

March 2022

Adversarial and Data Poisoning Attacks against Deep Learning

Jing Lin
University of South Florida

Follow this and additional works at: <https://digitalcommons.usf.edu/etd>



Part of the [Statistics and Probability Commons](#)

Scholar Commons Citation

Lin, Jing, "Adversarial and Data Poisoning Attacks against Deep Learning" (2022). *USF Tampa Graduate Theses and Dissertations*.
<https://digitalcommons.usf.edu/etd/10319>

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.

Adversarial and Data Poisoning Attacks against Deep Learning

by

Jing Lin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mathematics & Statistics
College of Arts and Sciences
University of South Florida

Major Professor: Kaiqi Xiong, Ph.D.
Kandethody M. Ramachandran, Ph.D.
Lu Lu, Ph.D.
Nasir Ghani, Ph.D

Date of Approval:
March 24, 2022

Keywords: Cybersecurity, Machine Learning, Machine Learning Security, Adversarial Example,

Copyright © 2022, Jing Lin

Dedication

I dedicate this dissertation to my family. Without their support, understanding, and unconditional love, the completion of this work would not have been possible.

Acknowledgments

I am deeply indebted to the following people: my major professor, Dr. Kaiqi Xiong, my committee members, Dr. Kandethody M. Ramachandran, Dr. Lu Lu, and Nasir Ghani, my mentors Dr. Ryan S. Luley and Dr. Laurent Njilla at Air Force Research Lab, who have supported me throughout my dissertation.

Finally, I would like to acknowledge that my graduate studies have been partially funded by the Air Force Research Laboratory under agreement number FA8750-20-3-1004 and the National Science Foundation (NSF) under grants CNS 1620862 and CNS 1620871, and BBN/GPO project 1936 through an NSF/CNS grant.

Table of Contents

List of Tables	iii
List of Figures	iv
Abstract	vi
Chapter 1 Introduction	1
1.1 Adversarial Examples	1
1.2 Malicious Mislabeling and Targeted Triggerless Clean-label Data Poisoning Attacks	2
1.3 Motivations	2
1.4 Problem Statement	2
1.5 Proposed Work and Contributions	3
Chapter 2 Background	4
2.1 Machine Learning Basics	4
2.1.1 Support Vector Machines	6
2.1.2 Feed-forward Neural Networks	7
2.1.3 Recurrent Neural Networks	10
2.1.4 Deep Learning Performance	11
2.2 Adversarial Attacks and Data Poisoning Attacks	12
2.2.1 White-box Adversarial Attacks	14
2.2.2 Black-box Adversarial Attacks	26
2.2.3 Data Poisoning Attacks	36
2.2.4 Summary	43
2.3 Active Learning Methods	45
Chapter 3 Secure Machine Learning Against Adversarial Samples at Test Time	48
3.1 Introduction	49
3.2 Threat Model	51
3.3 Related Work	51
3.4 Methodology	55
3.5 Evaluation	57
3.5.1 Datasets	57
3.5.2 Network Architectures	58
3.5.3 Adversarial Crafting	58
3.5.4 Results	59
3.6 Discussion	60

3.6.1	Accuracy vs. Resilience Against Adversarial Examples	61
3.6.2	Training Time	61
3.6.3	Ensembling	65
3.6.4	Limitations	66
3.7	Conclusions and Future Work	68
Chapter 4	Active Learning Under Malicious Mislabeling and Poisoning Attacks	69
4.1	Introduction	69
4.2	Threat Model	70
4.3	Related Work	74
4.4	Methodology	75
4.4.1	Performance Analysis	77
4.4.2	Querying Strategy	78
4.4.3	Artificial Data Crafting	79
4.5	Evaluation	80
4.5.1	Dataset and Model	81
4.5.2	Artificial Data Generation and Data Poisoning Attack	82
4.5.3	Mislabeling	83
4.5.4	Results	83
4.5.5	Computation Cost and Limitation	86
4.6	Conclusions and Future Work	87
Chapter 5	An Adversarial Attack Method for Poisoning a Transfer Learning Model	89
5.1	Introduction	89
5.2	Threat Model	91
5.3	Related Work	91
5.3.1	Transfer Learning	91
5.3.2	Data Poisoning Attacks	92
5.4	Methodology	93
5.5	Evaluation	96
5.5.1	Comparison to Existing Studies	97
5.5.2	Effect of a Dropout Layer and Optimizer	99
5.5.3	Effect of Upsampling and Downsampling on Attack Success Rate	100
5.5.4	Effect of Dataset Type, Dataset Size, and Pre-trained Models	101
5.6	Conclusions and Future Work	106
Appendix A	Copyright Permissions	124

List of Tables

Table 1	Training Parameters	58
Table 2	Classification Accuracy on the MNIST Test Dataset Under the White-box Attack	60
Table 3	Classification Accuracy on the MNIST Test Dataset Under the Grey-box Attack	60
Table 4	Classification Accuracy on the CIFAR-10 Test Dataset Under White-box Attacks	61
Table 5	GPUs vs. CPU	63
Table 6	Notation	72
Table 7	CIFAR-10 vs. CIFAR-2	81
Table 8	Statistics of Labeling Budgets	83
Table 9	Summary Statistics for Labeling Budget Under a Malicious Mislabeled Attack	84
Table 10	Summary Statistics for Labeling Budget Under a Poison Attack	86
Table 11	Adversarial Image Generation vs. Triggerless Clean-Label Data Poisoning	95
Table 12	Dataset Summary: CIFAR vs. STL	97
Table 13	Similarity Coefficients	98
Table 14	Effect of Dropout Rate on Attack Success Rate (L-BFGS-B)	100
Table 15	Effect of Dropout Rate on Attack Success Rate (PGD)	100
Table 16	Effect of Upsampling and Downsampling on Attack Success Rate	101
Table 17	Average Reduction in Test Accuracy	104

List of Figures

Figure 1	Illustration of an Adversarial Example	1
Figure 2	Active Learning Framework	46
Figure 3	Training and Validation Loss	62
Figure 4	Parallelization Framework	64
Figure 5	Adversarial Image Generation Speed for BIM Attacks	65
Figure 6	Adversarial Image Generation Speed for C&W Attacks	66
Figure 7	Reduction in Attack Success Rate	67
Figure 8	Active Learning Framework	74
Figure 9	The Proposed Active Learning Framework	76
Figure 10	Confidence Score Updating	78
Figure 11	An Example of Targeted Poisoning Attacks	79
Figure 12	Baseline Model Performance Under a Malicious Labeling Attack	85
Figure 13	Robust Active Learning Model Performance Under a Malicious Labeling Attack	86
Figure 14	Baseline Model Performance Under a Poisoning Attack	87
Figure 15	Robust Active Learning Model Performance Under a Poisoning Attack	88
Figure 16	Evasion Attack vs. Triggerless Clean Label Data Poisoning Attack	94
Figure 17	Poison Instances Created by the FA Attack	98
Figure 18	Poison Instances Created by the One-shot Attack	99
Figure 19	Distribution of Standard Test Accuracy for CIFAR-2 and CIFAR-10	102
Figure 20	Distribution of Standard Test Accuracy for ResNet-50 and VGG-19	102
Figure 21	Distribution of Standard Test Accuracy with Different Training Set Size	103
Figure 22	Distribution of Attack Success Rate for CIFAR-2 and CIFAR-10	104
Figure 23	Distribution of Attack Success Rate for ResNet-50 and VGG-19	105
Figure 24	QQ Plot: Check Normality	106
Figure 25	2020 IEEE ICC Copyright Permission	124
Figure 26	2021 IEEE Global Communications Conference Copyright Permission	124

