. In contrast to being highly transferable, universality is related to the attribute of a perturbation being ”input-agnostic.”

Adversarial machine learning is a burgeoning field of research, and we see a lot of new adversarial attack algorithms being proposed. Some recent studies are Square Attack (Andriushchenko, Croce, Flammarion, & Hein, 2020), Bandit (Ilyas, Engstrom, & Madry, 2019). Besides, some recent studies utilize MC Dropout sampling and uncertainty information to craft adversarial samples. Liu et al. (Liu et al., 2019) proposed Universal Adversarial Perturbation (UAP) method that utilizes a metric called virtual uncertainty obtained from the model's structural activation. However, estimating the model's uncertainty involves aggregating all the neurons' virtual uncertainties, which is computationally costly. Finally, Tuna et. al. proposed Uncertainty-Based Attacks (O. F. Tuna, Catak, & Eskil, 2022b, 2022c) which utilizes both the model's loss function and quantified epistemic uncertainty to generate more powerful attacks.

Figure 2.6 shows adversarial samples generated by some of the attack algorithms discussed earlier.



(a) Clean image **(b)** FGSM **(c)** BIM **(d)** PGD **(e)** DeepFool **(f)** CW

Figure 2.6 An example image from CIFAR10 dataset and some of the adversarial samples crafted by using previously mentioned attack types

2.3 Adversarial Defense

Since the discovery of DNN's vulnerability to adversarial attacks (Szegedy et al., 2014), a vast amount of research has been conducted on defending against these attacks. Defense against adversarial attacks can be divided into two categories; (i) improving the robustness of classifiers to existing attack types, (ii) methods for detecting adversarial samples. We first present important defense approaches in literature, then we will mention some notable methods for the detection of the adversarial samples.

2.3.1 Defensive Distillation

Although the idea of knowledge distillation was previously introduced by Hinton et al. (Hinton, Vinyals, & Dean, 2015) 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. (Papernot, McDaniel, Wu, Jha, & Swami, 2016). The algorithm starts with training a *teacher model* on training data by employing a high temperature (T) value in the softmax function as in Equation 2.6, where p_i is the probability of i^{th} class and z_i's are the logits.

$$p_i = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_i}{T})} \tag{2.6}$$

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 2.7 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).

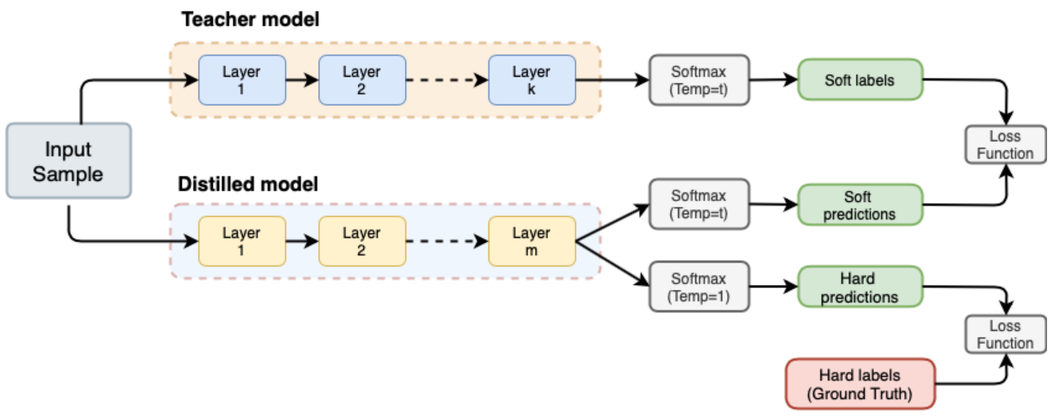
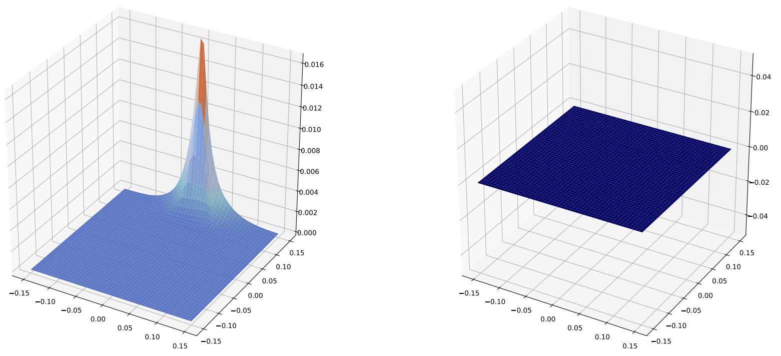


Figure 2.7 Defensive Distillation.



(a) normal model **(b)** distilled model

Figure 2.8 Loss surfaces of ”normally-trained” and ”distilled” models

As depicted in Figure 2.8, 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.

2.3.2 Adversarial Training

Adversarial training is recognized 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. 2.7:

$$\underset{\theta}{min}\ \underset{|\delta\|\leq\varepsilon}{max}\ J(h_\theta(x+\delta),y) \tag{2.7}$$

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. (Goodfellow et al., 2015) 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. And it is known that although adversarially trained models are somewhat resistant to adversarial samples to some extent, these models generally suffer from severe overfitting issue which is known as robustness-accuracy trade-off (Su et al., 2018).

2.3.3 Magnet

Meng et al. (Meng & Chen, 2017) suggested a defense mechanism with two components: a detector and a reformer. The former inspects input samples to determine if they are benign or not and the latter reforms inputs classified as benign by the detector to remove any leftover adversarial nature. Although the authors demonstrate the efficiency of their defense against several attack types, their approach was then shown to be vulnerable to CW attacks (Carlini & Wagner, 2017a).

Adversarial ML is a very active field of research, and new adversarial defense approaches are constantly being presented. Among the most notable are: i) High-Level Representation Guided Denoiser (HGD) (Liao et al., 2018) 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 (Jin, Shen, Zhang, Dai, & Zhang, 2019) which uses a Generative Adversarial Network (GAN) trained with adversarial samples to eliminate any adversarial perturbation of an input image, iii) Certified Defense (Raghunathan, Steinhardt, & Liang, 2018) which proposes a new differentiable upper bound yielding a model certificate ensuring that no attack can cause the error to exceed a specific value and iv) (O. F. Tuna, Catak, & Eskil, 2022a) which uses several uncertainty metrics for detecting adversarial samples.

2.3.4 Detection of Adversarial Samples

Kernel density and quantified epistemic uncertainty were the two indications used by Feinman et al. (Feinman et al., 2017) to identify adversarial samples. The variance of a Bayesian distribution obtained from a deep learning model with dropout was used to calculate the uncertainty estimate. Utilizing the activations of last hidden layer, the kernel density estimate score was computed. Nonetheless, aside from epistemic uncertainty, no other uncertainty metrics have been investigated, and calibrating the bandwidth for the kernel density estimation approach is a critical issue. For the purpose of detecting adversarial samples, Ma et al. (Xingjun Ma, 2018) suggested using an auxiliary classifier that has been trained to employ the expansion-based metric known as local intrinsic dimensionality. Metzen et al. (Metzen, Genewein, Fischer, & Bischoff, 2017) suggested enhancing a DNN with a detector subnetwork trained on the task of binary classification of normal and adversarial samples. Yang et al. (Yang, Chen, Hsieh, Wang, & Jordan, 2020) introduced ML-Loo, a technique for detecting adversarial samples by thresholding a scale estimate of feature attribution scores from Leave-One-Out (LOO).

CHAPTER 3

UNCERTAINTY IN MACHINE LEARNING

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

3.1 Types of Uncertainty in Machine Learning

In ML, there are two main kinds of uncertainty: aleatoric and epistemic uncertainty (An et al., 2020; Hüllermeier & Waegeman, 2021; Zheng, Zhang, Liu, Luo, & Sun, 2021). 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 (Antonelli et al., 2020). As shown in Figure 3.1, 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 $\mathscr{H}$ 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 he/she 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 $\mathscr{H}$. 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 $\mathscr{H}$, which is referred to as model uncertainty.

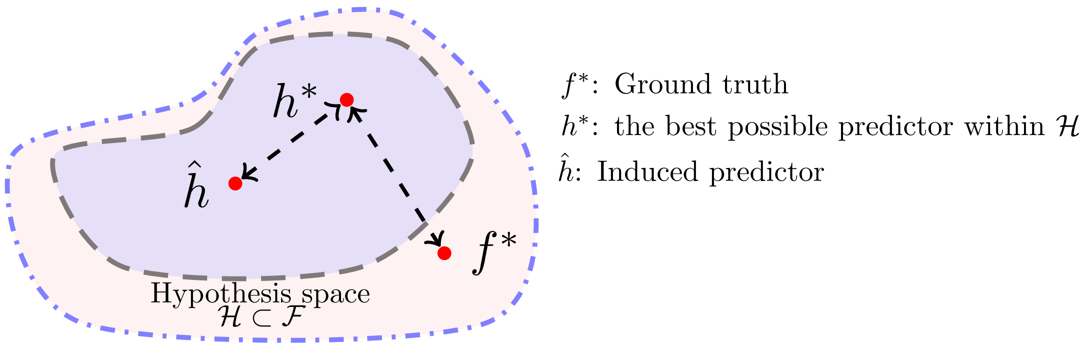

h^*
$\hat{h}$
f^*
Hypothesis space
$\mathcal{H} \subset \mathcal{F}$
f^*: Ground truth
h^*: the best possible predictor within $\mathcal{H}$
$\hat{h}$: Induced predictor

Figure 3.1 Different types of Epistemic Uncertainty.

The Universal Approximation Theorem, on the other hand, showed us that any target function f can be approximated by a neural network (Cybenko, 1989; Zhou, 2018). For deep neural networks, the hypothesis space $\mathscr{H}$ 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 (Loquercio, Segu, & Scaramuzza, 2020). 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 (Gurevich & Stuke, 2019). Despite having adequate training examples, this form of uncertainty cannot be reduced (Senge et al., 2014). The noise observed in a sensor's measurement data is an excellent example of this phenomena. Furthermore, aleatoric uncertainty may be divided into heteroscedastic uncertainty and homoscedastic uncertainty (Collier, Mustafa, Kokiopoulou, Jenatton, & Berent, 2020). Heteroscedastic uncertainty varies across various model inputs. Homoscedastic uncertainty, on the other hand, is insensitive to diverse inputs, which means it is consistent regardless of input.

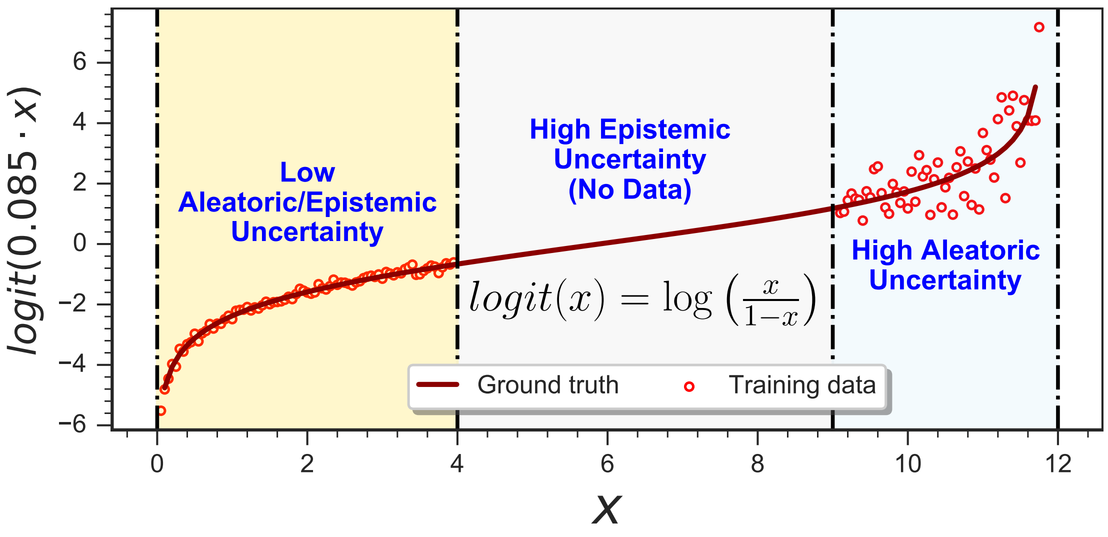


Figure 3.2 Illustration of the Epistemic and Aleatoric uncertainty.

A simple nonlinear function ($logit(0.085 \times x)$ in the interval $x \in [0, 12]$) is presented in Figure 3.2. Noisy samples are illustrated in the region at right where ($9 < x < 12$), and those samples leads 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 doesn't have enough training examples to accurately represent the data.

3.1.3 Scibilic Uncertainty

Reinhold et al.(Reinhold et al., 2020) 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 isn't caused by the model's difficulty for that particular input.

3.2 Quantifying Uncertainty in Deep Neural Networks

In recent years, many research studies have been conducted to quantify uncertainty in deep learning models. Most of the work was based on Bayesian Neural Networks, which learn the posterior distribution over weights to quantify predictive uncertainty (Hinton & Neal, 1995). However, the Bayesian NN's come with additional computational cost and inference issue. Therefore, several approximations to Bayesian methods have been developed which make use of variational inference (Blundell, Cornebise, Kavukcuoglu, & Wierstra, 2015; Graves, 2011; Hoffman, Blei, Wang, & Paisley, 2013; Paisley, Blei, & Jordan, 2012). On the other hand, Lakshminarayanan et al. (Lakshminarayanan, Pritzel, & Blundell, 2017) used the deep ensemble approach as an alternative to Bayesian NN's to quantify predictive uncertainty. But this approach requires training several NN's which may not be feasible in practice. A more efficient and elegant approach was proposed by Gal et al. The authors showed that a neural network model with inference time dropout is equivalent to a Bayesian approximation of the Gaussian process. In the following subsections, we will briefly mention their approach for quantifying different types of uncertainties, and then describe the enhanced version of quantifying uncertainties in DNN-based classifier models. Before that, we first introduce the notation as follows:

Given a dataset $\mathscr{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\} \subset X \times Y$ where $X \in \mathbb{R}^m$ is a m-dimensional input vector and $Y \in \{1 \cdots C\}$ where C is the output labels, training instances $\{\mathbf{x}_i, y_i\} \forall X$ are a set of independent and identically distributed (i.i.d.) training samples from some unknown probability measure P on $X \times Y$. The objective is to find a hypothesis $h : X \mapsto Y$ as close as possible to the original function that has generated the labels using weights of neural network architecture w and a loss function $\mathscr{L}$.

3.2.1 Quantification of Epistemic Uncertainty via MC-Dropout Sampling

Gal et al. (Gal & Ghahramani, 2016) demonstrated that using a neural network with dropout in inference time is identical to a specific variational inference on a Bayesian neural network model. The uncertainty of hypothesis model is approximated by averaging probabilistic feed-forward Monte Carlo dropout sampling throughout inference time.

It functions as an ensemble strategy. In each individual ensemble model, the system should discard different neurons in each layer of the network based on the dropout ratio in inference time. The average of the predictions made throughout dropout rounds is known as the predictive mean and the predictive mean is used as the final inference, $\hat{y}$, for the input sample $\hat{\mathbf{x}}$. The overall prediction uncertainty is approximated by finding the entropy and the variance of the probabilistic feed-forward Monte Carlo dropout sampling during prediction time. The prediction is defined as follows:

$$p(\hat{y} = c|\hat{\mathbf{x}}, \mathscr{D}) \approx \hat{\mu}_{pred} = \frac{1}{T} \sum_{y \in T} p(\hat{y}|\theta, \mathscr{D}) \tag{3.1}$$

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

Figure 3.3 illustrates a high-level overview of the Monte Carlo dropout based classification algorithm in the prediction time. In the prediction time, random neurons in each layer are dropped out, according to the p, from the base neural network model to create another model. As a result, T different classification models can be used for the prediction of the input instance's class label and uncertainty quantification of the overall prediction.

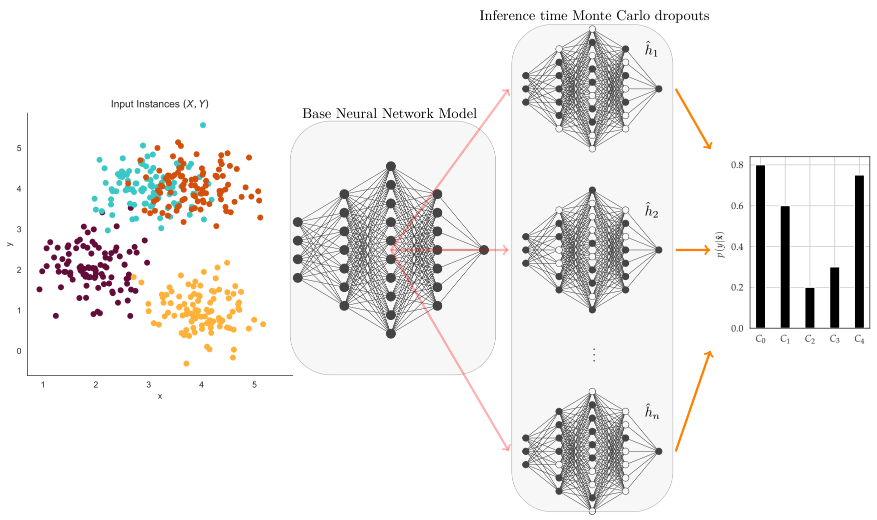


Figure 3.3 An example of Monte Carlo dropout based prediction.

For each testing input sample $\mathbf{x}$, the predicted label is assigned with the highest predictive mean. And the variance of the $p(\hat{y})$ is can be used as a measure of epistemic uncertainty of the model.

3.2.2 Quantification of Aleatoric Uncertainty via MC-Dropout Sampling

Let's assume that we would like to fit a regression model to some linear function and we employ a simple mean squared error (MSE) as our objective loss function for that regression model. In this scenario, as suggested by Kendal et al. (Kendall & Gal, 2017), we can replace our loss function as below :

$$Loss = \frac{||y - \hat{y}||_2}{2\sigma^2} + \frac{1}{2}\log\sigma^2 \tag{3.2}$$

Now, our model not only predicts $\hat{y}$, but also σ. From this updated loss function, we can see that when our model's prediction is very erroneous, loss will be attenuated by increased uncertainty σ^2. And we avert uncertainty from becoming too large by the term $\log\sigma^2$. Similarly, if the model can easily predict the actual value of the target, it should estimate a low variance term on that to reduce the loss.

Therefore, like in the above example, if we contain aleatoric uncertainty term in our loss function, our model will be predicting with high variance for data points residing in the areas where training data samples were noisy.

If we would like to train a classification model instead, then the loss would be as follows:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{y}} + \varepsilon_t \quad \varepsilon_t \sim \mathscr{N}\left(\mathbf{0}, \mathrm{diag}\left(\hat{\sigma}^2\right)\right)$$

$$\mathscr{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{T}\sum_{t=1}^{T} \text{Cross entropy } (\mathbf{x}, \hat{\mathbf{x}}_t)$$

The logic behind the above formulation is that when the model can easily predict the actual class for any input sample, the softmax output score for the predicted class will be high, and the model should estimate a low variance to reduce the extra noise. If, on the other hand, the model cannot easily predict the actual class, the softmax output probability score for the predicted class will be low. Therefore, introducing noise can raise the prediction of actual class by chance, hence lowering the value for the loss function.