Figure 27	SpringerOpen Copyright Permission	125
Figure 28	Open Access Statement	125
Figure 29	Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)	126
Figure 30	ArXiv License Information	126

Abstract

Machine translation software, image captioning, grammar check (Grammarly), chatbot, real-time captioning and translation, music genre classification, and document classification are a few examples of deep learning applications that achieve outstanding performance in areas where traditional statistical techniques have difficulty performing classification and/or regression. Google translator has over 100 billion daily users and can translate 109 languages instantly (much faster than a human translator). AlphaGo won Lee Sedol, the eighteen-time world champion. Microsoft Team's living captioning provides accurate real-time captioning as a speaker speaks. Deep learning achieves undoubtedly remarkable performance. However, recent studies on adversarial attacks and data poisoning attacks show that deep learning models are vulnerable to these attacks.

In this dissertation, we focus on a special type of data poisoning attack, triggerless clean-label data poisoning attacks. In such an attack, some training instances are manipulated in order to cause the misclassification of a target instance (that is protected from manipulation) on a targeted attack. The main difference between a triggerless data poisoning attack and adversarial examples is that a triggerless data poisoning attack assumes an attacker cannot manipulate (targeted) test data whereas an adversarial example assumes an attacker can manipulate test data to cause misclassification. Deep learning models are vulnerable to both attacks.

This dissertation research presents solutions and mechanisms for enhancing the deep learning models from adversarial examples, malicious mislabeling, and data poisoning attacks. Specifically, a new iterative retraining approach to reducing the effectiveness of adversarial examples is presented. To overcome the time complexity issue, a parallel implementation of the proposed approach is developed to make it scalable. Moreover, an efficient active learning method is developed to defend against malicious mislabeling and data

poisoning attacks. The empirical evaluation shows its effectiveness against various adversarial attacks and data poisoning attacks while maintaining the performance on the untampered dataset. Finally, a novel approach is introduced to alter an adversarial attack method to a stealthy triggerless clean-label data poisoning attack. All presented methodologies are comprehensively evaluated on either Gaivi or Research Computing Central Instructional and Research Computing Environment (CIRCE).

Chapter 1

Introduction

This dissertation studies the adversarial examples, malicious mislabeling, data poisoning attacks against deep learning models. This chapter presents an overview of these domains, along with the motivations of this work and the threat model. Finally, the main contributions of this research are presented.

1.1 Adversarial Examples



Figure 1. Illustration of an Adversarial Example. Reprinted from Lin, J., Njilla, L.L. & Xiong, K. Secure machine learning against adversarial samples at test time. EURASIP J. on Info. Security 2022, 1, doi: 10.1186/s13635-021-00125-2. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Image classification, natural language processing, malware detection [151], and self-driving cars are some tasks which in deep learning performs well. However, Szegedy *et al.* [133] showed how an adversarial example, a slightly modified instance that is indistinguishable from its natural copy, can fool the state-of-the-art deep learning model in 2013. For instance, Figure 1 shows an adversarial example generated by Carlini and Wagner’s (C&W) attack [20]. An original image of ‘5’ from the MNIST dataset [82] (left) and its adversarial example (right) look the same. Nevertheless, the start-of-the-art digit classifier only labels the left image as ‘5’ but not the one on the right (as ‘3’). This demonstrates the vulnerability of deep learning models.

1.2 Malicious Mislabeling and Targeted Triggerless Clean-label Data Poisoning Attacks

Supervised learning required the labeled dataset. Hand annotating millions of data points, such as one for ImageNet, is prone to unintentional mislabeling errors (i.e., due to the fatigue of a human labeler) and malicious mislabeling, an intentional act of a malicious labeler. In this dissertation, we consider both intentional and unintentional mislabeling on the performance of the active learning models. Besides that, data points themselves are vulnerable as well. In this dissertation, we consider a particular type of targeted data poisoning attack called the triggerless targeted clean-label data poisoning attack. Triggerless because the target instance is not modified (and so it is difficult to inspect or detect the attack by monitoring the training instances). Clean-label indicates that true labels are provided for training (and so the malicious attacker does not need to control the labeling process). Hence, this attack is very stealthy and we will introduce our defense method against it in this dissertation.

1.3 Motivations

Deep learning is increasingly used in safety and security-critical applications. However, adversarial attacks and data poisoning attacks question the applicability of deep learning to these domains. For instance, the data obtained from untrusted sources, such as the Internet, could be poisoned. The model trained based on this dataset may open a backdoor, such as permitting a malicious attack to enter a secure building monitored by the surveillance cameras and face recognition system. These challenges inspired this work.

1.4 Problem Statement

This dissertation focuses on the effect mislabeling attacks, adversarial attacks, and triggerless clean-label targeted data poisoning attacks have on the deep neural networks. In particular, we consider the adversarial attack and assume a strong attacker who has access to the trained model. Then, chapter 3 presents a distributive defense framework to reduce its effectiveness. Next, chapter 4 presents the mislabeling attacks and stealthy triggerless clean-label targeted data poisoning attacks against the active learning model and the defense methodology for maintaining the performance under these attacks. Lastly, chapter 5 shows the development of the triggerless clean-label targeted data poisoning attacks against transfer learning models using adversarial examples.

1.5 Proposed Work and Contributions

This dissertation addresses the security of deep learning models. The key contributions of this dissertation are in the following:

- Developed a parallelizable iterative approach for adversarial retraining with soft labels, instead of hard labels.
- Proposed an effective active learning method under mislabeling attacks and stealthy triggerless clean-label data poisoning attacks.
- Utilized an adversarial attack method to generate a stealthy poison instance for a triggerless clean-label data poisoning attack and compared its performance against the state-of-the-art one-shot attack.

The rest of this dissertation is organized as follows. Chapter 2 presents the background information. Building upon this, Chapter 3 describe in detail the proposed approach for iterative adversarial retraining. Chapter 4 presents a new heuristic methodology for active learning under data poisoning attacks and malicious mislabeling attacks. Chapter 5 presents a new data poisoning attack method that is more effective than the state-of-the-art.

Chapter 2

Background

This chapter¹ provides a brief introduction to machine learning basics, adversarial machine learning, malicious mislabeling attacks, and data poisoning attacks.

2.1 Machine Learning Basics

Machine learning (ML) is a research field focusing on the theory, properties, and performance of learning algorithms and systems. The ML field is considered highly interdisciplinary as it is built on the top of various disciplines, including, but not limited to, statistics, information theory, cognitive science, and mathematics (e.g., optimization theory). ML techniques have been intensively studied in the past. Notably, the advancement in many recent research studies related to smart city applications has been made possible using ML advances, where AI safety is directly relevant to ML security. As another example, ML implementations in an intelligent transportation system (ITS) can be deployed to analyze data generated by the different parts of the system (e.g., roads, number of passengers, and commute mode). The analysis of collected data is consequently used for future planning and decision-making within a transportation scheme [3]. Machine learning is a burgeoning new technique, and the top six best conferences on Computer Science Community are emphasizing machine learning and deep learning according to the World Ranking of Top Computer Science Conferences in 2020. This shows the popularity, interest, and excitement of the research community towards machine learning.

In 2020, Alpaydin [5] defined machine learning as ”computer programming that can learn the rules from input and output automatically”. Many machine learning techniques exist. Examples are neural networks (as known as artificial neural networks), support vector machine (SVM), k-nearest neighborhood,

¹Portions of this chapter have been previously published by the author of this dissertation in [85, 87, 88, 89], where no formal reuse license is needed for the IEEE © [2020] publication [87], the IEEE © [2021] publication [85], and Springer Open © The Authors 2022 [88], and remixing and adapting arXiv [89] CC BY-NC-SA 2021 is permitted for their senior authors.

Ada Boosting, Gradient Boosting, Gaussian Naive Bayes, linear discriminant analysis, and random forest, and decision tree.

Deep learning (DL) is a vital component of the ML field. Unlike many traditional learning techniques that are based on shallow-structured learning paradigms, DL is based upon deep architectures, where unsupervised and/or supervised learning strategies are mainly used to autonomously learn hierarchical representations. DL architectures (deep architectures) are often enabled to capture more hierarchically launched statistical patterns (complicated inputs patterns) in comparison to shallow-structured learning architectures [117]. It automates the feature selection process. That is, you do not even need to specify the features, and neural network can extract them from raw input data. For instance, in the image classification, an image is input to the neural network and the convolutional layers in the neural network will extract the important features from the image directly. This makes deep learning desirable for many complex tasks such as natural language processing [26, 39, 47, 64], image captioning, robotics, real-time captioning and translation offered by companies such as Microsoft and Zoom, chatbot, music genre classification, self-driving cars, cancer detection, and image classification [120], [161], [67].

Google translator has over 100 billion daily-users and can translate 109 languages instantly (much faster than a human translator). AlphaGo won Lee Sedol, the eighteen-time world champion. Microsoft Team's living captioning provides accurate real-time captioning as a speaker speaks. The performance of the neural network is undoubtedly remarkable.

By the Universal Approximation Theorem, any continuous function in a compact space can be approximate by a feed-forward neural network with at least one hidden layers and suitable activation function to any desirable accuracy [10]. This theorem explains the wide application of deep neural networks. However, it does not give any constructive guideline on how to find such universal approximator. In 2017, Lu *et al.* [92] established the Universal Approximation Theorem for Width-Bounded ReLU Networks, which showed that a fully connected width- $(n + 4)$ ReLU networks, where n is the input dimension, are universal approximators. These two Universal Approximation Theorems explain the ability of deep neural network to learn.

In the following subsection, we provide some background information on support vector machines (SVMs) and feed-forward neural networks, an artificial neural network (or sometimes simply called neural network) without a cycle. Then, we will introduce a family of sequence-to-sequence models in subsection 2.1.3. Finally, section 2.1.4 provides some discussions on deep learning performance.

2.1.1 Support Vector Machines

SVMs, known as large margin classifiers, are supervised learning models for both regression and classification problems. Formally, SVMs are ML methods to construct a hyperplane such that the distance between the hyperplane and the nearest training sample is maximized.

As an example, we focus on the hard-margin SVM for a linearly separable binary classification problem here. Given a training dataset of n points written as $(x_1, y_1), \dots, (x_n, y_n)$, where y_i is the class that x_i belongs to. Each data point x_i is a p -dimensional real vector and $y_i \in \{-1, 1\}$. The goal of SVMs is to search for the maximum-margin hyperplane that separates the group of points x_i for which $y_i = 1$ from the group of points x_i for which $y_i = -1$. The SVM, proposed by Cortes and Vapnik [32], finds the maximum margin hyperplane by solving the following quadratic programming (QP) problem:

$$\frac{1}{2} \|w\|^2, \quad (2.1)$$

subject to the constraints, $y_i(w^T x_i + b) \geq 1$, for all $i = 1, 2, \dots, n$, where w is the normal vector of the maximum margin hyperplane $y = w^T x + b$. This QP problem can be reformatted as an unconstrained problem using Lagrange multipliers $\alpha_i \geq 0$, for $i = 1, 2, \dots, n$, as follows.

$$L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i [y_i (w^T x_i + b) - 1]. \quad (2.2)$$

Taking the derivative of the above expression with respect to w , we have

$$w = \sum_{i=1}^n \alpha_i x_i y_i. \quad (2.3)$$

This indicates that the normal vector w of the hyperplane depends on α_i, x_i , and y_i . Similarly, taking the derivative of the above expression with respect to w_0 , we obtain

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (2.4)$$

Substituting equations (2.3) and (2.4) to L_p , we maximize the resulting L_p with respect to α_i subject to equation (2.4) by applying a quadratic optimization method. Then, we can be easily solved for α_i . Once solved, most α_i are equal to zeros. The set of data points (x_i, y_i) for which the corresponding $\alpha_i \neq 0$ are

called support vectors. Since the normal vector $w = \sum_{i=1}^n \alpha_i x_i y_i$, the resulting hyperplane depends on the support vectors only. However, if two classes are non-linear separable, soft-margin SVM with a kernel function can be utilized to learn a non-linear decision boundary (linear in transformed space).

2.1.2 Feed-forward Neural Networks

In this subsection, we discuss feed-forward neural networks, focusing on image-recognition systems since most existing adversarial examples introduced in the literature are related to image recognition.