3.2.3 Quantification of Epistemic and Aleatoric Uncertainty via MC-Dropout Sampling

Later, Kendall and Gal (Kendall & Gal, 2017) presented a technique in which both epistemic and aleatoric uncertainties are captured in a single model as shown in Figure 3.4. 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:

$$\underbrace{\frac{1}{T}\sum_{t=1}^{T} diag(\hat{\sigma}^2)}_{aleatoric} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}-\bar{y})^{\otimes 2}}_{epistemic} \tag{3.3}$$

where T denotes 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 $y^{\otimes 2} = yy^T$

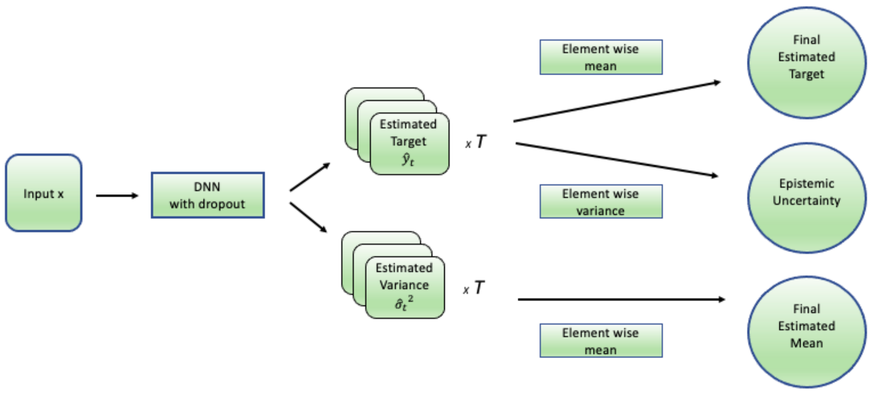


Figure 3.4 Quantification of Epistemic and Aleatoric Uncertainty via MC-Dropout Sampling.

3.2.4 Moment-Based Predictive Uncertainty Quantification

The method described above by Kendal and Gal 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).

In (Kwon, Won, Kim, & Paik, 2020), Kwon et al. proposed an alternative way of capturing both aleatoric and epistemic uncertainty for a classification model. The prediction variance is composed of two terms representing aleatoric and epistemic uncertainty in their approach, respectively. Let $\hat{\omega}$ represents parameters (learnt weights) used in the neural network, the number of different output classes is denoted by K, then the prediction y^* of a model for any test sample x^* given the weights of the model is denoted by $p(y^*|x^*, \hat{\omega})$ where $y^* \in \mathbb{R}^k$. The formulation for their method is given below:

$$Var_{p(y^*|x^*,\omega)}(y^*) = \mathbb{E}_{p(y^*|x^*,\omega)}(y^{*\otimes 2}) - \mathbb{E}_{p(y^*|x^*,\omega)}(y^*)^{\otimes 2} \tag{3.4}$$

$$Var_{p(y^*|x^*,\omega)}(y^*) = diag\{\mathbb{E}_{p(y^*|x^*,\omega)}(y^*)\} - \mathbb{E}_{p(y^*|x^*,\omega)}(y^*)^{\otimes 2} \tag{3.5}$$

$$\underbrace{= \frac{1}{T}\sum_{t=1}^{T}[diag\{p(y^*|x^*, \hat{\omega}_t)\} - p(y^*|x^*, \hat{\omega}_t)^{\otimes 2}]}_{aleatoric} \tag{3.6}$$

$$\underbrace{+\frac{1}{T}\sum_{t=1}^{T}\{p(y^*|x^*, \hat{\omega}_t)\} - \hat{p}(y^*|x^*, \hat{\omega}_t)^{\otimes 2}}_{epistemic} \tag{3.7}$$

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

Both of the above equations (3.6 and 3.7) output a matrix of shape $k \times k$ where the diagonal elements represents variance of each output class and we used the mean of the diagonal terms for quantifying uncertainty metrics for a given input $\mathbf{x}^*$.

Once the quantification of epistemic and aleatoric uncertainty is over, we simply calculate scibilic uncertanty as below:

$$Scibilic = Epistemic\,/\,Aleatoric \tag{3.8}$$

Eventually, for a given input $\mathbf{x}^*$, we have three different coulumn 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.

Figure 3.5 presents the general overview of the Monte Carlo dropout based classification algorithm. In the prediction time, random neurons in each layer are dropped out (based on the dropout ratio p) from the base neural network model to create a new model. As a result, T different classification models can be used to quantify uncertainty of the overall prediction. That is, the variance of the $p(\hat{y})$ is used to quantify epistemic uncertainty.

We have chosen the MC dropout method due to its simplicity and effectiveness. Although it requires a certain number of feed-forward queries, it is still more efficient than using Bayesian Networks or variational inference techniques which compute or approximate the posterior distribution of weights to quantify predictive uncertainty. The approach needs only a single trained model to measure the prediction's uncertainty, while other techniques such as Deep Ensemble need multiple models. Secondly, since the function used to calculate the variance is convex and smooth (Liu et al., 2019), one can take the backward derivative of the computed variance term for each input instance and use it to craft adversarial examples to evade the model.

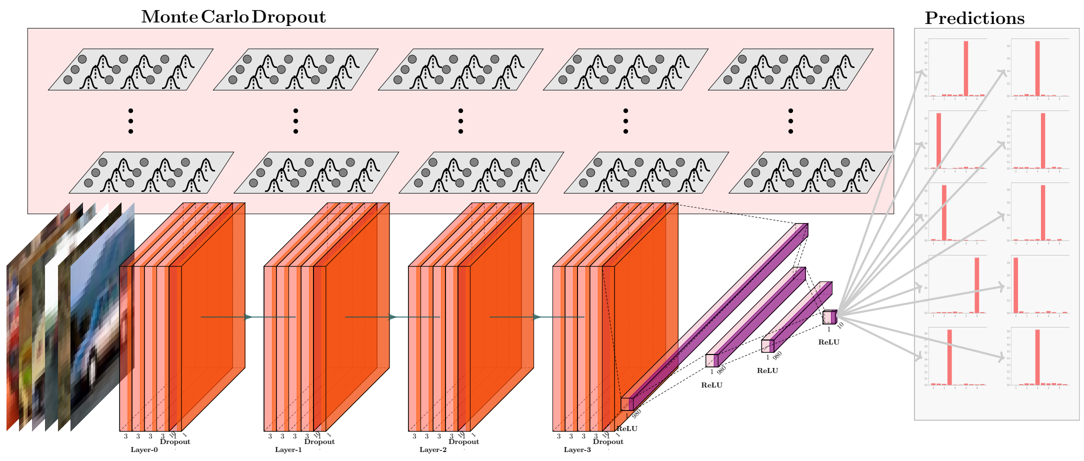


Figure 3.5 Illustration of the Monte Carlo dropout based Bayesian prediction.

CHAPTER 4

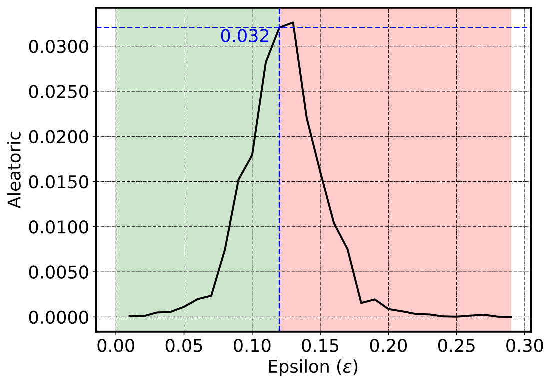
ADVERSARIAL SAMPLE DETECTION

4.1 Uncertainty Quantification

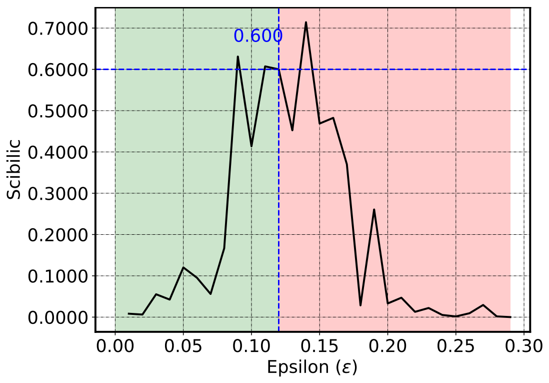
For the quantification of uncertainty metrics, we used Equation 3.6 for Aleatoric uncertainty, Equation 3.7 for Epistemic Uncertainty, Equation 3.8 for Scibilic Uncertainty.

4.1.1 Explanatory Research on Uncertainty Quantification Methods

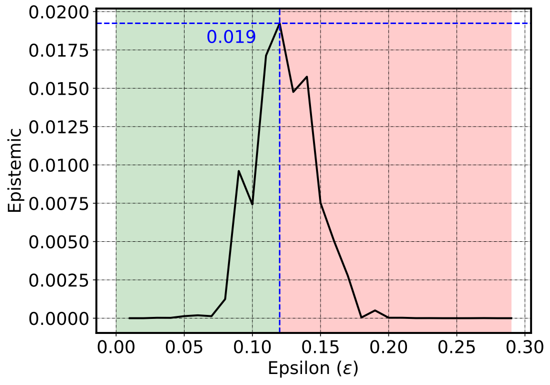
We have made a simple test on a sample image to visualize how the uncertainty metrics behave under an adversarial attack with varying strengths. Figure 4.1 shows how the uncertainty was affected under BIM attack to our model with different allowed perturbation amounts, ε. We can see that all the quantified uncertainty indicators increase as the amount of perturbation applied to the image becomes high enough to fool the classifier. Indeed, almost maximum epistemic uncertainty, aleatoric uncertainty and entropy values are observed at an ε value where the model starts mispredicting the input class. We can name this interval as ”low confidence interval”. The uncertainty metrics can be useful to differentiate the clean samples from their adversarial counterparts within this interval. Moreover, when the amount of perturbation used to fool the model is high, the model start to predict wrong class even more confidently, resulting a decrease in uncertainty estimates and thus they are not so reliable for detection as these metrics can not act as a separator anymore. We can call this interval as ”high confidence interval”. For those cases, we need an additional indicator to help us to increase detection accuracy scores. To overcome this problem, we used another metric which we call ”closeness score for predicted class” obtained from the activations of the last hidden layer.



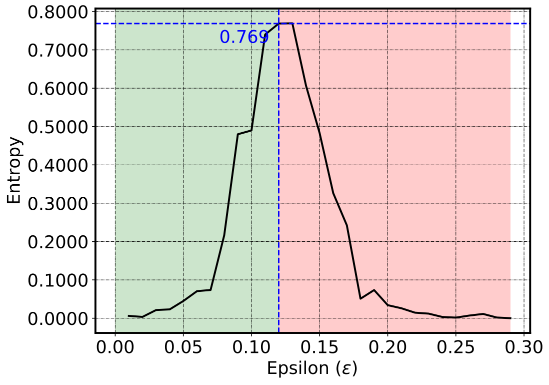
(a) Aleatoric



(b) Scibilic



(c) Epistemic



(d) Entropy

Figure 4.1 Change of uncertanty-based metrics under BIM attack with different amount of maximum allowed perturbation (ε) values.

4.2 Proposed Closeness Metric

Apart from the uncertainty metrics, one other possible way to understand the underlying mechanism of adversarial machine learning vulnerabilities, is to look at the manifold (low dimensional areas where the input data distribution is found) of the data used in the model training phase. High dimensional data like images are known to lie on low dimensional data manifold (Lee & Verleysen, 2007). And the manifolds of source classes which are the representations of the input instance in lower dimensional space become more linear and easy to work with as we go to the deep layers of DNN’s (Bengio, Mesnil, Dauphin, & Rifai, 2013). For this reason, we opted to work in the feature space of the last hidden layer activations. We used a tricky and trivial approach to understand the closeness of an input instance to the manifold which the predicted class of the input is represented. We grouped all the last hidden layer output vectors of clean, noisy and perturbed images of each class together and tried to train a secondary model so that all these lower dimensional representations of input instances actually correspond to the same class of input. By doing so, we could let our secondary model learn that the representation of any perturbed image in lower dimensional space should virtually be close to the its original class manifold rather than its wrongly predicted class manifold. The details of our approach is as follows:

Let $\mathscr{D} = \{(\mathbf{x}, y) | \mathbf{x} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^m\}$ be the training set for our CNN classifier (H_{cnn}) consisting of all the clean samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$, and their corresponding actual labels $\{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m\}$.

We first apply a noise with normal distribution to all the samples in training set as below:

$$\eta = \mathscr{N}(0, \varepsilon) \tag{4.1}$$

$$x^{\text{(noisy)}} = x + \eta, \quad \eta \in \mathbb{R}^{m \times n} \tag{4.2}$$

We then apply adversarial attack $\mathscr{F}$ (we used BIM) to all the training samples in our original training set with the same ε value that we used in crafting noisy samples. And we get perturbed samples as in below where δ is the perturbation amount derived from the attack algorithm:

$$\delta = \mathscr{F}(\mathbf{x}) \tag{4.3}$$

$$x^{\text{(pert)}} = x + \delta, \quad \delta \in \mathbb{R}^{m \times n} \tag{4.4}$$

After we increase the number of our training samples with crafted noisy and perturbed samples, we feed all these samples to our CNN classifier H_{cnn} to get their corresponding last hidden layer activation outputs v:

$$v^{\text{(clean)}} = H_{cnn}(\mathbf{x}^{\text{(clean)}}) \tag{4.5}$$

$$v^{(\text{noisy})} = H_{cnn}(\mathbf{x}^{(\text{noisy})}) \tag{4.6}$$

$$v^{(\text{pert})} = H_{cnn}(\mathbf{x}^{(\text{pert})}) \tag{4.7}$$

Then, we combine all the last hidden layer activation outputs in one pool to get $\mathbf{V} \in \mathbb{R}^{3 \cdot m \times j}$as in below, where j is the dimension of the last hidden layer for the CNN model:

$$\mathscr{V} = v^{(\text{clean})} \cup v^{(\text{noisy})} \cup v^{(\text{pert})} \tag{4.8}$$

The corresponding labels for these $\mathrm{v}^{(\text{clean})}$, $\mathrm{v}^{(\text{noisy})}$ and $\mathrm{v}^{(\text{pert})}$ vectors will all be $\mathrm{y}^{(\text{clean})}$. Because, we would like to teach our MLP model that all these vectors represent source class distribution in the sub-space of last hidden layer's activations and therefore they correspond to actual class labels of the inputs x from which they are derived from. Thus, we just concatenate $\mathrm{y}^{(\text{clean})}$ vector multiple times with itself to get $\mathbf{Y} \in \mathbb{R}^{3 \times m}$

$$\mathscr{Y} = y^{(\text{clean})} + y^{(\text{clean})} + y^{(\text{clean})} \tag{4.9}$$

Finally, using the new training set (V, Y) obtained from the last hidden layer activation outputs of clean, noisy and perturbed samples, we train our MLP model $H_{mlp} : \mathscr{V} \mapsto \mathscr{Y}$.

When the training of the MLP model is over, for any test image x_{test}, we can get the last hidden activations v_{test} of the CNN model and then feed it to our MLP model to get the softmax score outputs. Softmax score output vector ($\mathscr{O}$) of the MLP model will be of shape k, where k is the number of classes for our original training data. And we use the value at the index of the predicted label ($pred$) for the CNN model which is $\mathscr{O}[pred]$ as our last metric value for detecting adversarial samples.

4.2.1 Explanatory Research on our Closeness Metric

In Figure 4.2, we show the effectiveness of our proposed method. The y-axis shows the softmax prediction scores of the MLP model for the predicted class of the CNN model. MLP model was already trained on the last hidden layer activations of the CNN model for clean, noisy and perturbed samples. Since it learnt to map all the training samples (last hidden layer activations) into their related actual class labels accurately, even the CNN classifier is fooled and predicts a wrong class for a deliberately perturbed sample, the MLP model still predicts correct output given the last hidden layer activations of the CNN model for that perturbed sample.

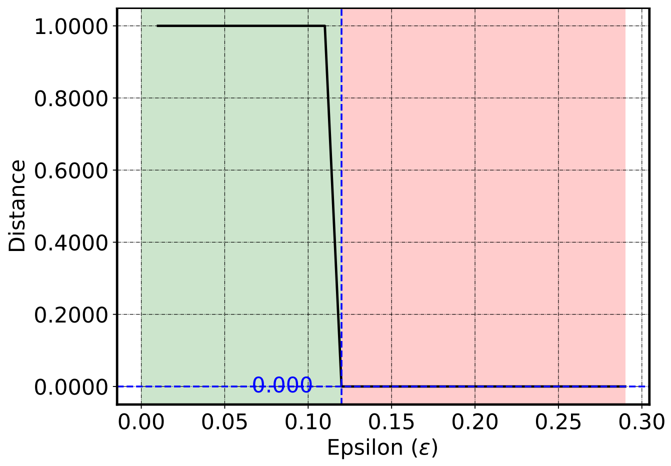


Figure 4.2 Change of prediction softmax score of MLP model obtained from last hidden layer activation of the CNN model under BIM attack with different amount of maximum allowed perturbation (ε) values.

In the figures, the green regions represent the areas of correct prediction and the red regions represent the areas of wrong prediction for the CNN model. Beginning with *eps* value of 0.12, the CNN model is fooled and starts to make wrong prediction. However, the MLP model prediction score decrease to 0 for the predicted class of the CNN model. This knowledge is already thought to the MLP model during its training. Thus, it successfully distinguishes the last hidden layer activation output of the perturbed sample as it is closer to the original class data distribution in the latent space rather than target class distribution and the predicted softmax score tends to rapidly decrease to zero for the wrongly predicted class of the CNN model.

4.2.2 Summary of the Algorithm

We provide the summary of our algorithm for adversarial sample detection in Figure 4.3. The overall process consists of five different stages. The first two stages are for the preparation of the helper MLP model which will then be used to compute the closeness score between the last hidden layer activations of the input sample and the lower dimensional representation of our CNN model's predicted class. Stage three is performed for the dataset preparation of our Logistic Regression (LR) classifier and this dataset, which consists of five different features, is then used to train the LR model. And in the final stage, our trained LR model is used for adversarial sample classification. Further details of our algorithm are explained in Section 5.7.2.