Let $X = R^{hwc}$ be the input image space, where $h \in Z_+, w \in Z_+,$ and $c \in Z_+.$ For instance, a gray-scale image $x \in X$ has color channel $c = 1,$ and a colored RGB image $x \in X$ has color channel $c = 3.$ The element $x_{i,j,k}$ of vector x denotes the value of pixel located at $(i, j, k).$ Each pixel value indicates how bright that pixel is, and it is an integer between 0 and 255. However, the pixel values are usually scaled to be in the range of $[0, 1].$

The logistic regression f is an extension of the linear regression function to a nonlinear function. A feed-forward neural network can be considered as an extension of the logistic regression function $f.$ A feed-forward neural network can be written as a function $F : \mathbf{X} \rightarrow \mathbf{R}^K$ that maps an input image $\mathbf{x} \in \mathbf{X}$ to an output $\mathbf{y} \in \mathbf{R}^K,$ where \mathbf{X} is the input space or sample space, \mathbf{R} is a set of real numbers, and K is the number of classes. Each element y_j of \mathbf{y} can be considered as the probability or likelihood that input x belongs to class $j,$ where j is an integer between 1 and $K.$ Each element y_j of \mathbf{y} satisfies $y_j \in [0, 1]$ and $\sum_{j=1}^K y_j = 1.$ The output vector \mathbf{y} can be considered as a probability distribution over the discrete label set $\{1, 2, \dots, K\}.$ Furthermore, let $C : \mathbf{X} \rightarrow \{1, 2, \dots, K\}$ be the corresponding classifier that maps an input image $\mathbf{x} \in \mathbf{X}$ to $\{1, 2, \dots, K\}.$ The relationship between C and the output vector \mathbf{y} is as follows:

$$C(\mathbf{x}) = \operatorname{argmax}_{i=1,2,\dots,K} y_i. \tag{2.5}$$

A feed-forward neural network F with $L \in Z_+$ layers can be considered as a composition function such that for each sample $\mathbf{x} \in \mathbf{X}, F(x) = f_L(f_{L-1}(\dots(f_1(\mathbf{x}))))),$ where each component function $f_i(\mathbf{x}) = A_i(w_i * \mathbf{x} + b_i);$ A_i is an activation function at layer $i,$ e.g., non-linear Rectified Linear Unit (ReLU), sigmoid, softmax, identity, hyperbolic tangent (tanh), etc.; w_i is a matrix of weight parameters at layer $i;$ b_i is a bias unit at layer $i;$ L is the total number of the layer in a neural network. For classification, the activation function for the last layers is usually softmax function for multiclassification (more than two classes) and

sigmoid for binary classification. Both $w = (w_1, w_2, \dots, w_L)$ and $b = (b_1, b_2, \dots, b_L)$ form model parameters $\theta = (w, b)$. To keep things simple, we do not explicitly state the above function dependence on θ . If an attack model explicitly depends on it, we would write it as $f(\mathbf{x}|\theta) = \mathbf{y}$. To find model parameters θ such that the classifier C can predict most instances in \mathbf{X} correctly, a stochastic gradient descent method is often utilized to minimize a loss function $J : X \times \{1, 2, \dots, K\} \times P \rightarrow \mathbf{R}^+$, where P is parameter space. To keep things simple, we sometimes simply use $J : X \times \{1, 2, \dots, K\} \rightarrow \mathbf{R}^+$ when there is no confusion. A loss function J is usually a measure of difference between the predicted output and its true label. A common method used to find the optimized weights is the back-propagation algorithm. For example, see [51] and [12] for an overview of the back-propagation algorithm.

The activation function $A(\cdot)$ is important in expanding the linear hypothesis space to the nonlinear hypothesis space. In fact, according to the Universal approximation theorem, a one-hidden layer feed-forward neural network can approximate any continuous function in a compact space. Let's talk a bit more about the rectified linear unit as it is so popular comparing to other activation functions, especially in an image classification task. The rectified linear unit $\text{ReLU} : R \rightarrow R$ is defined on a real number set R into a real number set R such that $y = \max(0, x)$ for each $x \in R$. Usually, data scientists are interested in a non-linear activation function that is bounded and with a smooth gradient. Boundedness is an important property in preventing the return values of the activation function from becoming overly large. A smooth gradient is desirable since it makes back-propagation possible. However, ReLU is neither upper-bounded nor non-differentiable at $x = 0$. Thus, why is it so popular and useful compared to traditional non-linear activation functions (such as sigmoid and hyperbolic tangent functions) that are bounded and with a smooth gradient? Existing studies have not laid a theoretical foundation to answer this question yet. However, we know that the only non-differentiable point for the ReLU is when $x = 0$ and an input value of exactly zero for a ReLU activation is not likely at any stage of the calculation, so back-propagation is working, although ReLU is not everywhere differentiable.

There are many variations of ReLU developed for the past few years, such as Exponential Linear Unit (Exponential LU or simply ELU) [29], Scaled Exponential Linear Unit (SELU) [78], Gaussian Error Linear Unit (GELU) [63], Leaky ReLU [93], Randomized Leaky Rectified Linear Activation (RLReLU), and Parametric ReLU [61]. Instead of assign all negative input zero, ELU transforms the negative input through an exponential function. That is,

$$ELU(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0, \end{cases}$$

where $\alpha > 0$ is commonly selected to be between 0.1 and 0.3. SELU has one extra parameter λ compared to ELU.

$$SELU(x) = \lambda \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0, \end{cases}$$

The value of λ and α is predetermined.

$$\alpha = 1.6732632423543772848170429916717$$

$$\lambda = 1.0507009873554804934193349852946$$

GELU, an empirical improvement of the ReLU and ELU activations and is the state-of-the-art activation function in NLP for past few years, is defined as follows:

$$\text{GELU}(x) = \frac{x}{2} \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right)$$

The Leaky ReLU scales down the negative input value through multiplying it by a small positive constant $a \in R^+$ instead of returning zero when $x < 0$. As the name implies, RLReLU adds a random component to Leaky ReLU as follows.

$$RLReLU(x) = \begin{cases} x & x \geq 0 \\ ax & x < 0, \end{cases}$$

where a is randomly selected from a uniform distribution with mean 0 and standard deviation 1 for RLReLU activation. RLReLU can be used for regularization purpose when a model is overfitted. However, a can also be considered as a hyperparameter. This is the case for Parametrized ReLU. Parametrized ReLU is usually used only when the training data size is big.

Lastly, we discuss the softmax function, a popular activation function for the last layer of a multi-classification model. The softmax function is a generalization of the logistic function or sigmoid function, and can be defined as follows: $Softmax : R^K \rightarrow [0, 1]^K$ such that i -th element of the output for an input $x \in R^K$ is

$$Softmax_i(x) = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)}. \quad (2.6)$$

2.1.3 Recurrent Neural Networks

In this subsection, we introduce a family of neural networks for processing the sequence data, the recurrent neural networks (RNNs). An RNN consists of a sequence of RNN cells. Each RNN cell does a two steps computation to extract some information and store it in the hidden state $a^{<t>}$. More precisely, an input sequence $x^{<t>}$ at time t and a hidden state $a^{<t-1>}$ at time $t - 1$ are input to an RNN cell at step 1. Using them the hidden state $a^{<t>}$ at time t and an output sequence $\hat{y}^{<t>}$ at time t are calculated as follows:

$$a^{<t>} = \tanh(W_{ax}x^{<t>} + W_{aa}a^{<t-1>} + b_a)$$

$$\hat{y}^{<t>} = \text{softmax}(W_{ya}a^{<t>} + b_y)$$

where W_{ax} , W_{aa} and W_{ya} are weighted matrices that are optimized by minimizing a defined loss function (such as the cross-entropy loss for classification task or the mean square error (MSE) or the mean absolute error (MAE) for regression task); b_a and b_y are biased terms; \tanh is hyperbolic tangent function; and softmax is the softmax function define on a k dimensional space R^k into R^k such that for each input $z = [z_1, z_2, \dots, z_k]^T \in R^k$, we have an output $\text{softmax}(z)$ whose i th element is

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}.$$

With these RNN cells as building blocks, there are many variations of RNN architectures possible. A simple RNN architecture that consists of sequence of RNN cells is a row is a type of many-to-many architecture. Other types of RNN architectures are one-to-many, and many-to-one. Hence, RNN architecture is very flexible comparing to the feed-forward neural network, introduced in section 2.1.2, which has one-to-one correspondence (one image input and one output indicates the class if the task is classification). One-to-one architecture is too constrained in many cases. For instance, an image capturing task requires a model

to output a sentence that describes a single input image (one-to-many architecture is needed here). That is, a single image as an input and a sequence of words (many) as outputs. A sentimental analysis of a movie review requires a model to read a movie review and then output a word that describes the overall sentiment implied in the review (many-to-one architecture is needed here). That is, a long sequence of words as inputs and one word indicates the sentiment as output. Machine translation, e.g., translation from English to Spanish, requires an input of a sequence of words (paragraph, sentence, whole article/document in English). And it also outputs a sequence of words, e.g., paragraph, sentence, whole article/document in Spanish (many-to-many architecture is needed here).

Theoretically, a simple RNN model can retain the information over many timesteps with the hidden state $a^{<t>}$. In practice, it is difficult to learn long-term dependencies due to the vanish gradient problem [11]. In 1997, Hochreiter and Schmidhuber [65] came up with a solution, Long-Short-Term-Memory (LSTM). LSTM adds a “conveyer belt” c to carry information across many timesteps. Hence, this “conveyer belt” saves the information from the sequence and the model can extract the saved information when needed. This solves the vanish gradient problem faced by a simple RNN model. To be more precise, there are three gates within the cell control the flow of the information. All three gates (forget gate, update gate, and output gate) are defined based on the sigmoid function. The forget gate Γ_f deliberately forgets irrelevant information in the carry dataflow. The update gate Γ_u updates the information about the present state. The output gate Γ_o controls how much information from the cell state/memory variable $c^{<t>}$ should be included in the output.

An LSTM layer is more computation-intensive comparing to a simple RNN layer. To reduce the computation intensity without incurring vanish gradient problem, another variation of RNN called gated recurrent unit (GRU) is developed in 2014 [27]. GRU is similar to LSTM but with only two gates (forget gate and reset gate). The performance of the GRU on music and signal processing modeling and NLP are similar to LSTM. However, GRU exhibits better performance on smaller and less frequent dataset. There are other more advancing variations such as bidirectional RNN and stacked RNNs. We will not discuss them here as the sentimental analysis on a movie review dataset is a simple task that do not require such advanced techniques.

2.1.4 Deep Learning Performance

The number of deep learning applications has exploded over the past decade. The performance of feed-forward neural networks largely depends on

- model complexity
- representative dataset
- size of dataset
- computational power

With a large amount of data, a feed-forward neural network can extract high-level features from raw data and obtain an effective representation of input space. In addition, model complexity increase with the complexity of the task in order to learn a thorough representation of input space. As the model got more complex (with hundreds of layers) and dataset size becomes bigger and bigger (petabytes or terabytes), the number of computational resources needed to solve a task increase as well. In fact, computational power is so important to the development of deep learning. As 1943, McCulloch and Pitts [96] already introduced the neural network idea; however, it is only until late 2000s deep learning can be trained due to advancement in technology. The large amount of data generated every day and available for analysis requires even better computational resources.

2.2 Adversarial Attacks and Data Poisoning Attacks

Many state-of-the-art ML models have outperformed humans in various tasks such as image classification [61]. However, the existence of adversarial attacks and data poisoning attacks really questions the robustness of ML models. For instance, Engstrom [43] demonstrated that the state-of-the-art image classifiers can be easily fooled by a small rotation on an arbitrary image. As ML systems are being increasingly integrated into safety and security-sensitive applications [66], adversarial attacks, and data poisoning attacks pose a considerable threat. This section focuses on the two broad and important areas of ML security: adversarial attacks and data poisoning attacks.

In adversarial attacks, attackers attempt to perturb a data point x to an adversarial data point x' so that x' is misclassified by an ML model with high confidence, although x' is visually indistinguishable from its original data point x by humans. A visual illustration of an adversarial attack is shown in Figure 1, which presents a C&W attack. Adversarial attacks can be conducted in different domains, such as audio [21], text [124], network signals [31], and images [112]. They can be further classified based upon the knowledge

level that adversaries maintain about a target model. The attack model here can be specified as either black-box, white-box, or gray-box. Black-box attacks refer to the case when an attacker has no access to the trained ML model. The attacker only knows the model outputs (logit, confident score, or label). Black-box attacks are commonly observed in online learning techniques used in anomaly detection systems that effectively retrain the detector when new data is available. In a gray-box attack, attackers may have some incomplete knowledge about the overall structure of a target model. Last, in a white-box threat model, attackers have complete knowledge about a target model, which therefore facilitates the task of generating adversarial examples and crafting poisoned input data. Even though there are more white-box attacks, black-box attacks are more practical attacks on ML systems because essential model information is often confidential and protected from a normal user interacting with a system. However, due to the phenomenon of attack transferability, white-box attacks can be utilized to attack black-box models (transfer attacks). In this chapter, we focus on the two primary threat models, white-box and black-box attacks.

In data poisoning attacks, adversaries try to manipulate training data in an attempt

- to decrease the overall performance (i.e., accuracy) of an ML model or
- to induce misclassification to a specific test sample or a subset of the test sample or
- to increase training time.

If an attacker has a specified target label to which a specific test sample is misclassified, the attack is called a targeted data poisoning attack; otherwise, it is called an untargeted data poisoning attack. Adversaries are assumed to either be able to contribute to the training data or have control over the training data. For instance, crowdsourcing is regarded as a vital data source for various smart cities, such as intelligent transportation systems and traffic monitoring [83]. Hence, such systems are vulnerable to data poisoning threats [70, 99] as data poisoning attacks are often difficult to be detected [98].

This section present adversarial attacks and data poisoning attacks in both white-box and black-box settings. These security attack classes are comprehensively discussed from different adversarial capabilities. The main objective of this section is to present the existing adversarial attacks and data poisoning attacks. The rest of this section is organized as follows: In section 2.2.1, we introduce some important definitions and typical white-box adversarial attacks. In section 2.2.2, we further discuss black-box adversarial attacks. Moreover, section 2.2.3 covers data poisoning attacks, along with their adversarial aspects. Finally, section 2.2.4 summarize the section.