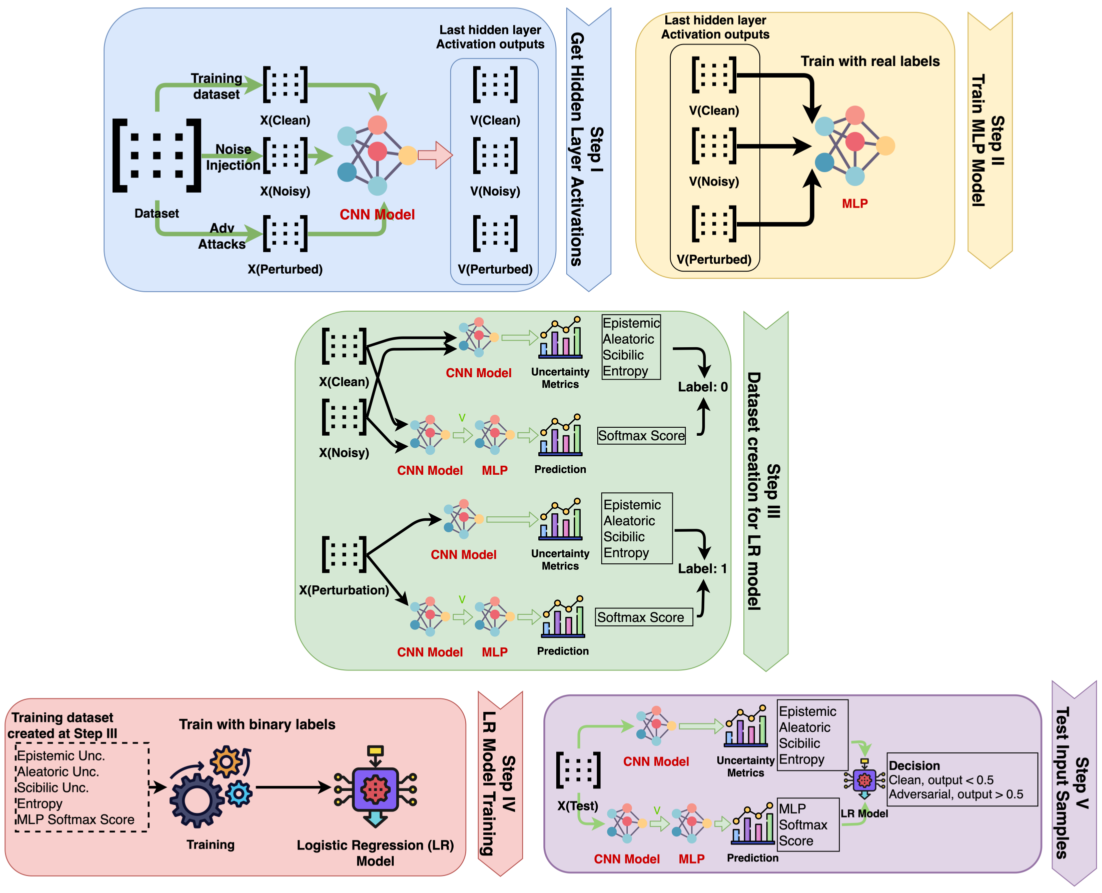


Figure 4.3 Overview of our algorithm.

4.3 Results

4.3.1 Experimental Setup

We trained our CNN models for the MNIST (Digit) (LeCun & Cortes, 2010), MNIST (Fashion) (Xiao, Rasul, & Vollgraf, 2017) and CIFAR10 (Krizhevsky, Nair, & Hinton, n.d.) datasets, and we achieved accuracy rates of 99.10 % , 91.52 % and 80.79 % respectively. The model architectures are given in Table 4.1 and the hyperparameters selected in Table 4.2. For training a classifier using last hidden layer activations of the CNN Models, we used simple Multilayer Perceptron (MLP) models with 2-hidden layers which are detailed in Table 4.3. The hyperparameters applied for these MLP models are shown in Table 4.4. In addition to the clean data, the noisy and perturbed samples which are used to train the MLP models are crafted using *eps* values of 0.2, 0.07 and 0.03 for MNIST Digit, MNIST Fashion and CIFAR datasets respectively. Finally, we used $T = 50$ as the number of MC dropout samples when quantifying uncertainty metrics.

Table 4.1 CNN model architectures

Dataset	Layer Type	Layer Information
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 10$
	Convolution (padding:1) + ReLU	$3 \times 3 \times 20$
	Dropout	p : 0.5
	Dense + ReLU	2880×128
	Dropout	p : 0.5
	Dense + ReLU	128×10
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 : 0.25
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.25
	Dense + ReLU	3136×600
	Dropout	p : 0.5
	Dense + ReLU	600×128
	Dense + ReLU	128×10
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
	Dropout	p : 0.5
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Max Pooling (Stride 2)	2×2
	Dense + ReLU	4096×1024
	Dropout	p : 0.5
	Dense + ReLU	1024×256
	Dropout	p : 0.5
	Dense + ReLU	256×10

Table 4.2 CNN model parameters

Parameter	MNIST (Digit)	MNIST (Fashion)	CIFAR10
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch Size	64	64	128
Dropout Ratio	0.5	0.25	0.5
Epochs	10	10	50

Table 4.3 MLP model architectures

Dataset	Layer Type	Layer Information
MNIST (Digit)	Fully Connected + ReLU	128×512
	Dense + ReLU	512×1024
	Dense + ReLU	1024×128
	Dense	128×10
MNIST (Fashion)	Fully Connected + ReLU	128×512
	Dense + ReLU	512×1024
	Dense + ReLU	1024×512
	Dense	512×10
CIFAR10	Fully Connected + ReLU	256×512
	Dense + ReLU	512×1024
	Dense + ReLU	1024×512
	Dense	512×10

Table 4.4 MLP model parameters

Parameter	MNIST (Digit)	MNIST (Fashion)	CIFAR10
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch Size	128	128	128
Epochs	50	50	150

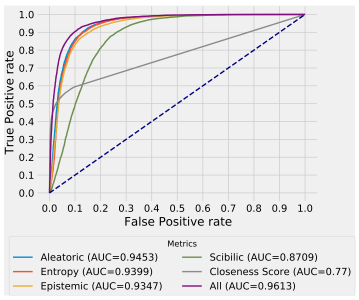
4.3.2 Experimental Results

To evaluate the performance of different metrics for adversarial detection, we have implemented each of the 5 attacks (FGSM, BIM, PGD, CW and Deepfool) with different allowed perturbation amounts (ε) under l_{inf} norm on MNIST (Digit), MNIST (Fashion) and CIFAR10 test data. Just for CW attack, we used the l_2 norm equivalent of the applied perturbation by using the formula $l_2 = l_{inf} \times \sqrt{n} \times \sqrt{2}/\sqrt{\pi e}$, where n is the input sample dimension. For the implementations of these attacks, we used a Python toolbox called Foolbox (Rauber, Brendel, & Bethge, 2018) and implement the attacks in their default settings. To be consistent with Feinman et al. (Feinman et al., 2017), we only perturbed those test samples which were accurately predicted by our models in their original states since an adversary would have no motive to perturb samples that are already misclassified. For each adversarial sample, we have also included normal and noisy counterparts in the pool as a benchmark. We craft noisy samples by applying Gaussian noise to each pixel with a scale similar to the adversarial samples. Then, all these normal, noisy and perturbed samples are used to train a Logistic Regression (LR) model to test the performance of our adversarial classifier. Adversarial samples are labeled as 1, which represents the positive class, whereas normal and noisy samples are labeled as 0, which represents the negative class. Total of five features which are computed for each sample in the pool before LR training are Epistemic Uncertainty, Aleatoric Uncertainty, Scibilic Uncertainty, Entropy and Closeness Score for predicted class. For MNIST (Digit) dataset, we showed ROC-AUC scores of our adversarial classifier in Figures 4.4 and 4.5. And the detailed results of our experiments are available in Tables 4.5 , 4.6 and 4.7. We aimed at evaluating the quality of our metrics under both medium and high level of adversarial threat by using different allowed perturbation amounts. We achieved almost perfect detection scores under high level of allowed perturbation (epsilon). And despite the risk of lowering attack success chance, if the intruder opts to chose a lower epsilon value for the attack, we again achieve very high degree of performance.

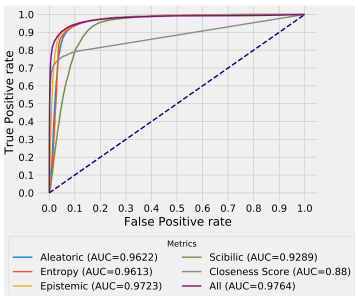
Table 4.5 Digit MNIST - Roc-Auc Scores of different metrics under various attack types and epsilon values

	eps = 0.12						eps = 0.30					
	Epis	**Ale**	**Scib**	**Ent**	**Dist**	**All**	**Epis**	**Ale**	**Scib**	**Ent**	**Dist**	**All**
FGSM	0.84	0.87	0.77	0.86	0.59	0.87	0.85	0.89	0.71	0.88	0.91	0.94
BIM	0.93	0.95	0.87	0.94	0.77	0.96	0.60	0.64	0.47	0.64	0.98	0.99
PGD	0.93	0.94	0.86	0.94	0.75	0.96	0.63	0.67	0.44	0.66	0.98	0.99
Deepfool	0.89	0.91	0.83	0.90	0.63	0.91	0.91	0.91	0.81	0.87	0.97	0.99
CW	0.97	0.96	0.93	0.96	0.88	0.98	0.98	0.90	0.94	0.91	0.98	1.00

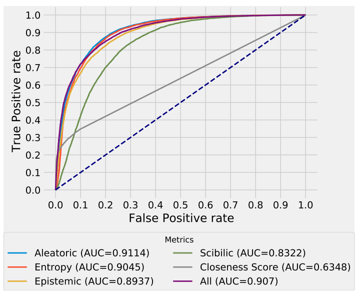
After verifying the effectiveness of our adversarial sample detection method on image domain with CNN architectures, we would like to test our method on other forms of input data and with a different model architecture. For this purpose, we opted to work on intrusion detection tasks and chose the KDD-Cup-99 Dataset from UCI ML repository for our additional experiment. We used %10 percent of the KDD99 dataset which contains around 494K data records. This dataset contains 42 features



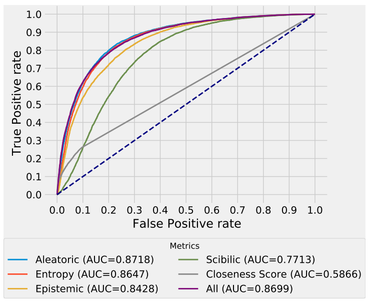
(a) BIM



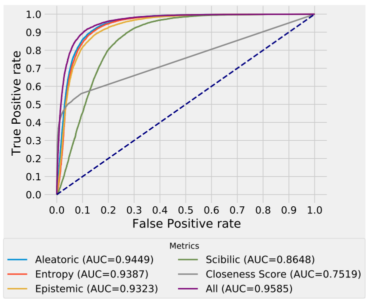
(b) CW



(c) DeepFool

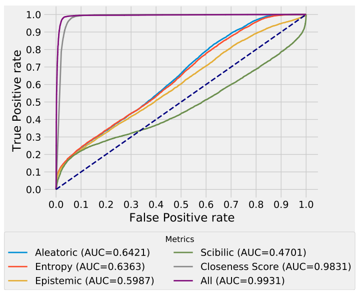


(d) FGSM

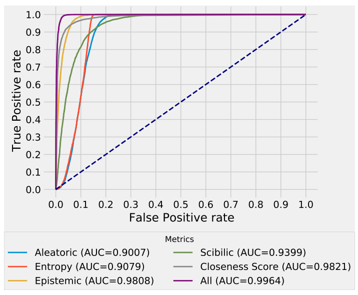


(e) PGD

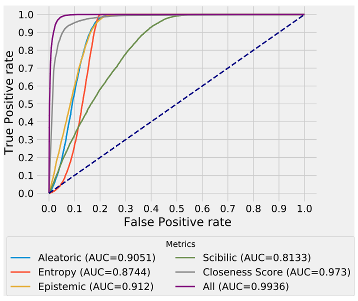
Figure 4.4 Roc-Auc Plots for MNIST - $\varepsilon = 0.12$



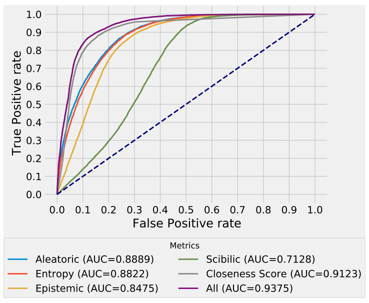
(a) BIM



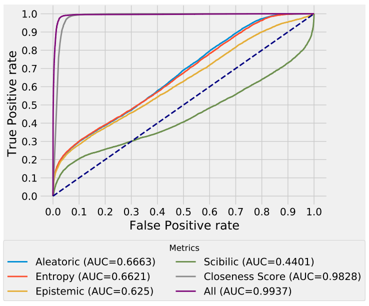
(b) CW



(c) DeepFool



(d) FGSM



(e) PGD

Figure 4.5 Roc-Auc Plots for MNIST - $\varepsilon = 0.30$

and 23 output classes and 3 of the input data features were categorical which we have removed. We've split %80 percent of the dataset for training and %20 percent of the dataset for test. After normalising the remaining features, we have trained a 5-layer standard DNN architecture. We used dropout with a rate of 0.5 for first and second

Table 4.6 Fashion MNIST - Roc-Auc Scores of different metrics under various attack types and epsilon values

	eps = 0.03						eps = 0.12					
	Epis	**Ale**	**Scib**	**Ent**	**Dist**	**All**	**Epis**	**Ale**	**Scib**	**Ent**	**Dist**	**All**
FGSM	0.77	0.76	0.77	0.76	0.61	0.78	0.80	0.77	0.79	0.79	0.76	0.86
BIM	0.77	0.72	0.78	0.74	0.74	0.85	0.69	0.72	0.33	0.72	0.99	0.99
PGD	0.78	0.74	0.79	0.75	0.71	0.84	0.71	0.73	0.69	0.73	0.99	0.99
Deepfool	0.89	0.85	0.88	0.86	0.76	0.90	0.95	0.89	0.91	0.90	0.93	0.98
CW	0.89	0.84	0.89	0.85	0.79	0.91	0.96	0.87	0.93	0.89	0.94	0.98

Table 4.7 CIFAR10 - Roc-Auc Scores of different metrics under various attack types and epsilon values

	eps = 0.02						eps = 0.04					
	Epis	**Ale**	**Scib**	**Ent**	**Dist**	**All**	**Epis**	**Ale**	**Scib**	**Ent**	**Dist**	**All**
FGSM	0.70	0.69	0.68	0.69	0.53	0.71	0.71	0.69	0.68	0.70	0.55	0.72
BIM	0.82	0.84	0.83	0.84	0.89	0.92	0.89	0.94	0.95	0.95	0.96	0.99
PGD	0.77	0.79	0.77	0.79	0.84	0.87	0.89	0.94	0.93	0.94	0.95	0.98
Deepfool	0.94	0.88	0.83	0.89	0.77	0.96	0.93	0.87	0.83	0.88	0.77	0.96
CW	0.94	0.87	0.85	0.88	0.79	0.97	0.93	0.85	0.84	0.86	0.80	0.96

inner-product layers. We trained our model for 5 epochs using Adam optimizer with a learning rate of 0.001 and achieved a test set accuracy of 99.60 % .Then, we used this trained standard DNN model and tested our adversarial sample detection method against different attack types with an ε value of 0.04. Table 4.8 shows the attack success rates of the attack algorithms and Roc-Auc detection scores of our proposed method. The results confirm once again the effectiveness of our proposed method.

Table 4.8 KDD-Cup-99 dataset - Roc-Auc detection scores of our method for various attack types

	Attack Success Rate	Roc-Auc Detection Score
FGSM	53,60%	98,92%
BIM	57,90%	99,32%
PGD	58,08%	99,35%
Deepfool	56,67%	99,52%
CW	67,30%	99,45%

We finally provide a comparison of our experimental results with Feinman et al. (Feinman et al., 2017). Results available in Table 4.9 shows that our method achieves better detection performance for each of the attacks on different datasets.

Table 4.9 Comparison of ROC-AUC detection scores

	MNSIT - Digit			CIFAR10		
	FGSM	BIM	CW	FGSM	BIM	CW
Feinman et al.	90,57%	82,06%	97,94%	72,23%	95,41%	92,17%
Ours	94,12%	99,07%	99,78%	72,41%	99,10%	96,29%

4.3.3 Further Results and Discussion

We have finally made an in depth series of experiments to see the performance of each of the metrics under the application of an adversarial attack (BIM) with different level of perturbation amounts. Figure 4.6 summarizes the results of our experiments. Results show that when the perturbation amount is low or moderate, the contribution of uncertainty metrics to adversarial detection performance is high. However, when we apply the attack with high level of perturbation, this time our closeness metric takes the lead and plays the key role. The reason why the closeness metric performs poorly under an attack with low perturbation amount is that for those cases the attack success rates are actually not so high and the attack barely succeeds in fooling the CNN Model. Therefore, the predictions of the CNN and MLP models are mostly the same and softmax output score of the MLP model for the predicted class can not act as a successful separator for clean and perturbed samples for the LR classifier. Ultimately, the combined usage of all the metrics is observed to be the best choice for securing the model prediction performance.

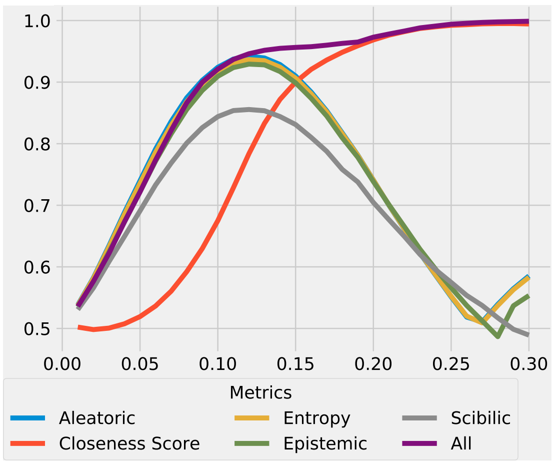


Figure 4.6 ROC-AUC scores of different metrics under an attack with varying level of allowed perturbation amounts ε

CHAPTER 5

ADVERSARIAL ATTACK

5.1 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 5.1 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 $y = -2x + 1$. 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 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.

$$\underset{\|\delta\| \leq \varepsilon}{arg\,max} \ell(F_\theta(x+\delta), y) \tag{5.1}$$

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.

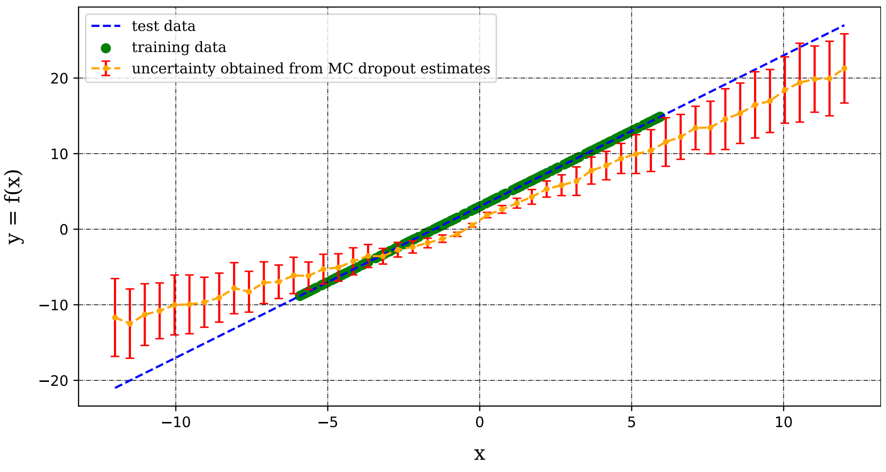
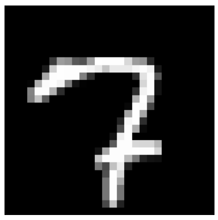
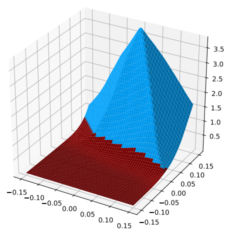


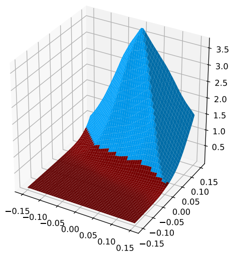
Figure 5.1 Uncertainty estimates obtained from a regression model

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 Figure 5.2. Figure 5.2b 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 Figure 5.2c, 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-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 5.2d. We reached a loss value of 4.167, which is bigger than the previous two attempts. In Figures 5.2b,5.2c and 5.2d, 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.

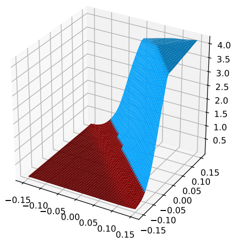

(a) MNIST Digit test image number 37



(b) Loss gradient direction



(c) Uncertainty gradient direction

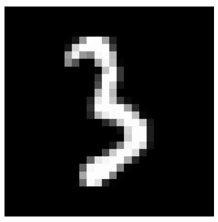


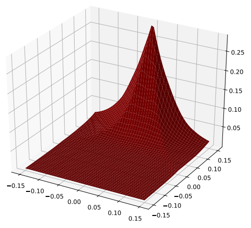
(d) Hybrid gradient direction

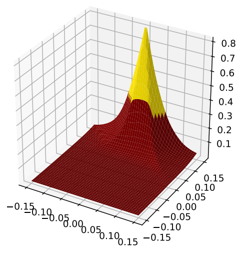
Figure 5.2 Loss surfaces in different directions. The maximum loss values are 3.783, 3.713, 4.167 for b, c and d respectively

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 (Blum & Rivest, 1992; Judd, 1990). 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 5.3 shows that the maximum loss value in uncertainty's gradient direction (0.811) is far greater than the maximum loss value in model loss' gradient direction (0.285). And the maximum loss value in the hybrid direction (0.845) 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 gradient direction, as there is no visible colour change in the surface plot of Figure 5.3b, whereas in Figure 5.3c 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.


(a) MNIST Digit test image number 45


(b) Loss gradient direction


(c) Uncertainty gradient direction

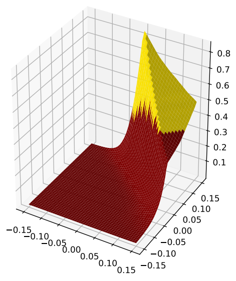

(d) Hybrid gradient direction

Figure 5.3 Loss surfaces in different directions. The maximum loss values are 0.285, 0.811, 0.845 for b, c and d respectively

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.

5.2 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 Figure 5.2 and Figure 5.3 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. In sections 5.2.1 - 5.2.4, we present our first group of proposed attack variants together with their original counterparts.

5.2.1 Fast Gradient Sign Method (Uncertainty-Based)

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

$$\mathbf{x}^{adv} = \mathbf{x} + \varepsilon \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}, y_{true})) \tag{5.2}$$

where $\mathbf{x}$ denotes input (clean) image, x^{adv} represents the perturbed adversarial image, ℓ is the classification loss function, y_{true} is the actual label for the input $\mathbf{x}$. And our modified FGSM attack (uncertainty-based) is shown as;

$$\mathbf{x}^{adv} = \mathbf{x} + \varepsilon \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}, F, p, T)) \tag{5.3}$$

where $\mathbf{x}$ denotes the input (clean) image, $\mathbf{x}^{adv}$ represents the perturbed adversarial image, U is the uncertainty metric (mean variance) derived from T different MC dropout estimates, F is the prediction model in training mode, p denotes the dropout ratio used in the dropout layers, T represents 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.

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

Step 2: The next step is to calculate 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: Calculate 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.

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

5.2.2 Basic Iterative Attack (BIM-A Uncertainty-Based)

In this section, we first provide the pseudo-codes for known loss-based BIM attack types as in Algorithm 5.1 and 5.2. Then, we provide our proposed uncertainty-based BIM attack variants in Algorithm 5.3 and 5.4. All the attack types proposed here are designed under L_∞ norm.

5.2.3 Basic Iterative Attack (BIM-A Hybrid Approach)

Here, we present the pseudo-code for our Hybrid Approach in Algorithm 5.5. 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 5.1: Loss-based BIM Algorithm (type-A)
x is the benign image, y_{true} is the true label for **x**, F is the deep learning model learnt in training, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
    Input: $\mathbf{x} \in \mathbb{R}^m, y_{true}, F, N, \alpha, \varepsilon$
    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       $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$
4       if $argmax(F(\mathbf{x}_{t+1})) \neq y_{true}$ then
5           end while
6   return $\mathbf{x}_{t+1}$
```

Algorithm 5.2: Loss-based BIM Algorithm (type-B)
x is the benign image, y_{true} is the true label for **x**, F is the deep learning model learnt in training, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
    Input: $\mathbf{x} \in \mathbb{R}^m, y_{true}, F, N, \alpha, \varepsilon$
    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       $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$
4   return $\mathbf{x}_{t+1}$
```

Algorithm 5.3: Uncertainty-based BIM Algorithm (type-A)
x is the benign image, F is the deep learning model learnt in training, p denotes the dropout ratio used in dropout layers, T represents the number of MC dropout samples in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
    Input: $\mathbf{x} \in \mathbb{R}^m, F, p, T, N, \alpha, \varepsilon$
    Output: $\mathbf{x}_{t+1}$
1   $\mathbf{x}_0 \leftarrow \mathbf{x}$
2   $initialprediction = argmax(F(\mathbf{x}_0))$
3   while $n < N$ do
        /* update X by using below formula, F in training mode
           */
4       $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)))$
5       if $argmax(F(\mathbf{x}_{t+1})) \neq initialprediction$ then
6           end while
7   return $\mathbf{x}_{t+1}$
```

Algorithm 5.4: Uncertainty-based BIM Algorithm (type-B)
x is the benign image, F is the deep learning model learnt in training, p denotes the dropout ratio used in dropout layers, T represents the number of MC dropout samples in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
    Input: $\mathbf{x} \in \mathbb{R}^m, F, p, T, N, \alpha, \varepsilon$
    Output: $\mathbf{x}_{t+1}$
1   $\mathbf{x}_0 \leftarrow \mathbf{x}$
2   while $n < N$ do
3       if $argmax(F(\mathbf{x}_{t+1})) \neq initialprediction$ then
4           $condition = True$
5       if $condition = False$ then
            /* update X by using below formula, F in training
               mode                                              */
6           $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)))$
7       else
            /* update X by using below formula, F in training
               mode                                              */
8           $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)))$
9   return $\mathbf{x}_{t+1}$
```

5.2.4 Basic Iterative Attack (BIM-B Hybrid Approach)

In the last part of this proposed algorithms section, we present the pseudo-code for our Hybrid Approach for BIM-B as in Algorithm 5.6. Like the other proposed BIM attack variants, our Hybrid Approach for BIM-B is also designed under ℓ_∞ norm. Until the perturbed input sample is pushed away from the model's decision boundary, we step into both the model loss' gradient direction and model uncertainty's gradient direction at each iteration. And after passing the decision boundary, we try to decrease the quantified uncertainty and also try to increase the loss at the same time. The success rates of Algorithm 5.5 and 5.6 are equal in the white-box setting. Therefore, we suggest to use this method while crafting black-box adversarial samples using attack transferability.

5.3 Visualizing Gradient Path for Uncertainty-Based Attacks

Figure 5.4 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

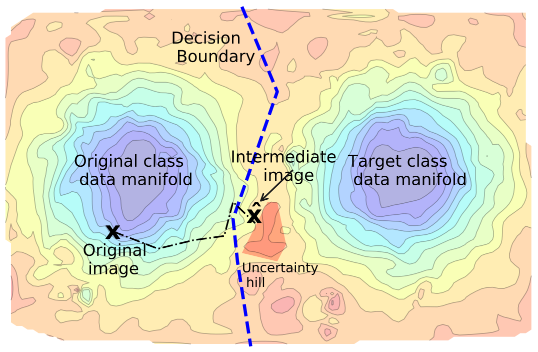
Algorithm 5.5: Algorithm for BIM A (Hybrid Approach)
$\mathbf{x}$ is the benign image, F is the deep learning model learnt in training, p denotes the dropout ratio used in dropout layers, T represents the number of MC dropout samples in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
   Input: $\mathbf{x}, F, p, T, N, \alpha, \varepsilon$
   Output: $\mathbf{x}_{t+1}$
1  $\mathbf{x}_0 \leftarrow \mathbf{x}$
2  $initialprediction = argmax(F(\mathbf{x}_0))$ /* 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      $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)) + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$
5      if $argmax(F(\mathbf{x}_{t+1})) \neq initialprediction$ then
6          end while
7  $return\ \mathbf{x}_{t+1}$
```

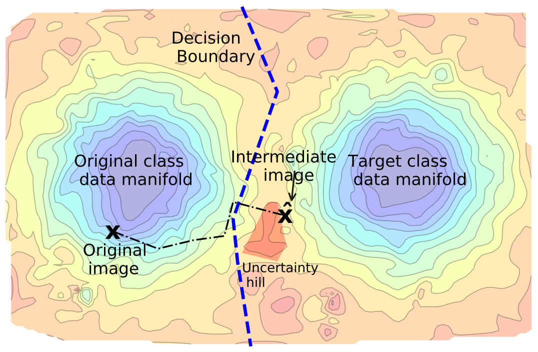
Algorithm 5.6: Algorithm for BIM B (Hybrid Approach)
$\mathbf{x}$ is the benign image, F is the deep learning model learnt in training, p denotes the dropout ratio used in dropout layers, T represents the number of MC dropout samples in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
   Input: $\mathbf{x}, F, p, T, N, \alpha, \varepsilon$
   Output: $\mathbf{x}_{t+1}$
1  $\mathbf{x}_0 \leftarrow \mathbf{x}$
2  $initialprediction = argmax(F(\mathbf{x}_0))$ /* 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      $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t + \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)) + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$
5      if $argmax(F(\mathbf{x}_{t+1})) \neq initialprediction$ then
6          $\mathbf{x}_{(t+1)} =$
             $clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, F, p, T)) + \alpha \cdot sign(\nabla_{\mathbf{x}} \ell(\mathbf{x}_t, y_{true})))$
7          end while
8  $return\ \mathbf{x}_{t+1}$
```

regions are shown in blue, while the high uncertainty regions are shown in red. Figure 5.4a 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 Figure 5.4b, 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.



(a) Type A success, Type B fail

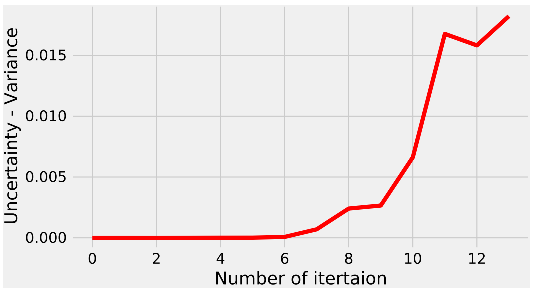


(b) Type A and B success

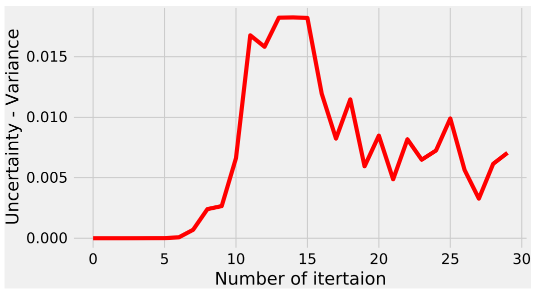
Figure 5.4 Uncertainty gradient path

5.4 Visualizing Uncertainty Under Different Attack Variants

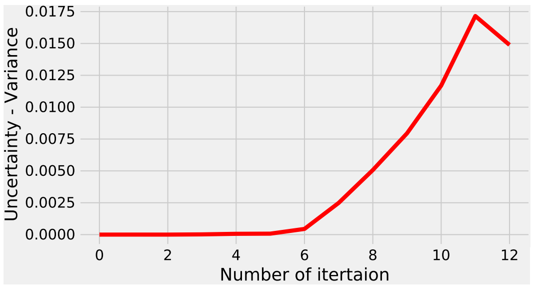
Figure 5.5 shows the change in our quantified uncertainty values of the model during different BIM attack variants. In this experiment, we applied all of the attack variants to the 23th test sample from MNIST (Digit) dataset. The original label of the input image was 6. For type A and B of loss and uncertainty based attacks, we observed that at 11th and 13th iterations, respectively, the attack is successful, and the input image was misclassified as 4. In Figure 5.5a, we stop the iteration as soon as we succeeded in fooling the model, whereas in Figure 5.5b, 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 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.



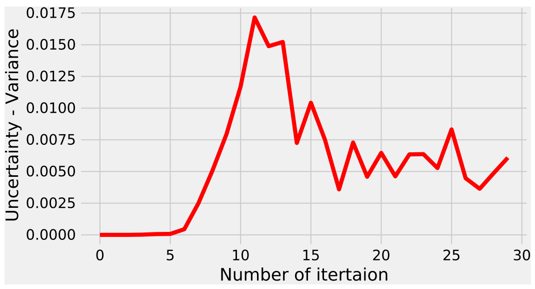
(a) BIM-A (Uncertainty Based)



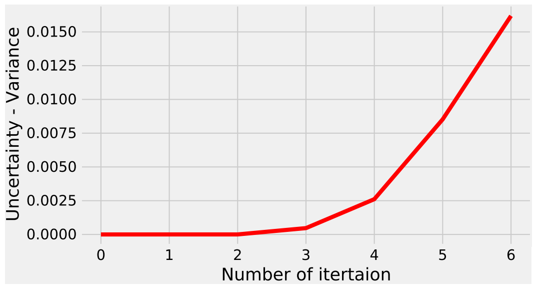
(b) BIM-B (Uncertainty Based)



(c) BIM-A (Loss Based)



(d) BIM-B (Loss Based)



(e) BIM-Enhanced (Hybrid Approach)

Figure 5.5 Change of uncertainty values during different BIM attack variants

5.5 Search For a More Efficient Attack Algorithm

So far, we demonstrated that we can increase the loss based attack performances by additionally using the quantified uncertainty information of the model. We have exceeded the attack success rates of traditional loss based attacks with our hybrid approach. However, one can argue that even the total perturbation amount is constrained to the same ε, the amount of perturbation applied upon the image might be higher than the loss based attack in case of our proposed hybrid attack variant. In an aim to do the better, we come up with a more efficient attack idea. In our new attack proposal, we will be using only a subset of directions based on some defined logic. We name our proposed attack algorithm as Rectified Basic Iterative Attack (Rectified-BIM). Because the direction pointed out by the gradient of the loss function is updated using the reference information obtained from the quantified epistemic uncertainty.

5.5.1 Rectified Basic Iterative Attack

Algorithm 5.7 shows our proposed uncertainty based Rectified-BIM attack. The algorithm is designed under L_∞ norm.

Algorithm 5.7: Algorithm for Rectified-BIM attack:
$\mathbf{x}$ is the benign image, y_{true} is the actual label for $\mathbf{x}$, h is the hypothesis function learnt during training, p denotes the dropout ratio used in dropout layers, T represents the number of MC dropout samples in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