2.2.1 White-box Adversarial Attacks

We start with a reasonably comprehensive definition of adversarial attacks based on L_p -norm.

Definition 1. (Adversarial Attack): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . Given a target class t such that $t \neq y$. An adversarial attack is a mapping $\alpha : R^n \rightarrow R^n$ such that the adversarial example $\alpha(x) = x'$ is misclassified as class t by F , whereas the difference between x' and x is trivial, i.e., $\|x' - x\|_p < \epsilon$ for some small value ϵ .

The L_p -norm $\|x' - x\|_p$ used in Definition 1 measures the magnitude of perturbation generated by an adversarial attack α . The L_p -norm of a vector v is defined as

$$\|v\|_p = \left(\sum_{i=1}^n v_i^p \right)^{1/p},$$

where v_i is the i -th component of v . Here are three frequently used norms in literature.

- L_0 -norm:

$$\|x' - x\|_0 = \sum_{i=1}^n I[x'_i \neq x_i],$$

where

$$I[x'_i \neq x_i] = \begin{cases} 1 & x'_i \neq x_i \\ 0 & x'_i = x_i, \end{cases}$$

counts the number of coordinates i such that $x_i \neq x'_i$, where x_i and x'_i are the i -th component of x and x' , respectively. The norm is useful when an attacker wants to limit the number of attack pixels without limiting the size of the change to each pixel.

- L_2 -norm:

$$\|x' - x\|_2 = \sqrt{\sum_{i=1}^n (x'_i - x_i)^2}$$

measures the Euclidean distance between x' and x .

- L_∞ -norm or the Chebyshev distance: $\|x' - x\|_\infty = \max(|x'_1 - x_1|, |x'_2 - x_2|, \dots, |x'_n - x_n|)$
measures the maximum absolute change to any of the coordinates of x .

Definition 1 is recognized as “targeted adversarial attacks.” The “targeted” attacks comprise a target class t and a function α aims at identifying a legitimate input data x' (called adversarial example) such that the classifier f misclassifies x' as an instance of class t . In some scenarios, an attacker may not have a target class on her/his mind. Their goal is simply to mislead the classifier f to misclassify an adversarial example x' to a class other than its original class y .

This section mainly focuses on white-box attacks, assuming that an attacker has complete knowledge about the trained ML model. When attackers have access to this essential information, they can modify any arbitrary clean input using adversarial attack models. White-box attacks are the worst-case scenario because attackers have access to a model architecture and model weights.

2.2.1.1 L-BGFS Attack

Adversarial examples are often generated based on either first-order gradient information (such as the FGSM and DeepFool attack [101]) or gradient approximations. We first discuss some classical gradient-based attacks. In this case, the goal of an attacker is either to minimize the norm of the added perturbation that is needed to cause misclassification or to maximize the loss function with respect to an input data. In 2013, Szegedy et al. [133] demonstrate the first targeted adversarial attack on deep neural networks. They find a minimal distorted adversarial example x' by solving the following optimization problem:

$$\min \|\mathbf{x} - \mathbf{x}'\|_p, \quad (2.7)$$

subject to the constraints, $C(x') = t$ and $x' \in [0, 1]^n$, where $\| * \|_p$ is the L_p -norm. Finding a precise solution to this problem is very challenging since the constraint $C(x') = t$ is non-linear. Szegedy et al. replaced the constraint $C(x') = t$ with its associated continuous loss function L , such as the cross-entropy for classification. That is, they solved the following optimization problem instead:

$$\min c\|\mathbf{x} - \mathbf{x}'\|_p + L(\mathbf{x}', t), \quad (2.8)$$

subject to the constraint $x' \in [0, 1]^n$, where constant $c > 0$. By first adaptively choosing different values of constant c and then iteratively solving the problem (2.8) using the L-BFGS algorithm [90] for each selected c , an attacker can obtain an adversarial example x' that has the minimal distortion to x and the $C(x') = t$. The L-BGFS attack is formally defined as follows.

Definition 2. (L-BFGS Attack by Szegedy et al. 2013): Let $x \in R^n$ be a legitimate input data that are correctly classified as class y by an ML classifier C . Given a loss function L , the L-BFGS attack generates an adversarial example $x' \in R^n$ by minimizing the following objective function:

$$c\|\mathbf{x} - \mathbf{x}'\|_p + L(\mathbf{x}', t), \quad (2.9)$$

subject to the constraint $x' \in [0, 1]^n$, where x' is misclassified as class $t \neq y$ by C .

2.2.1.2 Fast Gradient Sign Method

Unlike the L-BFGS attack, the fast gradient sign method (FGSM) proposed by Goodfellow et al. [52] focuses on finding an adversarial perturbation limited by L_∞ -norm efficiently rather than producing an optimal adversarial example. In the training of an ML model, a given loss function is minimized to find an optimal parameter set θ for a classifier C so that C classifies most training data correctly. In contrast, FGSM maximizes the loss function as it tries to make the classifier perform poorly on the adversarial example x' . Therefore, the FGSM perturbed image for an untargeted attack is constructed by solving the following maximization problem:

$$\max_{\mathbf{x}' \in [0,1]^n} L(x', y) \quad (2.10)$$

subject to the constraint $\|x' - x\|_\infty \leq \epsilon$, where y is the ground-truth label of the original image x , and ϵ is some application-specific maximum perturbation.

Applying the first-order Taylor series approximation, we have

$$L(x', y) = L(x + \delta, y) \approx L(x, y) + \nabla_x L(x, y)^T \delta \quad (2.11)$$

where $\delta = x' - x$ is an adversarial perturbation, and $\nabla_x L(x, y)^T$ refers to the gradient of the loss function L with respect to input x and it can be computed quickly by a back-propagation algorithm. The objective function in (2.10) can be rewritten as:

$$\min_{\mathbf{x}' \in [0,1]^n} -L(x, y) - \nabla_x L(x, y)^T \delta \quad (2.12)$$

subject to the constraint $\|\delta\|_\infty \leq \epsilon$, where we convert the maximization problem to the minimization problem by flipping the sign of the two components. Note that $L(x, y)$ and $\nabla_x L(x, y)^T$ are constant and already known because the attack is in the white-box attack setting.

Furthermore, since $0 \leq \|\delta\|_\infty \leq \epsilon$, we have

$$\delta = \epsilon \text{sign}(\nabla_x L(x, y)) \quad (2.13)$$

where sign function outputs the sign of its input value.

For the targeted attack setting, an attacker tries to minimize the loss function $L(x, t)$, where t is the target class. In this case, we can show that

$$\delta = -\epsilon \text{sign}(\nabla_x L(\mathbf{x}, t)).$$

Formally, we can define FGSM with L_∞ -norm bounded perturbation magnitude as follows.