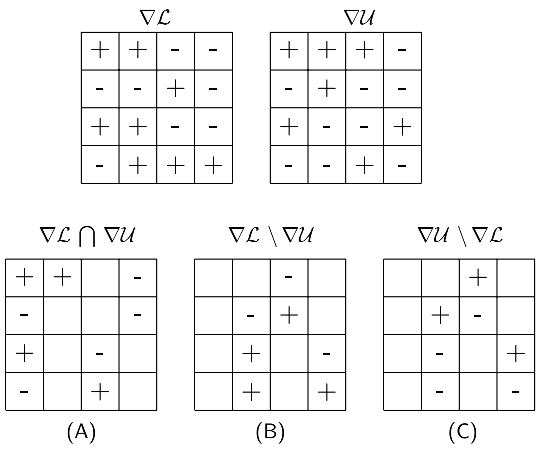
```
    Input: $\mathbf{x} \in \mathbb{R}^m, h, p, T, N, \varepsilon, \alpha$
    Output: $\mathbf{x}_{t+1}$
 1  $\mathbf{x}_0 \leftarrow \mathbf{x}$
 2  $condition \leftarrow False$
 3  while $n < N$ do
 4      Compute $\nabla_{\mathbf{x}}\ell(h(\mathbf{x}_t, y_{true}))$ while $h$ in evaluation mode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

At each iteration of the Rectified-BIM algorithm, we calculate $\nabla_{\mathbf{x}}\ell(h(\mathbf{x}_t, y_{true}))$ and $\nabla_{\mathbf{x}}U(\mathbf{x}_t, h, p, T)$. We then get the sign of these gradient vectors which show the sub-directions for each input dimension. If the sign of the gradient for any input pixel is positive, it means that we can increase loss or uncertainty by increasing the value of that pixel. For any input sample fed to the attack algorithm, we start perturbing the input image by first using the intersection of the sub-directions pointed by the derivative of loss and uncertainty information. This is valid until the input sample is pushed off of the decision boundary. Once the input sample passes the decision boundary, this time we use the sub-directions of loss' gradient which are not shared by the uncertainty's gradient. This idea can be understood better by looking at Figure 5.6 - (A) and Figure 5.6 (B). In the first part of the proposed attack: Instead of trusting only loss or only uncertainty, we trust the information provided by both. This way, we only use a subset of sub-directions. Thus, at each iteration, the amount of perturbation applied to the input sample is being decreased without compromising the adversarial attacking strength.

In the low confidence regions where the input sample is close to the model's decision boundary, the uncertainty has a friction effect against loss. It is known that the quantified uncertainty is higher near the decision boundaries (O. F. Tuna et al., 2022a). Thus, for adversarial attack purposes, perturbing the input sample in the direction of uncertainty's gradient is not a good idea after passing the decision boundary. This is because the direction in which the gradient of uncertainty points will keep the sample near the boundary regions and prevent the perturbed sample from being pushed far away. Therefore, for the second part of our Rectified-BIM Attack, we consciously decided not to use the common sub-directions which are shared by both loss' and uncertainty's gradients, instead we used the sub-directions which are left from the loss' gradient after discarding the common sub-directions as in Figure 5.6 - (B). By gradients, here we mean the sign of the gradient vectors.

5.6 Attacker's Capability

We hypothesized that the main objective of the attacker is to deceive the model by introducing cleverly crafted perturbation to the model input. In a practical case, the white-box setting is the most preferred option for the adversary. This demands the attacker to gain access to the model for generating adversarial samples. After capturing



$\nabla\mathcal{L}$
$\nabla\mathcal{U}$
$\nabla\mathcal{L} \bigcap \nabla\mathcal{U}$
$\nabla\mathcal{L} \setminus \nabla\mathcal{U}$
$\nabla\mathcal{U} \setminus \nabla\mathcal{L}$
(A)
(B)
(C)

Figure 5.6 Sub-directions (sign of the gradients) used in attack purposes (A,B)

model information, the attackers can leverage the vulnerabilities of the model to their advantage.

However, the adversary must solve an optimization problem to decide which regions of input data should be changed to prevent this manipulation from being easily noticed by the human eye. By resolving this optimization problem by employing one of the available attack algorithms (Aladag, Catak, & Gul, 2019; Goodfellow et al., 2015; Kurakin et al., 2017; Madry et al., 2018), the attacker seeks to lower the classification performance of the model on the adversarial data as much as possible. In this work, to restrict the maximum permissible perturbation for the attacker, we utilized ℓ_∞ norm, which is the maximum pixel difference limit between adversarial and benign images.

5.7 Results

5.7.1 Experimental Setup

We begin our experiments by assessing the performance of our first group of attack variants. For this purpose, we trained our CNN models for the MNIST (Digit) (LeCun & Cortes, 2010) and MNIST (Fashion) (Xiao et al., 2017) datasets, and we achieved accuracy rates of 99.05 % and 91.15 % respectively. For the CIFAR10 dataset (Krizhevsky et al., n.d.), we used a pretrained VGG-A (11 weight layers) model (Simonyan & Zisserman, 2015) 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 5.1. Finally, we used $T = 50$ as the number of MC dropout samples when quantifying uncertainty.

Table 5.1 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	$\alpha = \varepsilon \cdot 0.2$, i = 10 (for MNIST)	l_∞
BIM	$\alpha = \varepsilon \cdot 0.3$, i = 5 (for CIFAR-10)	l_∞

5.7.2 Experimental Results

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

The results shown in Table 5.2 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 maximization journey. Figure 5.7 shows some examples of adversarial samples crafted using different methods mentioned in this study.

Table 5.2 Attack success rates on different datasets

	MNIST (Digit) $\varepsilon = 0.15$	MNIST (Fashion) $\varepsilon = 0.05$	CIFAR10 $\varepsilon = .2/255$
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

In the last part of experimental results for this section, we tried to compare the success ratio of our hybrid attack variants in black-box setting. For this, we first trained different surrogate models using MNIST Digit, MNIST Fashion and also CIFAR10 datasets. Then, we tried to craft adversarial samples using both traditional approaches and also using our proposed algorithms. Later, we tested these adversarial samples on a different target models. As can be seen in Table 5.3 and 5.4, the success rates of our attack algorithms are higher which shows the effectiveness of our method in black-box setting as well.

Table 5.3 Comparison of attack transferability rates - part 1

	Attack transferability rates		
	MNIST DIGIT	MNIST FASHION	CIFAR10
	eps : 0.2	eps : 0.03	eps : 6.0/255
Bim-A (loss-based)	2.99%	9.03%	8.19%
Bim-A (hybrid approach)	6.35%	13.37%	11.57%
Deepfool	1.16%	5.01%	5.82%

Table 5.4 Comparison of attack transferability rates - part 2

	Attack transferability rates		
	MNIST DIGIT	MNIST FASHION	CIFAR10
	eps : 0.2	eps : 0.03	eps : 6.0/255
Bim-B (loss-based)	17.62%	27.26%	18.15%
Bim-B (hybrid approach)	19.16%	28.88%	20.30%

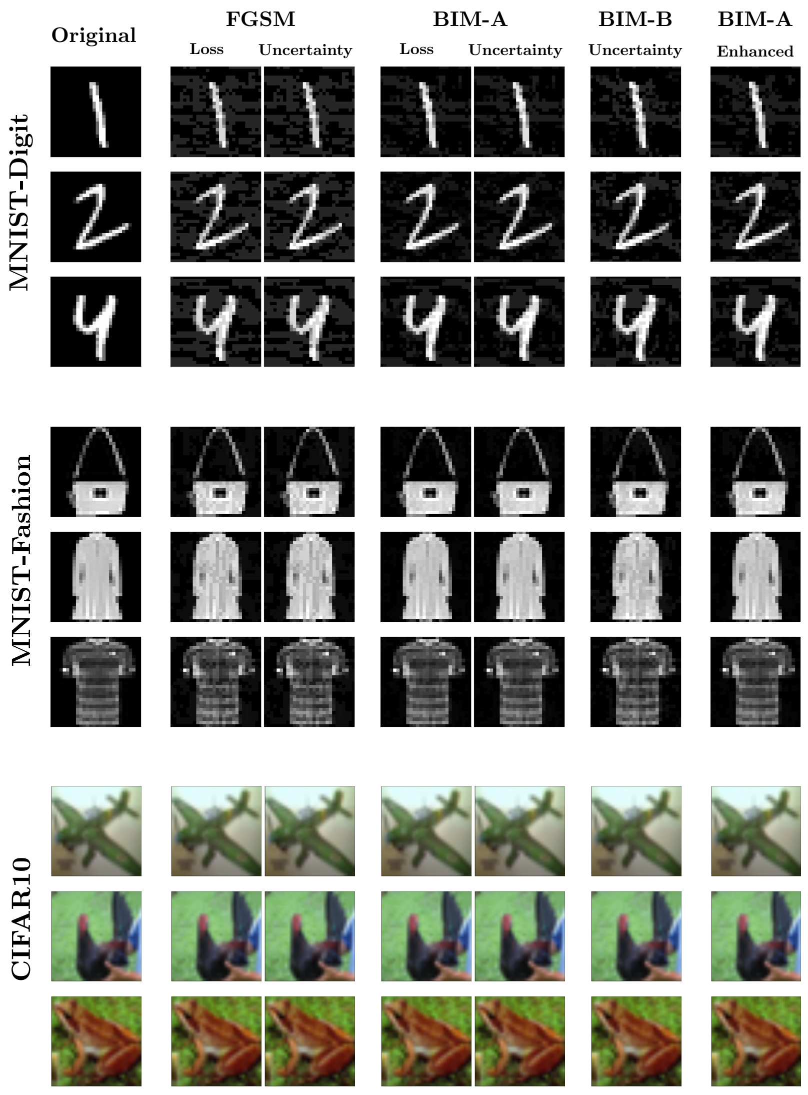


Figure 5.7 Some example images from MNIST(Digit), MNIST(Fashion) and CIFAR-10. The original image is shown in the left-most column and adversarial samples crafted based on different methods are on the other columns.

5.7.3 Further Results and Discussion

After verifying the performances of our first group of attack variants, we tried to assess the performance of second group of attack idea. For this purpose, we trained our CNN models for the MNIST (Digit) (LeCun & Cortes, 2010), MNIST (Fashion) (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., n.d.) datasets, and we achieved accuracy rates of 99.26 % , 92.63 % and 83.91 % respectively. For the CIFAR10 dataset we applied a normalization procedure by normalizing all the pixels with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$. The adversarial settings that have been used throughout of our experiments is provided in Table 5.5. Finally, we used $T = 50$ as the number of MC dropout samples when quantifying epistemic uncertainty.

Table 5.5 Adversarial settings of our experiments: α, i respectively represent the step-size and the number of attack steps for a perturbation budget ε, r denotes the number of restarts

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

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

The results in table 5.6 show that our Rectified-BIM algorithm and other two attack variants (Rectified-FGSM and Rectified-PGD) which leverage both the quantified epistemic uncertainty and the model's loss value result into better performances than their originals. Once we verified the success of our attack variants on our comparably small models with different epsilon (ε) values, we tried to test their performances on larger models. For this purpose, we first trained VGG-19 (Simonyan & Zisserman, 2015) (with batch normalization) and ResNet (with custom dropout layers) models on CIFAR-10 dataset and achieved accuracy rates of % 85.79 and % 86.24. Then, we have applied all our attack variants and compared the attack success rates with their originals. The results available in Table 5.7 reveal once again the effectiveness of our proposed attack method. After verifying the efficacy of our approach on various model architectures, we conducted a separate experiment to observe the effect of chosen *number of iterations* parameter on the performance of our proposed iterative attack variant. The results available in Table 5.8 show that our PGD attack variant outperforms its original counterpart in each attempt (PGD5, PGD10, PGD20).

Unlike known loss-based attacks, our proposed attack variants (rectified attacks) perturb less number of pixels at each iteration and still can reach a higher degree of adversarial attack success rate. To verify this, we have conducted an additional experiment to compare the resulting perturbation amounts of our attack variants with their

Table 5.6 Attack success rates on different datasets

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

Table 5.7 Attack success rates under l_∞ norm against VGG-19 and ResNet models

CIFAR10 Dataset	**VGG19bn**	**ResNet**
FGSM ($\varepsilon = 3/255$)	56,78%	74,33%
Rectified-FGSM ($\varepsilon = 3/255$)	59,18%	78,11%
BIM ($\varepsilon = 3/255$, $\alpha = \varepsilon \cdot 0.4$, i = 5)	73,45%	94,99%
Rectified-BIM ($\varepsilon = 3/255$, $\alpha = \varepsilon \cdot 0.4$, i = 5)	84,67%	96,42%
PGD ($\varepsilon = 3/255$, $\alpha = \varepsilon \cdot 0.4$, i, r = 5)	74,95%	95,06%
Rectified-PGD ($\varepsilon = 3/255$, $\alpha = \varepsilon \cdot 0.4$, i, r = 5)	83,97%	96,17%

Table 5.8 Results under PGD attack with various number of iterations

VGG19bn on CIFAR10	**PGD**	**Rectified-PGD**
Number of iterations = 5, $\varepsilon = 3/255$	73,25%	83,01%
Number of iterations = 10, $\varepsilon = 3/255$	82,13%	91,30%
Number of iterations = 20, $\varepsilon = 3/255$	83,60%	92,24%

originals (FGSM, BIM, PGD). We have used a batch of input from each dataset and computed the resulting perturbation amounts based on both L_1 and L_2 norms. The results that are shown in Table 5.9 support our claim. We know that any trained ML model is not the perfect predictor and is just an approximation to the oracle function. The model itself has an inevitable inherent approximation uncertainty which sometimes induce to sub-optimal solutions. Consequently, any method which only relies on the trained model will result in less effective performance. By double-checking the sub-directions pointed by the derivative of the model's loss with the ones available in the quantified epistemic uncertainty derivative, we could discard the sub-directions, which are suspected to be unreliable. The attack success rates are inline with this fact. There have been previous studies in literature which try to minimize the amount of perturbation applied to the input of the AI model as in (Akan, Genc, & Vural, 2020) for crafting adversarial samples. However, the adversarial success rate of these attack algorithms are questionable and mostly not comparable to their traditional counterparts as primary goal of these method is to generate adversarial samples with least perceptible difference instead of most powerful ones.

Table 5.9 Mean Perturbation Amount Comparison

	MNIST Digit		MNIST Fashion		CIFAR10	
	($\varepsilon = 0.16$)		($\varepsilon = 0.06$)		($\varepsilon = 3/255$)	
	L_2 norm	L_1 norm	L_2 norm	L_1 norm	L_2 norm	L_1 norm
FGSM	3,36	71,57	1,44	35,19	0,64	35,90
Rect.-FGSM	3,06	59,53	1,29	28,56	0,56	27,48
BIM	2,75	53,41	1,21	27,40	0,55	28,27
Rect.-BIM	2,69	52,09	1,16	25,99	0,54	27,70
PGD	2,79	54,79	1,22	27,96	0,55	28,39
Rect.-PGD	2,74	53,70	1,19	26,92	0,54	28,02

As a last experiment for the attack part, we have measured the time spent by each attack method for a batch of input of size 64 from the MNIST (Digit) Dataset. The results available in Table 5.10 show the measured execution time of our attack variants together with their originals (FGSM, BIM, PGD). As expected, the execution time of our attack variants is longer than their originals due to additional uncertainty quantification steps. However, this can be tolerated thanks to the higher success rate and smaller perturbation need of our attack variants.

Table 5.10 Execution Time Comparison (in seconds)

Attack Type	**Time**
FGSM	0,0104
Rectified-FGSM	0,4613
BIM	0,0444
Rectified-BIM	6,4161
PGD	0,4695
Rectified-PGD	31,3377

CHAPTER 6

ADVERSARIAL DEFENSE

6.1 Approach

In regions with a low number of training samples, model uncertainty is larger. We can't get a perfect model to precisely predict all test data because we don't have any ground truth in these areas. Figure 6.1 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 don't 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 quality. Therefore, testing the model in severe settings with input that it has never encountered before will lead to model prediction failure as the wights of the model are not properly optimized to correctly predict these regions (O. F. Tuna et al., 2022b). 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 against predicted label.

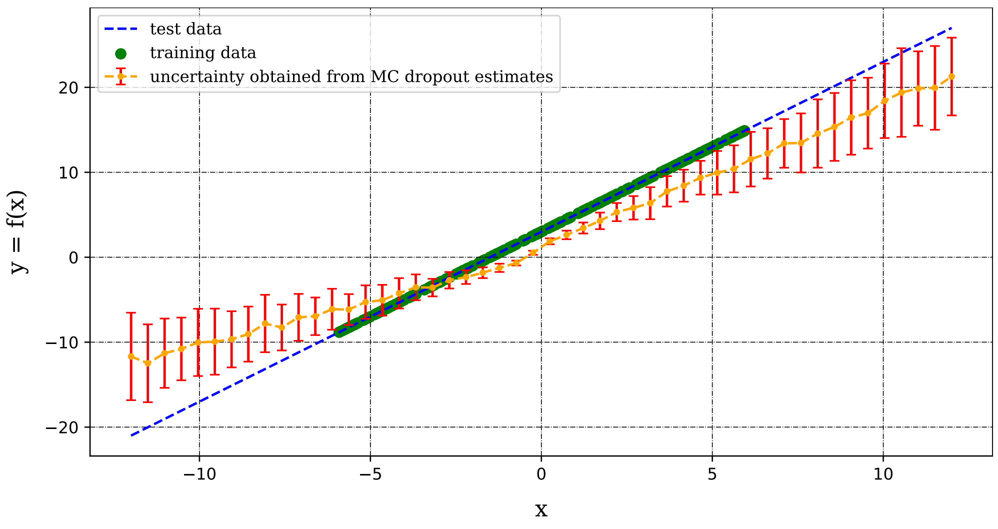


Figure 6.1 Uncertainty values obtained from a regression model

6.2 Intuition Behind Using Uncertainty-based Reversal Process

For the quantification of uncertainty metrics, we use MC-Dropout Estimates of DNNs. For instance, quantification of epistemic uncertainty involves using the model in training mode, getting T different feed-forward predictions and calculating the variance of these predictions. What we actually do during this operation is to use T different models, because dropout mechanism will randomly zeros out the activations of different neurons of the model as displayed in right side of Figure 6.2. However, during normal operation of the model in inference time, we use all the neurons and do not force the activations of any neuron to zero as in left side of the Figure 6.2. Therefore, the model that is used in inference time and all the models that are used during uncertainty quantification are different from each other.

The adversarial samples target the model to force them to incorrect predictions during inference time. For example, white-box adversarial attacks use the models that are used in their inference time to compute the gradients to increase the loss against any desired class. But, these crafted perturbations are designed specifically for the model that is used in inference time. The adversarial perturbation will infect part of the neurons in the network which will result them to perform abruptly (Wu & Wang, 2021), (Guan, Tu, He, & Tao, 2021). Because the weights of these neurons are not optimized well during the learning process, making them prone to unseen form of data. However, when we use the model in training mode, the crafted perturbations will

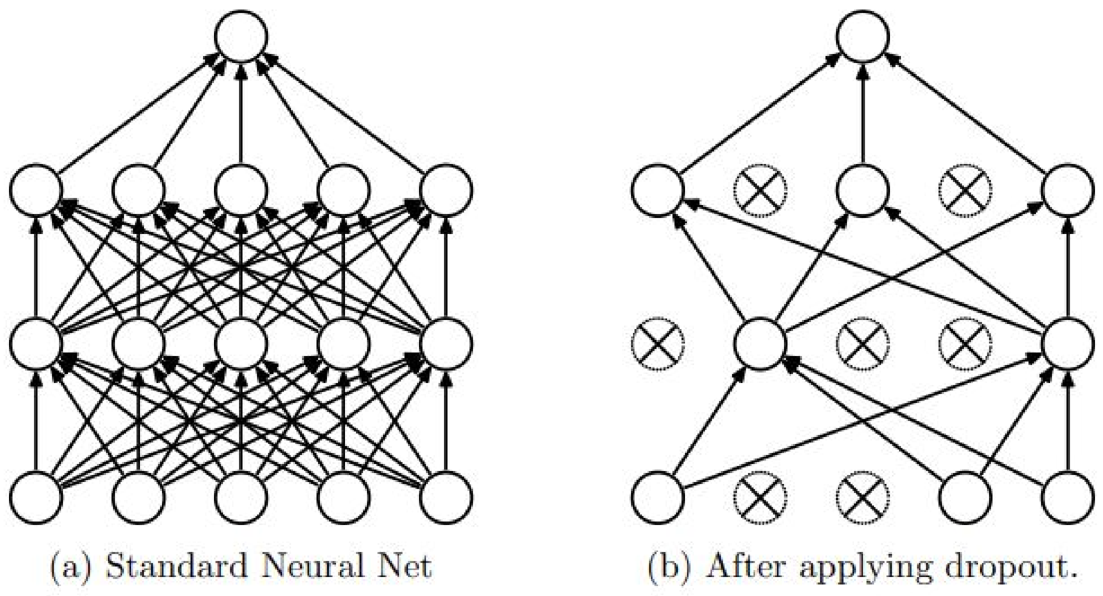


Figure 6.2 Dropout mechanism - Srivastava Nitish et al. 2014

not be as effective as it is intended. Because some of the infected neurons will not be active in that particular model. Dropout mechanism will unintentionally prune some of these infected neurons. To verify this, we conducted a simple experiment and assess the success rate of Deepfool attack against a model in evaluation mode and against the exactly equal model in training mode where dropout is enabled. We observed that the success rate in evaluation mode is 25.01 % whereas the success rate in training mode is 15.96 %. Because, when we randomly disable part of the neurons based on dropout probability, the infected neurons that are available in the actual model that is used in inference time will not be active and therefore predictions of the model will not be depending on them as illustrated in Figure 6.3.

Misclassification of adversarial sample is the result of the infected neurons in the model whose weights are sensitive to adversarial perturbations. The aim of our uncertainty based reversal method is to decrease the sensitivity of these infected neurons to the input sample. When we perturb the image in a direction to minimize the model's quantified uncertainty, we are updating the pixels of the input image so that the erroneous activations of these infected neurons are minimized. Because the uncertainty or the variance of T feed-forward predictions results from the existence of infected neurons and their erroneous activations.

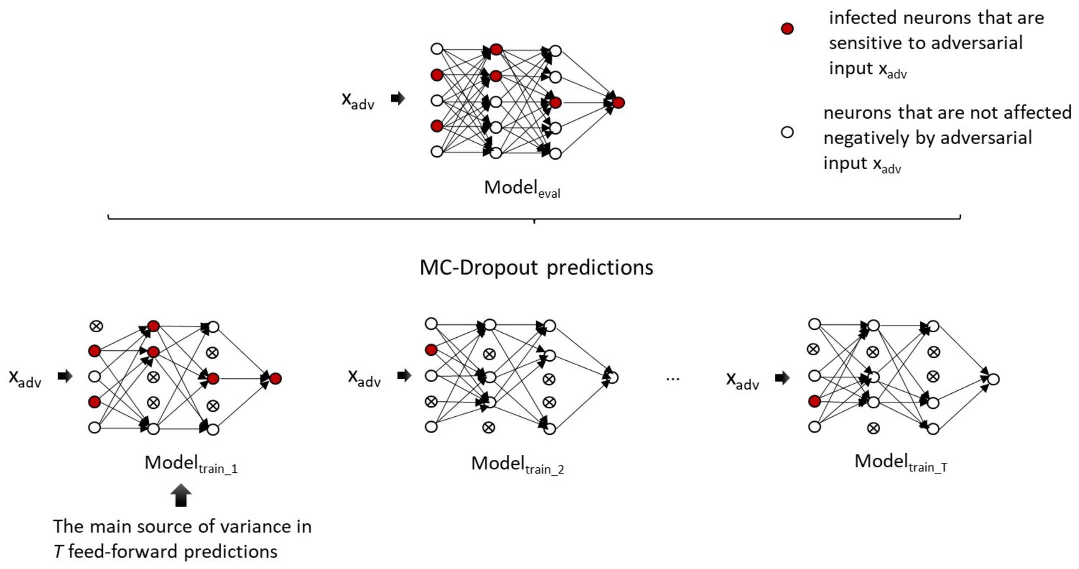

infected neurons that are sensitive to adversarial input x_adv
neurons that are not affected negatively by adversarial input x_adv
x_adv
Model_eval
MC-Dropout predictions
Model_train_1
Model_train_2
Model_train_T
The main source of variance in T feed-forward predictions

Figure 6.3 Sensitivity of different models to adversarial perturbations

6.3 Uncertainty-Based Reversal Operation

We start this section by presenting the pseudo-code for our uncertainty-based reversal procedure, as described in Algorithm 6.1. This reversal method is designed under L_∞ norm.

Algorithm 6.1: Algorithm for uncertainty-based reversal:
$\mathbf{x}$ is the input image, h is the learnt hypothesis function, p denotes the dropout ratio of the model used in dropout layers, T represents the number of MC dropout samples at prediction time in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
   Input: $\mathbf{x} \in \mathbb{R}^m, h, p, T, N, \varepsilon, \alpha$
   Output: $\mathbf{x}_{t+1}$
1  $\mathbf{x}_0 \leftarrow \mathbf{x}$
2  $condition \leftarrow False$
3  while $n < N$ do
4      Compute $\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T)$ while $h$ in training mode
5      if $argmax(h(\mathbf{x}_{t+1})) \neq y_{pred}$ then
6          $condition = True$
7          break
8      if $condition = False$ then
9          $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}} U(\mathbf{x}_t, h, p, T))$
10 return $\mathbf{x}_{t+1}$
```

Via this algorithm, we propose a method in which before feeding any input to the DNN model, we try to revert it back to its original data manifold by minimizing its quantified epistemic uncertainty. Since uncertainty quantification metrics does not depend on any kind of label information, by minimizing the uncertainty value, one can have a chance to revert the input instance back to its original data manifold depending on where the adversarial sample resides.

6.4 Enhanced Uncertainty-Based Reversal Operation

In a typical production environment where an ML model is used for any classification problem, the input is fed to the model, and the final decision is observed after the input is processed and mapped to an output, as shown in the upper example of Figure 6.4.

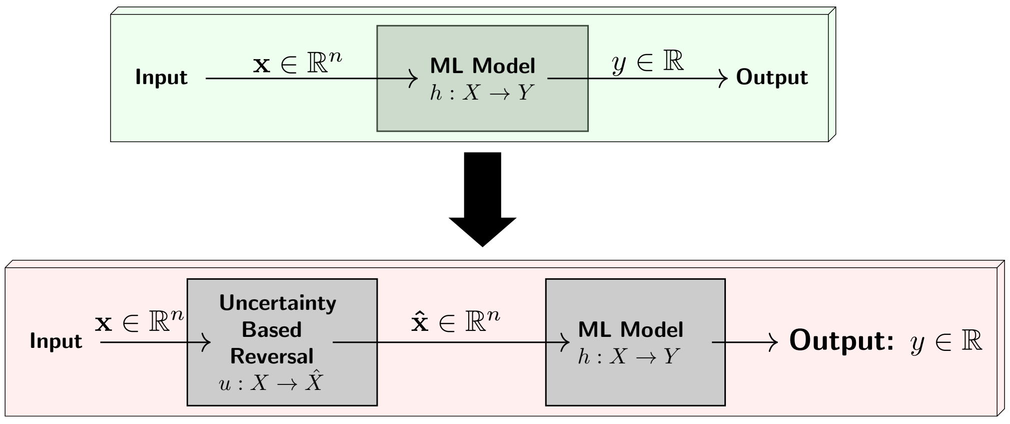

Input
$\mathbf{x} \in \mathbb{R}^n$
ML Model
$h : X \to Y$
$y \in \mathbb{R}$
Output
Input
$\mathbf{x} \in \mathbb{R}^n$
Uncertainty Based Reversal
$u : X \to \hat{X}$
$\hat{\mathbf{x}} \in \mathbb{R}^n$
ML Model
$h : X \to Y$
Output: $y \in \mathbb{R}$

Figure 6.4 Deployment options of the ML model - with and without uncertainty based reversal step

During this prediction time, the model does not have any information about the actual label for the input, yet it certainly has an opinion about the predicted label. And this predicted label can be used to quantify the loss. Suppose the prediction of the ML model is correct. In that case, the derivative of the loss against the predicted label gives us an idea about the possible direction by which we can decrease the predicted loss. However, if the initial prediction is wrong, in that case, the model's prediction will be even more erroneous. In this enhanced version of our uncertainty-based reversal method as described in Algorithm 6.2, we use both the quantified uncertainty and loss value against predicted class. At each iteration of the enhanced uncertainty-based reversal algorithm, we calculate $\nabla_{\mathbf{x}}\ell(h(\mathbf{x}_t, y_{pred}))$ and $\nabla_{\mathbf{x}}U(\mathbf{x}_t, h, p, T)$. Then, we simply perturb the input image by using the sub-directions of uncertainty's gradient which are not shared by the loss' gradient.

Algorithm 6.2: Algorithm for enhanced uncertainty-based reversal:
$\mathbf{x}$ is the input image, y_{pred} is the predicted label for $\mathbf{x}$, h is the learnt hypothesis function, p denotes the dropout ratio of the model used in dropout layers, T represents the number of MC dropout samples at prediction time in model training mode, N denotes iteration number, ε is the maximum amount of perturbation allowed, α denotes step size.

```
   Input: $\mathbf{x} \in \mathbb{R}^m, h, p, T, N, \varepsilon, \alpha$
   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 $argmax(h(\mathbf{x}_{t+1})) \neq y_{pred}$ 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         $\mathbf{x}_{(t+1)} = clip_{\mathbf{x},\varepsilon}(\mathbf{x}_t - \alpha \cdot sign(\nabla_{\mathbf{x}}U(\mathbf{x}_t, h, p, T))$
12 return $\mathbf{x}_{t+1}$
```

This idea can be understood better by looking at Figure 6.5 - (C). Different from Algorithm 6.1, this time we will only take the sub-directions from the uncertainty's gradient which are not shared by the loss' gradient. Since, for the perturbed images that are already pushed away from the decision boundary, the gradient of the loss against predicted label will point to the target class data manifold. Therefore, these sub-directions should not be taken into account if we wish to revert the input sample back to its own data manifold. After discarding some portion of the sub-directions, the resulting sub-directions left in the uncertainty's gradient can be safely used to revert the input back to its original data manifold.

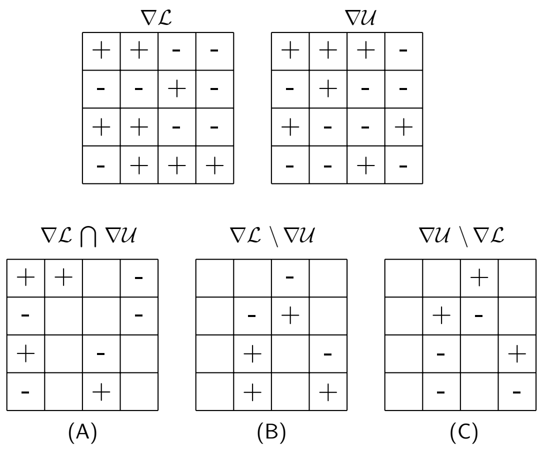


Figure 6.5 Sub-directions (sign of the gradients) used in defense purposes (C)

6.5 The Usage of Uncertainty-based Reversal

Our proposed uncertainty-based reversal algorithm can be implemented as a *pre-processing* module prior to any classification model in production. As seen in the bottom section of Figure 6.4, 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.

The uncertainty based reversal operation is visualized in Figure 6.6. The key point for a successful reverse operation is that the point where the input image resides should not be far away (on the wrong side) from the model's decision boundary. The attack types that we used to show our approach's effectiveness are chosen based on this fact.

Our defense method is evaluated in terms of the error rate across a maliciously perturbed version of the chosen test set. This error rate metric is proposed by Goodfellow et al. (Goodfellow et al., 2015) and is still suggested by Carlini et al. (Carlini et al., 2019).

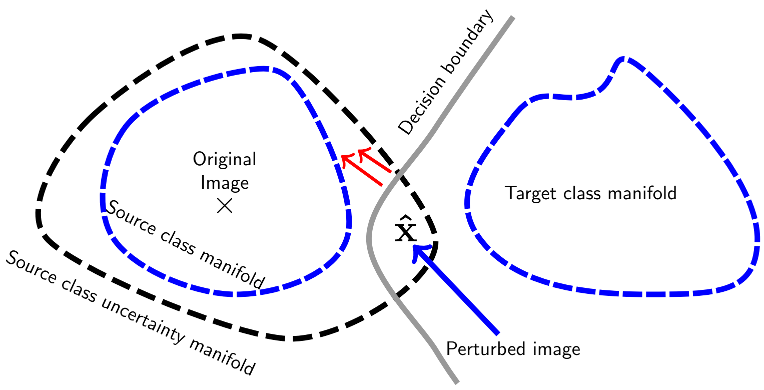


Figure 6.6 Reverting perturbed image back to its original class data manifold

6.6 The Effect of Uncertainty-based Reversal

It was shown that no matter what kind of defensive approaches are utilized like adversarial training or defensive distillation technique, strong attack types such as Carlini&Wagner (CW) or Deepfool attacks, can break these defensive methods and successfully fool the ML model (Carlini & Wagner, 2017b). However, when we analyze the softmax scores of the perturbed samples which are crafted by using Deepfool attack and CW attack with default setting (confidence parameter set to 0), we see that the resulting confidence level for the wrong class is only slightly larger than the original class. That means the perturbed image is actually not pushed far away from the decision boundary. Consequently, by applying our uncertainty based reversal operation, one can actually revert the perturbed sample back to the original class data manifold. We have picked random samples from the MNIST Digit dataset to illustrate this phenomenon, then applied CW and Deepfool attacks on them to craft adversarial samples and finally applied our uncertainty-based reversal operation on these perturbed samples. For each of the original, perturbed and reversed samples, we have also depicted the output softmax scores of the ML model used as in Figure 6.7. We observe that although these attacks are successful, the wrong classes' output softmax scores are not large enough in favor of the wrong class, and the difference between the softmax scores of the correct and wrong classes is considerably small. As a result, by applying our uncertainty based reversal operation on these perturbed samples, we could successfully revert them back to their original class manifold so that the ML model will predict these reversed samples correctly, and therefore will not be fooled. And as a natural consequence of a successful reversal operation, the adversarial inputs can be detected as well. Line 6 of Algorithm 6.2 checks whether the predicted label for the input has changed from the initial prediction. If this is the case, one can use this as a possible sign of adversarial detection. Therefore, our approach can be used for both detecting adversarial samples and defending against them.

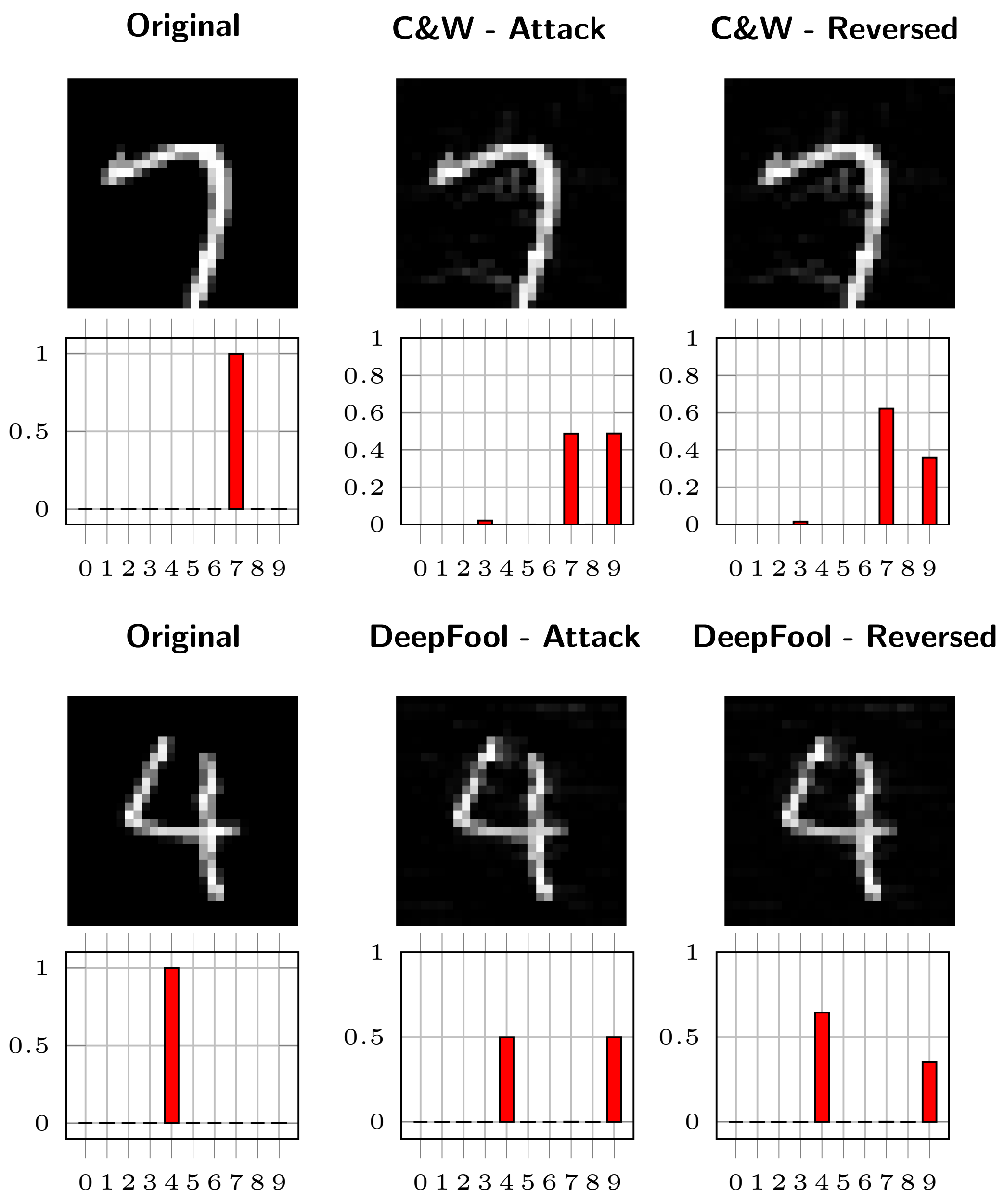


Figure 6.7 Effect of Adversarial Attack and Reversal Operation on Model Prediction

6.7 Variants of the Enhanced Uncertainty-Based Reversal Operation

In this study, we have first used uncertainty-based reversal method which is based on epistemic uncertainty and then developed 2 additional variants of the same algorithm 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 3.7. We used the expected value(mean) of the epistemic uncertainty (*EP*) for the uncertainty quantification as in the case of Eq. 6.1. Then, we tried scibilic uncertainty (SC) and used mean of the *SC* via Eq.6.2. Lastly, instead of using the average scibilic uncertainty measure of all classes, we used the uncertainty value of the predicted class only as in Eq. 6.3. In this way, we used the following three equations for uncertainty quantification.

$$U(h, \mathbf{x}_t, p, T) = \frac{1}{K}\sum_{k=1}^{K} EP[k] \tag{6.1}$$

$$U(h, \mathbf{x}_t, p, T) = \frac{1}{K}\sum_{k=1}^{K} SC[k] \tag{6.2}$$

$$U(h, \mathbf{x}_t, p, T) = SC[pred] \tag{6.3}$$

6.8 Hybrid Deployment Options

Our uncertainty-based reversal method works well for the cases where the resulting adversarial sample resides in the vicinity of the decision boundary of the model. Black-box attack type like Hopskipjump Attack or some powerful white-box attack types like Carlini-Wagner (CW) or Deepfool Attack can be considered in this category. However, some gradient-based attack-types like FGSM, BIM, PGD might result in adversarial samples that are pushed far away from the decision boundary. which makes our approach less effective. To defend the model from vast variety of attack threats, we propose to apply a hybrid approach and apply our method prior to adversarially trained or defensively distilled models. These additional defense mechanisms will complement with our reversal method and provide excellent robustness to wide range of threats.

6.8.1 Via Adversarial Training

It has been shown that adversarially trained models provide important robustness to most of the existing attacks but there exist in literature some known attack algorithms which are still effective against these models. One can regard Deepfool , CW or Hoskipjump attacks among these algorithms. Luckily the adversarial samples resulted by these algorithms are not pushed far away from the decision boundary of the models as can be understood from the top-2 class probability differences. And, via combining our uncertainty-based reversal approach with adversarial training, we have a chance to defend most of the existing attack methods.

6.8.2 Via Defensive Distillation

Although our 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 (O. Tuna, Catak, & Eskil, 2022). 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 boundary. Thanks to the second effect, the crafted adversarial samples by using CW attack with confidence parameter set to a high value are found near the decision boundary. Therefore, when we combined reversal procedure with defensive distillation, we can achieve much better results. The results available in experimental proves this fact.

Figure 6.8 shows one of our proposed hybrid deployment options where our uncertainty based reversal procedure is applied as a preprocessing module prior to a defensively distilled model.