Definition 3. (FGSM Adversarial Attack by Goodfellow et al. 2014 [52]): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . The FGSM with L_∞ -norm bounded perturbation magnitude generates an adversarial example $x' \in R^n$ by maximizing the loss function $L(x', y)$ subject to the constraint $\|x' - x\|_\infty < \epsilon$, where ϵ is an application-specific imperceptible perturbation adversary intent to inject. That is,

$$\mathbf{x}' = \begin{cases} \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}, y)), & \text{untargeted} \\ \mathbf{x} - \epsilon \text{sign}(\nabla_{\mathbf{x}} L(\mathbf{x}, t)), & \text{targeted on } t, \end{cases}$$

where $\text{sign}(\nabla_{\mathbf{x}} L(x, y))$ is the sign of the loss function.

Furthermore, we can easily generalize the attack to other L_p -norm attacks . For example, we can use the Cauchy-Schwarz inequality, a particular case of the Holder's inequality with $p = q = 2$, to find the lower bound of the objective function in problem (2.12) subject to the constraint $\|x' - x\|_2 < \epsilon$ and to obtain the perturbation for an untargeted FGSM for L_2 -norm-based constraint

$$\delta = \frac{\epsilon \nabla_{\mathbf{x}} L(\mathbf{x}, y)}{\|\nabla_{\mathbf{x}} L(\mathbf{x}, y)\|_2} \quad (2.14)$$

2.2.1.3 Basic Iterative Method

The basic iterative method (BIM) proposed by Kurakin et al. [81] is an iterative refinement of the FGSM. BIM uses an iterative linearization of the loss function rather than the one-shot linearization in FGSM. Furthermore, multiple smaller but fixed step sizes α are taken instead of a single larger step size ϵ in the direction of the gradient sign. For instance, in an untargeted attack with L_∞ -norm bounded perturbation magnitude, the initial point $x^{(0)}$ is set to the original instance x , and in each iteration, the perturbation $\delta^{(k)}$ and $x^{(k+1)}$ are calculated as follows:

$$\delta^{(k)} = \alpha + \text{sign}(\nabla_x L(x^{(k)}, y)) \quad (2.15)$$

$$x^{(k+1)} = \text{clip}_{x, \epsilon}\{x^{(k)} + \delta^{(k)}\} \quad (2.16)$$

where $\epsilon > 0$ denotes the size of the attack and $\text{clip}_{x, \epsilon}(x')$ is a projection operator that projects the value of x' onto the intersection of the box constraint of x (for instance, if x is an image, then a box constraint of x can be the set of integer values between 0 and 255) and the ϵ neighbor ball of x . The above procedure ensures that the produced adversarial examples are within ϵ bound of the input x . Using a reduced perturbation magnitude α limits the number of active attack pixels and thus, prevents a simple outlier detector from detecting the adversarial examples.

Definition 4. (BIM Adversarial Attack by Kurakin et al. 2016 [81]): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . The BIM attack generates an L_∞ -norm bounded adversarial example $x' \in R^n$ by iteratively maximizing the loss function $L(x', y)$, subject to the constraint $\|x' - x\|_\infty < \epsilon$. For iteration k , $x^{(k+1)}$ is calculated as equation (2.15) and (2.16) and the total number of iterations is chosen by the user.

Iterative Least-likely Class Method (ILLC) [81] is similar to BIM but use least likely class as targeted class to maximize the cross-entropy loss; Hence, it is a targeted attack algorithm. Another popular variant of BIM is the projected gradient descent (PGD) attack, which uses a uniformly random noise as an initialization instead of setting $x^{(0)} = x$. The random initialization is used to explore the input space. Later, Croce and Hein [34] improved PGD by adding a momentum term for a gradient update and utilizing the exploration vs. exploitation concept for optimization. They called this approach the auto projected gradient descent (Auto-PGD) and showed that Auto-PGD is more effective than PGD [34]. It is further improved

with AutoAttack [34], an ensemble of various attacks. Indeed, AutoAttack combines three white-box attacks with one black-box attack. The four attack components are two extensions of Auto-PGD attacks (one maximizes cross-entropy loss, and another maximizes the difference of logistic ratio loss) and two existing supporting attacks, a fast adaptive boundary (FAB) attack [33] and a square attack [6], respectively. It has been shown that using the ensemble can improve the attack effectiveness over multiple defense strategies [33]. An attack is considered successful for each test example if at least one of the four attack methods finds an adversarial example.

Furthermore, Auto-Attack can run entirely without predefined user inputs across benchmark datasets, models, and norms. Therefore, it can provide a reliable, quick, and parameter-free evaluation tool when a researcher develops adversarial defenses [34]. We will introduce fast adaptive boundary attacks in the following subsection.

2.2.1.4 Jacobian-based Saliency Map Attack (JSMA)

JSMA uses Jacobian matrix of a given image to find the input features of the image that made most significant changes to output [113]. Then, the value of this pixel is modified so that the likelihood of the target classification is higher. The process is repeated until the limited number of the pixel can be modified is reached or the algorithm has succeeded in generating the adversarial example that is misclassified as target class. Therefore, this is another targeted attack but has high computation cost.

2.2.1.5 DeepFool

The DeepFool [101] attack is an untargeted white-box attack. Like Szegedy et al. [133], Moosavi-Dezfooli et al. [101] studied the minimally distorted adversarial example problem. However, rather than finding the gradient of a loss function, DeepFool searches for the shortest distance from the original input to the nearest decision boundary using an iterative linear approximation of the decision boundary/hyperplane and the orthogonal projection of the input onto the approximated decision boundary.

Moosavi-Dezfooli et al. searched the shortest distance path for an input x to cross the decision boundary and get misclassified. Formally, the DeepFool attack can be defined as follows.

Definition 5. (DeepFool Adversarial Attack by Moosavi-Dezfooli et al. 2016 [101]): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . The DeepFool

L_2 -norm attack generates adversarial example $x' \in R^n$ by solving the following optimization problem

$$\min_{x'} \|x' - x\|_2^2$$

subject to the constraint $x' \in g$, where g is the decision boundary that separating class c_i from class c_j where $j \neq i$.

To find such a path, an attacker first approximates the decision boundary using the iterative linear approximation method and then calculates the orthogonal vector from x to that linearized decision boundary. Furthermore, we present the solution to the binary classification case and extend the solution to the multi-classification case later.