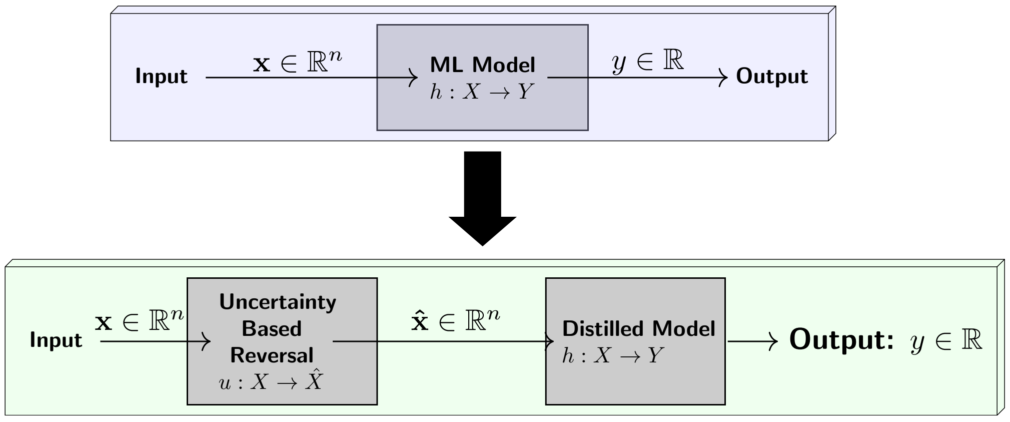
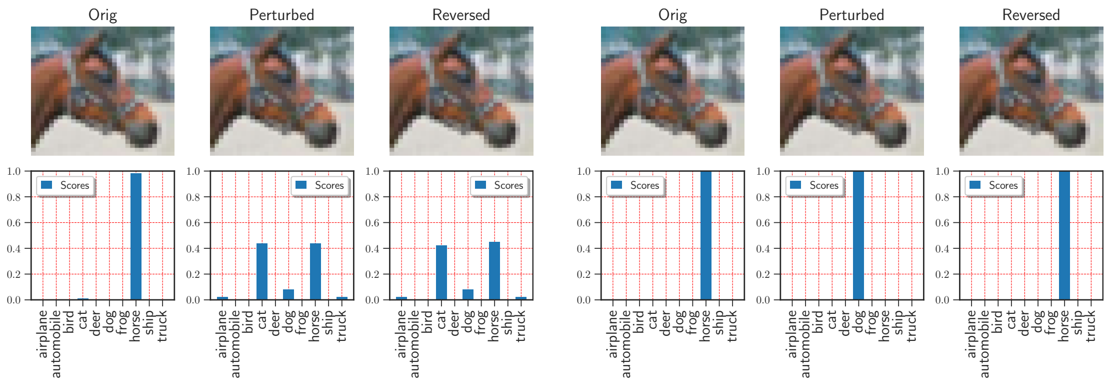

Input
$\mathbf{x} \in \mathbb{R}^n$
ML Model
$h : X \to Y$
$y \in \mathbb{R}$
Output
Input
$\mathbf{x} \in \mathbb{R}^n$
Uncertainty Based Reversal
$u : X \to \hat{X}$
$\hat{\mathbf{x}} \in \mathbb{R}^n$
Distilled Model
$h : X \to Y$
Output: $y \in \mathbb{R}$

Figure 6.8 Hybrid option for ML model deployment

Last the effect of our uncertainty based reversal method applied on a defensively distilled model can be seen in Figure 6.9. The Deepfool attack algorithm used in this example 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.



(a) normal model **(b)** Student model

Figure 6.9 The effect of uncertainty-based reversal procedure on normal and distilled models

6.9 The Effect on Clean Data Performance

For our proposed defense method, we employed a proactive strategy. We anticipate the potential threat of adversarial activity throughout the deployment life cycle of the ML model regardless of whether an attack attempt exists or not. The same preprocessing operation is being applied to every input that is planned to be fed to the ML model regardless of being an adversarial or benign sample. Obviously, a key point that should be considered for any kind of preprocessing operation on the input samples of an ML model is that the preprocessing operation should not negatively affect the ”clean data” performance of the model. Any kind of intervention to the ML model’s deployment which will decrease the prediction performance higher than an acceptable level might not be tolerated no matter how much robustness it provides. We have performed a test to measure the effect of our uncertainty based reversal operation on the clean data performance of the model and verified that the accuracy rate does not decrease as presented in the experiments section. The results prove that one can safely use our approach to increase the robustness against malicious attempts to the deployed ML model.

6.10 Results

6.10.1 Part-1

6.10.1.1 Experimental Setup

For the first part of our experiments, we trained our CNN models for the MNIST (Digit) (LeCun & Cortes, 2010), MNIST (Fashion) (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., n.d.) datasets, and we achieved accuracy rates of 99.26 % , 92.63 % and 83.91 % respectively. The model architectures are given in Table 6.1 and the hyperparameters selected in Table 6.2. For the CIFAR10 dataset we applied a normalization procedure by normalizing all the pixels with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$. The adversarial settings that have been used throughout of our experiments is provided in Table 6.3. Finally, we used $T = 50$ as the number of MC dropout samples when quantifying epistemic uncertainty.

Throughout our experiments, we applied attack algorithms on the test samples only if they were previously classified correctly by our models. Because, an attacker would obviously have no motivation to perturbed samples that have already been misclassified. We utilized an open source Python library called Foolbox (Rauber et al., 2018) to implement the attacks used in this study.

Table 6.1 CNN model architectures

Dataset	Layer Type	Layer Information
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Dropout	p : 0.2
	Dense + ReLU	1568×100
	Dense + ReLU	100×10
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 : 0.2
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 64$
	Dropout	p : 0.2
	Dense + ReLU	3136×600
	Dense + ReLU	600×120
	Dense + 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
	Dropout	p : 0.1
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Convolution (Padding = 1) + ReLU	$3 \times 3 \times 256$
	Max Pooling (Stride 2)	2×2
	Dense + ReLU	4096×512
	Dropout	p : 0.5
	Dense + ReLU	512×512
	Dropout	p : 0.5
	Dense + ReLU	512×10

Table 6.2 CNN model parameters

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

Table 6.3 Parameters that are used in uncertainty reversal operation: α, i respectively represent the step-size and the number of reversal steps for a perturbation budget ε

Dataset	Parameters	l_p norm
MNIST Digit	$\varepsilon = 0.02$, $\alpha = \varepsilon \cdot 0.2$, i = 10	l_∞
MNIST Fashion	$\varepsilon = 0.001$, $\alpha = \varepsilon \cdot 0.2$, i = 10	l_∞
CIFAR10	$\varepsilon = 0.1/255$, $\alpha = \varepsilon \cdot 0.2$, i = 10	l_∞

6.10.1.2 Experimental Results

We begin this section by presenting the results of Algorithm 6.1 against some of the well-known attack algorithms. The results available in Table 6.4 show that our uncertainty-based reversal method provide important degree of robustness to the deployed model. However, we still see some portion of adversarial samples which our method could not revert back to their original manifolds.

Table 6.4 Performance of uncertainty-based reversal operation (Algorithm-1) on different attack types

		success. rev.	unsuccess. rev.	rev. succ. rate
MNIST (Dig.)	Deepfool (L_∞, ε : 0.1)	1837	668	73,33%
	CW (L_2, ε : 1.35)	4553	608	88,21%
	Hopskipjump (L_∞, ε : 0.1)	1942	403	82,81%

After checking the defensive performance, we wanted to check whether the implementation of our method has some negative effect on the natural performance of the ML model against clean dataset. The result available in Table 6.5 show that the accuracy of the model does not drop and it is safe to use our method as a preprocessing module prior to feeding any input to the model.

Then, we would like to check whether our enhanced uncertainty based reversal method (Algorithm 6.2) has a better performance compared to Algorithm 6.1. We start this by checking the clean data performance first. The results in Table 6.6 show that our enhanced proposed uncertainty based reversal method does not decrease the classification performance of the deployed models on clean data. While some minor portion of data samples were wrongly classified even if they were previously classified correctly, there is an almost similar amount of samples that happened to be predicted correctly after the reversal process. Therefore, we can conclude that on average, there is no negative effect of deploying our uncertainty based reversal method as a preprocessing module prior to feeding any input to an ML model in production.

Table 6.5 Performance of uncertainty based reversal (Algorithm-1) operation on clean data

		number of successful reversal	number of unsuccessful reversal
MNIST (Dig.)	on corrects	9924	2
	on wrongs	5	69

Table 6.6 Performance of enhanced uncertainty based reversal (Algorithm-2) operation on clean data

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

We then tested the enhanced version of our proposed defense technique (Algorithm 6.2) against two powerful attack types, namely Deepfool attack and Carlini&Wagner attack (with confidence = 0). The results presented in table 6.7 prove the effectiveness of our enhanced uncertainty-based reversal method. For the MNIST datasets, our proposed defense method provides perfect robustness and almost totally revert all the perturbed samples crafted by these strong attacks. And for the CIFAR10 dataset, we achieved reversal success rates of around % 96 for Deepfool attack and almost % 98 for CW attack. Thanks to our defense method, we could lower the final attack success rates of these attacks to very low levels as shown in Table 6.8.

Table 6.7 Performance of enhanced uncertainty based reversal (Algorithm-2) operation on different attack types

		success. rev.	unsuccess. rev.	rev. success. rate
MNIST (Dig.)	Deepfool (L_∞, ε : 0.1)	2505	0	100%
	Deepfool (L_2, ε : 1.35)	2935	27	99,09%
	Hopskipjump (L_∞, ε : 0.1)	2345	5	99,78%
	CW (L_2, ε : 1.35)	5161	0	100%
MNIST (Fash.)	Deepfool (L_∞, ε : 0.05)	7940	19	99,76%
	Deepfool (L_2, ε : 0.67)	7268	25	99,66%
	CW (L_2, ε : 0.67)	7898	19	99,76%
CIFAR10	Deepfool (L_∞, ε : 2/255)	5078	255	95,22%
	Deepfool (L_2, ε : 53/255)	4081	117	97,21%
	CW (L_2, ε : 53/255)	4904	102	97,96%

Table 6.8 Effect of enhanced uncertainty-based reversal (Algorithm-2) on attack success rates

		Attack Success Rates	
		without reversal	with reversal
MNIST (Dig.)	Deepfool (L_∞, ε : 0.1)	25,24%	0,00%
	Deepfool (L_2, ε : 1.35)	29,84%	0,27%
	CW (L_2, ε : 1.35)	51,99%	0,00%
MNIST (Fash.)	Deepfool (L_∞, ε : 0.05)	85,92%	0,21%
	Deepfool (L_2, ε : 0.67)	78,73%	0,27%
	CW (L_2, ε : 0.67)	85,47%	0,21%
CIFAR10	Deepfool (L_∞, ε : 2/255)	63,56%	3,04%
	Deepfool (L_2, ε : 53/255)	50,03%	1,39%
	CW (L_2, ε : 53/255)	59,66%	1,22%

The results available above show that we significantly increased the defensive performance of our uncertainty-based reversal method by simply discarding the sub-directions available in the gradient of the model's loss against predicted class in Step 10 of Algorithm 6.2. Therefore, we picked our enhanced uncertainty based reversal method as our base solution and used Algorithm 6.2 in the rest of the experiments.

6.10.1.3 Discussions and Results with Hybrid Approach (Adversarial Training)

We have tested and verified our approach's effectiveness in three different datasets, which are heavily used by the adversarial research community. Experimental results show that our proposed methods generalize well across datasets. Our defense method is not concerned with the internal dynamics of the attack. Instead, it is concerned with the eventual result of the attack, which is the perturbed sample. For normally trained models, the underlying factor for the success of our approach can be understood better by analyzing the resulting final probability scores of the perturbed samples. Since the gap between the predicted probability scores of original and target classes is not high for Deepfool attack and CW attack (when confidence parameter is set to 0), we could successfully revert almost all perturbed images with a reasonably small perturbation because the perturbed images are not pushed far away from the decision boundary of the model for these attacks. Other known attacks like FGSM, BIM or PGD might not result in such a small gap in the final softmax scores. This is also valid if one applies CW attack with a high confidence value as a parameter as well. Thus, our reversal process's success rate will not be so high as in the case of the previously mentioned attack implementations. However, we have empirically verified tools like adversarial training to defend against those attacks (FGSM, BIM, PGD), and the success rate of those attacks can be lowered to an acceptable level.

To verify this, we have used two other models, which are naturally and adversarially trained for the MNIST Digit classification task. We used a slightly different architecture this time and applied the dropout in the first convolutional layer as in Table 6.9. We then checked the attack success rates of FGSM, BIM, Deepfool and CW attacks against these two trained models. As expected, the success rates for FGSM and BIM were significantly lowered via adversarial training. Additional implementation of our uncertainty based reversal approach decreased the attack success rates even further, as shown in Table 6.10.

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

Finally, when we check if our uncertainty based reversal operation negatively affect the model's performance for legitimate inputs as shown in Table 6.11, we once again confirm that the accuracy of the model does not decrease considerably at all. Hence, our approach can be safely used in production when there is no security threat. Suppose that the model owner is worried about any suspicious activity and thinks that the model is under attack. In that case, he/she can tune the ε parameter for the reversal operation to increase the power of defense at the cost of slightly lowering clean data performance.

Table 6.9 CNN model architecture for Adversarially Trained MNIST Model

Dataset	**Layer Type**	**Layer Information**
MNIST (Digit)	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 16$
	Dropout	p : 0.2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Max Pooling	2×2
	Convolution (padding:1) + ReLU	$3 \times 3 \times 32$
	Max Pooling	2×2
	Dense + ReLU	1568×100
	Dense + ReLU	100×10

Table 6.10 Effect of Adversarial Training and Uncertainty Based Reversal on attack success rates

		Attack Success Rates	
		without reversal	with reversal
Normal Model	FGSM ($\varepsilon = 0.1$)	% 11.82	% 9.01
	BIM (L_∞, $\varepsilon = 0.1$)	%32.02	% 22.13
	Deepfool (L_∞, $\varepsilon = 0.1$)	% 17.28	% 0.02
	Deepfool (L_2, $\varepsilon = 1.35$)	% 22.79	% 0.30
	CW (L_2, $\varepsilon = 1.35$, confidence = 0)	% 50.42	% 0
	CW (L_2, $\varepsilon = 1.35$, confidence = 10)	% 40.42	% 30.38
Robust Model	FGSM (eps = 0.1)	% 1.32	% 1.02
	BIM (L_∞, $\varepsilon = 0.1$)	% 1.43	% 1.11
	Deepfool (L_∞, $\varepsilon = 0.1$)	% 1.34	% 0.01
	Deepfool (L_2, $\varepsilon = 1.35$)	% 4.14	% 0.20
	CW (L_2, $\varepsilon = 1.35$, confidence = 0)	% 7.78	% 0.03
	CW (L_2, $\varepsilon = 1.35$, confidence = 10)	% 4.82	% 4.04

Table 6.11 Effect of Uncertainty Based Reversal on clean performance for normal and adversarially trained model

		# of success. reversal	# of unsuccess. reversal
Normal Model	on corrects	9916	17
	on wrongs	19	48
Robust Model	on corrects	9921	13
	on wrongs	4	62

6.10.2 Part-2

For the second part of 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) (LeCun & Cortes, 2010), MNIST (Fashion) (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., n.d.) 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%.

We started the second part of our experiments by first evaluating the contribution of different uncertainty metrics on uncertainty-based reversal procedure performance. To do this, we used a normal CNN model which is trained on MNIST (Digit) Dataset, 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 explained in Section 6.7. Table 6.12 summarizes the values of the parameters that are used in the reversal procedure.

Table 6.12 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**	**l_p norm**
MNIST Digit	$\epsilon = 0.02$, $\alpha = \epsilon \cdot 0.2$, i = 10	l_∞
MNIST Fashion	$\epsilon = 0.006$, $\alpha = \epsilon \cdot 0.2$, i = 20	l_∞
CIFAR10	$\epsilon = 0.2/255$, $\alpha = \epsilon \cdot 0.2$, i = 10	l_∞

The results of this experiment is provided in Table 6.13. 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. We also observe a considerable difference 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. 6.1) to Scibilic Uncertainty (Eq. 6.2). And instead of using the mean of Scibilic Uncertainty vector, if we use the uncertainty value of the predicted class only (Eq. 6.3), we can even lower the final attack success rate to 18.16 %.

Table 6.13 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_∞, $\epsilon : 0.1$)	13.25%	9.04%	13.25%	6.41%	13.25%	5.82%
BIM (l_∞, $\epsilon : 0.1$)	38.94%	31.47%	38.94%	20.48%	38.94%	18.16%
PGD (l_∞, $\epsilon : 0.1$)	35.32%	27.34%	35.32%	18.64%	35.32%	16.68%
Deepfool (l_∞, $\epsilon : 0.1$)	21.60%	0.07%	21.60%	0.01%	21.60%	0.01%
Deepfool (l_2, $\epsilon : 0.1$)	26.78%	1.07%	26.78%	0.20%	26.78%	0.11%
CW (l_2, $\epsilon : 1.35$ $c : 0$)	67.71%	0,00%	67.71%	0.02%	67.71%	0.01%
CW (l_2, $\epsilon : 1.35$ $c : 40$)	21.15%	16.87%	21.15%	7.44%	21.15%	7.16%

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 6.14, 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 the rest of our experiments.

As previously explained in Part-1, 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

Table 6.14 Effect of reversal procedure on clean performance of normally trained model - MNIST (Digit) Dataset

	Model Performance	
	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 %

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 adversarial defense 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 boundary. Therefore, when we combined reversal procedure with defensive distillation, we achieved much better results. The results in Table 6.15 show that our proposed hybrid architecture 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 6.16).

Table 6.15 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 6.16 Effect of reversal procedure on clean performance of distilled model - MNIST (Digit) dataset

	Model Performance	
	without reversal	with reversal
Scibilic Unc. Based Reversal (pred class)	99,41 %	99,10 %

To assess and verify the efficacy of our proposed architecture, we have conducted additional experiments on different datasets. Table 6.17 shows the performance of reversal procedure on MNIST (Fashion) dataset for both *normal* and *distilled* models.

Table 6.17 Attack success rates on MNIST (Fashion) dataset with and without uncertainty based reversal procedure

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

And Table 6.18 shows the effect of our proposed architecture on clean data classification performance.

Table 6.18 Effect of reversal procedure on clean performance - MNIST (Fashion) dataset

	Model Performance	
	without reversal	with reversal
Normal Model	92,61 %	90,18 %
Distilled Model	92,62 %	90,25 %

Finally, we have performed the same set of experiments on CIFAR-10 dataset. The results are available in Table 6.19 and Table 6.20. The results of our detailed experiments on all the datasets reveal that our reversal procedure and defensive distillation technique can handle complimentary situations and together they provide very high degree of robustness against various kinds of untageted attacks.

Table 6.19 Attack success rates on CIFAR-10 dataset with and without uncertainty based reversal procedure

		Attack Success	
		w/o rev.	with rev.
Normal Model	FGSM (ε = 4/255)	% 62.91	% 59.29
	BIM (l_∞, ε = 4/255)	%77.92	% 77.61
	PGD (l_∞, ε = 4/255)	%76.36	% 75.89
	Deepfool (l_∞, ε = 4/255)	% 76.23	% 0.28
	Deepfool (l_2, ε = 0.2/255)	% 73.73	% 0.12
	CW (l_2, ε = 0.2/255, conf. = 0)	% 78.26	% 4.15
	CW (l_2, ε = 0.2/255, conf. = 10)	% 74.39	% 72.50
Distilled Model	FGSM (ε = 4/255)	% 4.21	% 4.16
	BIM (l_∞, ε = 4/255)	% 4.24	% 4.24
	PGD (l_∞, ε = 4/255)	% 4.58	% 4.58
	Deepfool (l_∞, ε = 4/255)	% 77.94	% 0.64
	Deepfool (l_2, ε = 0.2/255)	% 74.95	% 0.23
	CW (l_2, ε = 0.2/255, conf. = 0)	% 79.71	% 0.16
	CW (l_2, ε = 0.2/255, conf. = 10)	% 79.73	% 1.87

Table 6.20 Effect of reversal procedure on clean performance - CIFAR10 dataset

	Model Performance	
	without reversal	with reversal
Normal Model	79,38 %	77,84 %
Distilled Model	80,47 %	79,45 %

6.10.2.1 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 6.21 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 evidenced by the outcomes, 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.

Table 6.21 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%

We then wanted to evaluate the performance of our proposed defense method to one of the most effective defense approaches in literature, which is adversarial training. The results available in Table 6.22 show that our proposed TENET architecture outperforms adversarial training in terms of robustness in all the experiments we conducted with different datasets.

Table 6.22 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%

After analyzing 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 (Simonyan & Zisserman, 2015) 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 6.23 reveal once again the efficacy of our proposed defense approach.

Table 6.23 Attack success rates on normally trained VGG19 model and VGG19 with TENET architecture

	VGG19	VGG19
	Normal Model	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%

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 seconds 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. However, we believe that this can be tolerated thanks to the robustness provided by our proposed method.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The core objective of this thesis is to leverage the concept of uncertainty in the context of adversarial machine learning. Current research in adversarial machine learning is divided into two group; attack and defense. Adversarial attacks have been studied to find new and efficient ways to craft more powerful attack types. Adversarial defense can also be divided in two categories: improving the robustness of machine learning models and detecting adversarial attempts. In this thesis, we aim to propose solutions in each of these fronts using an approach that leverages the notion of uncertainty.

In the first part of this thesis work, we investigated the utilization of different measures for detecting adversarial samples and demonstrated that moment-based predictive uncertainty estimates combined with the closeness score for predicted class obtained from the last hidden layer activation can be effectively used as a tool for a successful defense mechanism against adversarial threats. We investigated and validated the efficacy of our strategy using four benchmark datasets that are widely utilized in the adversarial research field. Our extensive studies indicate that our suggested technique generates high ROC-AUC scores on a variety of datasets and generalizes well across diverse attack types. Last but not least, we have shown the contribution of different metrics to adversarial sample identification under attack threats of varying strengths.

In the second part of this thesis work, we initially presented novel attack methods by perturbing the input in a direction that maximizes the model's epistemic uncertainty instead of its loss. When compared to loss-based techniques, we found nearly identical results. 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 towards gradient directions of both loss and uncertainty in each gradient descent step. We demonstrated that this strategy significantly increases the attack success rate. Subsequently, we suggested a more efficient attack idea based on the combined usage of the model's epistemic uncertainty and the model's loss function. We experimentally showed that our rectified attack variants outperform their original counterparts which rely solely on the model loss function. We also showed that our rectified attack algorithm achieves higher attack rates by applying less amount of perturbation. In essence, our goal was to demonstrate the existence of strong measures other than model loss that might be used to create adversarial samples. We intend to contribute to future research attempting to create more efficient and uncertainty-aware techniques by approaching the problem from a new angle and offering other ways of crafting adversarial samples. Furthermore, we empirically showed that relying just on the trained model may not always be the best course of action because it is only a rough approximation of the best predictor, but knowing the degree of epistemic uncertainty may be highly helpful in situations where the model is inaccurate.

We commenced the last part of this study by developing a unique uncertainty-based reversal operation that may be utilized as a pre-processing module before feeding any input into a deployed model. Our technique delivers a high level of adversarial robustness for adversarial samples positioned nearby model's decision boundary. We then considerably improved the uncertainty-based reversal method, evaluated the use of several uncertainty metrics, and coupled our suggested method with well-known adversarial defense methodologies such as defensive distillation and adversarial training. On three distinct datasets that are extensively used in the adversarial research area, we examined and confirmed the effectiveness of our technique. Our thorough studies show that the suggested architecture generalizes successfully across different datasets and exhibits a significant level of adversarial robustness.

Adversarial ML is one of the important topics that is intensively being studied and robustness of AI models is an extremely important concern that needs to be secured as part of the trustworthy AI research. However, there are also other important principles which needs to be guaranteed for the successful deployment of AI-driven systems in our everyday life. These principles include preserving the privacy of the user data together with the serving AI model and introducing some degree of explainability to the predictions of AI-based systems. As a future work, we believe that it could be interesting to investigate the role of uncertainty in privacy-preserving ML and Explainable-AI (XAI) research. For instance, most of the existing methods in XAI investigate the role of each input feature on the prediction probability score of the AI models. It could be worthwhile to investigate whether the deviations in these features would also yield to high uncertainty as we have thoroughly studied in developing our attack algorithms.

REFERENCES

Akan, A. K., Genc, M. A., & Vural, F. T. Y. (2020). Just noticeable difference for machines to generate adversarial images. In *2020 ieee international conference on image processing (icip)* (pp. 1901–1905). doi:10.1109/ICIP40778.2020.9191090

Aladag, M., Catak, F. O., & Gul, E. (2019). Preventing data poisoning attacks by using generative models. In *2019 1st international informatics and software engineering conference (ubmyk)* (pp. 1–5). doi:10.1109/UBMYK48245.2019.8965459

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. *Journal of Systems and Software*, *167*, 110617. doi:https://doi.org/10.1016/j.jss.2020.110617

Andriushchenko, M., Croce, F., Flammarion, N., & Hein, M. (2020). Square attack: A query-efficient black-box adversarial attack via random search. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer vision – eccv 2020* (pp. 484–501). Cham: Springer International Publishing.

Antonelli, F., Cortellessa, V., Gribaudo, M., Pinciroli, R., Trivedi, K. S., & Trubiani, C. (2020). Analytical modeling of performance indices under epistemic uncertainty applied to cloud computing systems. *Future Generation Computer Systems*, *102*, 746–761. doi:https://doi.org/10.1016/j.future.2019.09.006

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved from http://arxiv.org/abs/1409.0473

Bengio, Y., Mesnil, G., Dauphin, Y., & Rifai, S. (2013). Better mixing via deep representations. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning* (Vol. 28, pp. 552–560). Atlanta, Georgia, USA: PMLR. Retrieved from http://proceedings.mlr.press/v28/bengio13.html

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is np-complete. *Neural Networks*, *5*(1), 117–127. doi:https://doi.org/10.1016/S0893-6080(05)80010-3

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd* *international conference on machine learning* (Vol. 37, pp. 1613–1622). Lille, France: PMLR. Retrieved from http://proceedings.mlr.press/v37/blundell15.html

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., . . . Kurakin, A. (2019). On evaluating adversarial robustness. arXiv: 1902.06705 [cs.LG]

Carlini, N., & Wagner, D. (2017a). Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. arXiv: 1711.08478 [cs.LG]

Carlini, N., & Wagner, D. (2017b). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)* (pp. 39–57). doi:10.1109/SP.2017.49

Carlini, N., & Wagner, D. A. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE security and privacy workshops, SP workshops 2018, san francisco, ca, usa, may 24, 2018* (pp. 1–7). doi:10.1109/SPW.2018.00009

Causey, J. L., Zhang, J., Ma, S., Jiang, B., Qualls, J. A., Politte, D. G., . . . Huang, X. (2018). Highly accurate model for prediction of lung nodule malignancy with ct scans. *Scientific Reports*, *8*(1), 9286. doi:10.1038/s41598-018-27569-w

Chen, J., Jordan, M. I., & Wainwright, M. J. (2020). Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)* (pp. 1277–1294). doi:10.1109/SP40000.2020.00045

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). ZOO. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*. doi:10.1145/3128572.3140448

Chen, S., Carlini, N., & Wagner, D. (2020). Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st acm workshop on security and privacy on artificial intelligence* (pp. 30–39). doi:10.1145/3385003.3410925

Chouard, T. (2016). The go files: Ai computer wraps up 4-1 victory against human champion. *Nature*. doi:10.1038/nature.2016.19575

Ciodaro, T., Deva, D., de Seixas, J. M., & Damazio, D. (2012). Online particle detection with neural networks based on topological calorimetry information. *Journal of Physics: Conference Series*, *368*, 012030. doi:10.1088/1742-6596/368/1/012030

Collier, M., Mustafa, B., Kokiopoulou, E., Jenatton, R., & Berent, J. (2020). A simple probabilistic method for deep classification under input-dependent label noise. doi:10.48550/ARXIV.2003.06778

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, *2*(4), 303–314. doi:10.1007/BF02551274

Feinman, R., Curtin, R. R., Shintre, S., & Gardner, A. B. (2017). Detecting adversarial samples from artifacts. In *International conference on machine learning*.

Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., & Kohane, I. S. (2019). Adversarial attacks on medical machine learning. *Science*, *363*(6433), 1287–1289.

Gal, Y., & Ghahramani, Z. [Zoubin]. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (Vol. 48, pp. 1050–1059). New York, New York, USA: PMLR. Retrieved from http://proceedings.mlr.press/v48/gal16.html

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452–459. On Probabilistic models. doi:10.1038/nature14541

Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Iclr2014*.

Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International conference on learning representations*. Retrieved from http://arxiv.org/abs/1412.6572

Graves, A. (2011). Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 24, pp. 2348–2356). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf

Greenspan, H., van Ginneken, B., & Summers, R. M. (2016). Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, *35*(5), 1153–1159. doi:10.1109/TMI.2016.2553401

Guan, J., Tu, Z., He, R., & Tao, D. (2021). Few-shot backdoor defense using shapley estimation. doi:10.48550/ARXIV.2112.14889

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... Webster, D. R. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, *316*(22), 2402–2410. doi:10.1001/jama.2016.17216. eprint: https://jamanetwork.com/journals/jama/articlepdf/2588763/joi160132.pdf

Gurevich, P., & Stuke, H. (2019). Pairing an arbitrary regressor with an artificial neural network estimating aleatoric uncertainty. *Neurocomputing*, *350*, 291–306. doi:https://doi.org/10.1016/j.neucom.2019.03.031

Guynn, J. (2015). Google photos labeled black people 'gorillas', USA Today.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (pp. 770–778). doi:10.1109/CVPR.2016.90

Hinton, G. E., & Neal, R. (1995). Bayesian learning for neural networks.

Hinton, G. E., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, *abs/1503.02531*. arXiv: 1503.02531. Retrieved from http://arxiv.org/abs/1503.02531

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. W. (2013). Stochastic variational inference. *J. Mach. Learn. Res.*, *14*(1), 1303–1347. Retrieved from http://dblp.uni-trier.de/db/journals/jmlr/jmlr14.html#HoffmanBWP13

Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., ... 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. doi:https://doi.org/10.1016/j.cosrev.2020.100270

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Mach. Learn.*, *110*(3), 457–506. doi:10.1007/s10994-021-05946-3

Ilyas, A., Engstrom, L., & Madry, A. (2019). Prior convictions: Black-box adversarial attacks with bandits and priors. In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*, OpenReview.net. Retrieved from https://openreview.net/forum?id=BkMiWhR5K7

Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Found. Trends Comput. Graph. Vis.*, *12*(1-3), 1–308. doi:10.1561/0600000079

Jin, G., Shen, S., Zhang, D., Dai, F., & Zhang, Y. (2019). Ape-gan: Adversarial perturbation elimination with gan. In *Icassp 2019 - 2019 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 3842–3846). doi:10.1109/ICASSP.2019.8683044

Judd, J. S. (1990). *Neural network design and the complexity of learning*. Cambridge, MA, USA: MIT Press.

Justesen, N., Bontrager, P., Togelius, J., & Risi, S. (2020). Deep learning for video game playing. *IEEE Transactions on Games*, *12*(1), 1–20. doi:10.1109/TG.2019.2896986

Katzir, Z., & Elovici, Y. (2021). Gradients cannot be tamed: Behind the impossible paradox of blocking targeted adversarial attacks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 128–138. doi:10.1109/TNNLS.2020.2977142

Kendall, A. (2018). *Geometry and uncertainty in deep learning for computer vision* (Doctoral dissertation, University of Cambridge).

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30), Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf

Krizhevsky, A., Nair, V., & Hinton, G. (n.d.). Cifar-10 (canadian institute for advanced research). Retrieved from http://www.cs.toronto.edu/~kriz/cifar.html

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial examples in the physical world. *ICLR Workshop*. Retrieved from https://arxiv.org/abs/1607.02533

Kurakin, A., Goodfellow, I. J., & Bengio, S. (2016). Adversarial machine learning at scale. *CoRR*, *abs/1611.01236*. arXiv: 1611.01236

Kwon, Y., Won, J.-H., Kim, B. J., & Paik, M. C. (2020). Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics and Data Analysis*, *142*, 106816. doi:https://doi.org/10.1016/j.csda.2019.106816

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30), Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi:10.1038/nature14539

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. Retrieved from http://yann.lecun.com/exdb/mnist/

Lee, J. A., & Verleysen, M. (2007). *Nonlinear dimensionality reduction* (1st). Springer Publishing Company, Incorporated.

Li, H., Shan, S., Wenger, E., Zhang, J., Zheng, H., & Zhao, B. Y. (2022). Blacklight: Scalable defense for neural networks against Query-Based Black-Box attacks. In *31st usenix security symposium (usenix security 22)* (pp. 2117–2134). Boston, MA: USENIX Association. Retrieved from https://www.usenix.org/conference/usenixsecurity22/presentation/li-huiying

Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., & Zhu, J. (2018). Defense against adversarial attacks using high-level representation guided denoiser. arXiv: 1712.02976 [cs.CV]

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). doi:10.1109/ICCV.2019.00303

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. doi:10.1109/lra.2020.2974682

Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1412–1421). doi:10.18653/v1/D15-1166

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., & Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, *55*(2), 263–274. PMID: 25635324. doi:10.1021/ci500747n. eprint: https://doi.org/10.1021/ci500747n

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*, OpenReview.net. Retrieved from https://openreview.net/forum?id=rJzIBfZAb

McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., & Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI-17* (pp. 4745–4753). doi:10.24963/ijcai.2017/661

Meng, D., & Chen, H. (2017). Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 acm sigsac conference on computer and communications security* (pp. 135–147). doi:10.1145/3133956.3134057

Metzen, J. H., Genewein, T., Fischer, V., & Bischoff, B. (2017). On detecting adversarial perturbations. arXiv: 1702.04267 [stat.ML]

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr).*

Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). Deepfool: A simple and accurate method to fool deep neural networks. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (pp. 2574–2582). doi:10.1109/CVPR.2016.282

Morgulis, N., Kreines, A., Mendelowitz, S., & Weisglass, Y. (2019). Fooling a real car with adversarial traffic signs. arXiv: 1907.00374 [cs.CR]

Paisley, J., Blei, D. M., & Jordan, M. I. (2012). Variational bayesian inference with stochastic search. In *In icml.*

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 acm on asia conference on computer and communications security* (pp. 506–519). doi:10.1145/3052973.3053009

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 ieee symposium on security and privacy (sp)* (pp. 582–597). doi:10.1109/SP.2016.41

Raghunathan, A., Steinhardt, J., & Liang, P. (2018). Certified defenses against adversarial examples. In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*, OpenReview.net. Retrieved from https://openreview.net/forum?id=Bys4ob-Rb

Rauber, J., Brendel, W., & Bethge, M. (2018). Foolbox: A python toolbox to benchmark the robustness of machine learning models. arXiv: 1707.04131 [cs.LG]

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 ieee conference on computer vision and pattern recognition (cvpr)* (pp. 779–788). doi:10.1109/CVPR.2016.91

Reinhold, J. C., He, Y., Han, S., Chen, Y., Gao, D., Lee, J., ... Carass, A. (2020). Finding novelty with uncertainty. arXiv: 2002.04626 [eess.IV]

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28), Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Sato, M., Suzuki, J., Shindo, H., & Matsumoto, Y. (2018). Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18* (pp. 4323–4330). doi:10.24963/ijcai.2018/601

Senge, R., Bösner, S., Dembczyński, K., Haasenritter, J., Hirsch, O., Donner-Banzhoff, N., & Hüllermeier, E. (2014). Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, *255*, 16–29. doi:https://doi.org/10.1016/j.ins.2013.07.030

Shen, L., Margolies, L. R., Rothstein, J. H., Fluder, E., McBride, R., & Sieh, W. (2019). Deep learning to improve breast cancer detection on screening mammography. *Scientific Reports*, *9*(1), 12495. doi:10.1038/s41598-019-48995-4

Shridhar, K., Laumann, F., & Liwicki, M. (2018). Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. doi:10.48550/ARXIV.1806.05978

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved from http://arxiv.org/abs/1409.1556

Sitawarin, C., Bhagoji, A. N., Mosenia, A., Chiang, M., & Mittal, P. (2018). DARTS: deceiving autonomous cars with toxic signs. *CoRR*, *abs/1802.06430*. arXiv: 1802.06430. Retrieved from http://arxiv.org/abs/1802.06430

Smith, L., & Gal, Y. (2018). Understanding measures of uncertainty for adversarial example detection. In A. Globerson & R. Silva (Eds.), *Proceedings of the thirty-fourth conference on uncertainty in artificial intelligence, UAI 2018, monterey, california, usa, august 6-10, 2018* (pp. 560–569). AUAI Press. Retrieved from http://auai.org/uai2018/proceedings/papers/207.pdf

Stutz, D. (2022). *Understanding and improving robustness and uncertainty estimation in deep learning*. doi:http://dx.doi.org/10.22028/D291-37286

Stutz, D., Hein, M., & Schiele, B. (2020). Confidence-calibrated adversarial training: Generalizing to unseen attacks. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (Vol. 119, pp. 9155–9166). PMLR. Retrieved from https://proceedings.mlr.press/v119/stutz20a.html

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., & Gao, Y. (2018). Is robustness the cost of accuracy? – a comprehensive study on the robustness of 18 deep image classification models. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer vision – eccv 2018* (pp. 644–661). Cham: Springer International Publishing.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., & Fergus, R. (2014). Intriguing properties of neural networks. In Y. Bengio & Y. LeCun (Eds.), *2nd international conference on learning representations, ICLR 2014, banff, ab, canada, april 14-16, 2014, conference track proceedings*. Retrieved from http://arxiv.org/abs/1312.6199

Technical Report, N. P. 1.-0. (Jan 2017). NHTSA PE 16-007 Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration.

Tuna, O., Catak, F., & Eskil, M. (2022). Unreasonable effectiveness of last hidden layer activations for adversarial robustness. In *2022 ieee 46th annual computers, software, and applications conference (compsac)* (pp. 1098–1103). doi:10.1109/COMPSAC54236.2022.00172

Tuna, O. F., Catak, F. O., & Eskil, M. T. (2022a). Closeness and uncertainty aware adversarial examples detection in adversarial machine learning. *Computers and Electrical Engineering*, *101*, 107986. doi:https://doi.org/10.1016/j.compeleceng.2022.107986

Tuna, O. F., Catak, F. O., & Eskil, M. T. (2022b). Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples. *Multimedia Tools and Applications*, *81*(8), 11479–11500. doi:10.1007/s11042-022-12132-7

Tuna, O. F., Catak, F. O., & Eskil, M. T. (2022c). Uncertainty as a swiss army knife: New adversarial attack and defense ideas based on epistemic uncertainty. *Complex & Intelligent Systems*. doi:10.1007/s40747-022-00701-0

Wu, D., & Wang, Y. (2021). Adversarial neuron pruning purifies backdoored deep models. doi:10.48550/ARXIV.2110.14430

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. arXiv: 1708.07747 [cs.LG]

Xingjun Ma, J. B., Bo Li. (2018). Characterizing adversarial subspaces using local intrinsic dimensionality. In *Iclr*.

Yang, P., Chen, J., Hsieh, C., Wang, J., & Jordan, M. I. (2020). ML-LOO: detecting adversarial examples with feature attribution. In *The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, new york, ny, usa, february 7-12, 2020* (pp. 6639–6647). AAAI Press. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/6140

Zheng, R., Zhang, S., Liu, L., Luo, Y., & Sun, M. (2021). Uncertainty in bayesian deep label distribution learning. *Applied Soft Computing*, *101*, 107046. doi:https://doi.org/10.1016/j.asoc.2020.107046

Zhou, D. (2018). Universality of deep convolutional neural networks. *CoRR*, *abs/1805.10769*. arXiv: 1805.10769. Retrieved from http://arxiv.org/abs/1805.10769

CURRICULUM VITAE