Given a non-linear binary classifier with discriminant functions g_1 and g_2 , the decision boundary between class 1 and class 2 is $g = \{x : g_1(x) - g_2(x) = 0\}$. A per-iteration approximate solution x'_{i+1} can be derived by approximating the decision boundary/hyperplane g based on the first-order Taylor expansion:

$$g(x') = g(x_i) + \nabla_x g(x_i)^T (x' - x_i), \quad (2.17)$$

where x_i is the i^{th} iterate of the solution and $x_0 = x$. Then, we solve the following optimization problem:

$$\min_{x'} \|x'_{i+1} - x_i\|_2^2 \quad (2.18)$$

subject to $g(x_i) + \nabla_x g(x_i)^T (x' - x_i) = 0$ by finding the saddle node of the Lagrangian given by:

$$\frac{1}{2} \|x'_{i+1} - x_i\|_2^2 + \lambda (g(x_i) + \nabla_x g(x_i)^T (x' - x_i)). \quad (2.19)$$

The optimization problem can be solved to obtain the saddle node

$$x_{i+1} = x_i - \left(\frac{g(x_i)}{\|\nabla_x g(x_i)^T\|_2} \right) \frac{\nabla_x g(x_i)}{\|\nabla_x g(x_i)\|_2} \quad (2.20)$$

This saddle node can be considered as the orthogonal projection of x_i onto the decision hyperplane g at iteration i , and the iteration stops when x_{i+1} is misclassified; i.e., $\text{sign}(g(x_{i+1})) \neq \text{sign}(g(x_0))$.

This untargeted attack generates adversarial examples with smaller perturbation compared to L-BFGS, FGSM, and JSMA [20], [155]. For detail, see [101]. Universal Adversarial Perturbation is an updated

version of DeepFool that is transferable [102]. It uses DeepFool method to generate minimal perturbation for each image and find the universal perturbation that satisfies the following two constraints:

$$\|\eta\|_p \leq \epsilon,$$

$$P(F(\mathbf{x}) \neq F(\mathbf{x} + \eta)) > 1 - \delta,$$

where $\|\cdot\|_p$ is p-norm, ϵ specifies the upper limit for perturbation, and $\delta \in [0, 1]$ is a small constant that specifies fooling rate of all the adversarial images.

The multi-class (in one-vs-all scheme) approach of the DeepFool attack follows the iterative linearization procedure in the binary case, except that an attacker needs to determine the closest decision boundary l to the input x . The iterative linear approximation method used to find the minimum perturbation is a greedy method that may not guarantee the global minimum. Moreover, the closest distance from the original input to the nearest decision boundary may not be equivalent to the minimum difference observed by human eyes. In practice, DeepFool usually generates small unnoticeable perturbations. So far, we have only considered untargeted DeepFool attacks. A targeted version of DeepFool can be achieved if the input x is pushed towards the boundary of a target class $t \neq y$ subject to $C(x'_{i+1}) = t$. Furthermore, the DeepFool attack can also be adapted to find the minimal distorted perturbation for any L_p -norm, where $p \in [1, \infty]$. If interested, see [101] for details.

2.2.1.6 Fast Adaptive Boundary Attack

The fast adaptive boundary attack, proposed by Croce and Hein [33], is an extension of the DeepFool attack [101]. Specifically, the computation of x'_i for a FAB attack follows the DeepFool algorithm closely. The main differences between the two types of attacks are that a DeepFool attacker projects x'_i onto the decision hyperplane g only, $\text{Proj}_g(x'_i)$, whereas a FAB attacker projects x'_i onto the intersection of $[0, 1]^n$ and the approximated decision hyperplane g , $\text{Proj}_{g \cap [0, 1]^n}(x'_i)$, with the following additional steps.

1. Add a momentum term α to regulate the influence of the additional term $x + \eta \text{Proj}_{g \cap [0, 1]^n}(x)$ on the modified image $x'_i + \eta \text{Proj}_{g \cap [0, 1]^n}(x'_i)$. Thus, x'_i is biased toward x and hence, the x'_i generated by the FAB attack method is closer to the input x than one generated by the DeepFool attack. The momentum term α is updated for each iteration i as follows. First, a FAB attacker computes the minimal perturbations $\delta_i = \text{Proj}_{g \cap [0, 1]^n}(x'_i)$ and $\delta_0 = \text{Proj}_{g \cap [0, 1]^n}(x_0)$

as the current best point x'_i and the input x are projected onto the intersection of $[0, 1]^n$ and the approximated linear decision boundary, respectively. Then, α is chosen as follows.

$$\alpha = \min \left\{ \frac{\text{Proj}_{g \cap [0,1]^n}(x'_i)}{\text{Proj}_{g \cap [0,1]^n}(x'_i) + \text{Proj}_{g \cap [0,1]^n}(x)}, \alpha_{max} \right\} \in [0, 1],$$

where $\alpha_{max} \in [0, 1]$ is a hyperparameter that provides an upper limit value for α . α is defined such a way that the per-iteration x'_i might go too further away from x .

2. Add a backward step at the end of each iteration to increase the quality of the adversarial example even further. The DeepFool stops when x'_i is successfully misclassified. However, the FAB attack has this additional step for enhancement. The backward step is simply a movement of x'_i closer to the input x by calculating the linear combination of x and x'_i with a hyperparameter $\beta \in (0, 1)$. Croce and Hein [33] used $\beta = 0.9$ for all their experiments.
3. Add random restarts to widen the search space for adversarial examples. That is, instead of initializing $x'_0 = x$, a FAB attacker sets x'_0 as a random sample in a ϵ' -neighborhood of x , where $\epsilon' = \min\{\|x'_i - x\|_p, \epsilon\}$.
4. Add a final search step to further reduce the distance between the adversarial example and its original input. This step uses a modified binary search on x'_i and x to find a better adversarial example within a few iterations; e.g., Croce and Hein [33] set the number of iterations to 3 for their experiments. For details of the final search step and the random restarts, see [33].

2.2.1.7 Carlini and Wagner (C&W)'s Attack

Carlini and Wagner's (C&W) attack [20] aims to find the minimally disturbed perturbation, where Carlini and Wagner solved the box-constrained optimization problem in (2.7). Specifically, they converted the box-constrained optimization problem into an unconstrained optimization problem, which can then be solved through standard optimization algorithms instead of the L-BFGS algorithm. Carlini and Wagner investigated three different methods to get rid of the box-constraint $x' \in [0, 1]^n$, projected gradient descent, clipped gradient descent, and change of variables. They concluded that the method of the change of variables is most effective in generating adversarial examples fast. That is, they introduced a set of new variables

$w_i \in R, i = 1, 2, \dots, n$ such that

$$x'_i = \frac{1}{2}(\tanh(w_i) + 1). \quad (2.21)$$

Although $x'_i \in [0, 1], w_i$ can be any real number so that the box-constraint is eliminated. This allows to use other optimization algorithms that don't natively support box constraints. In their paper, the Adam optimizer is used since it converges faster than the standard gradient descent and gradient descent with momentum. For detail, see [20].

Furthermore, the constraint $C(x') = t$ is highly non-linear. Carlini and Wagner [20] considered seven objective candidate functions g to replace the constraint $C(x') = t$. Each g satisfies the condition that $C(x') = t$ if and only if $g(x') \leq 0$ (for targeted attacks) and $C(x') \neq y$ if and only if $g(x') \leq 0$ (for untargeted attacks). For instance, C&W's L_2 -norm attack has g defined as follows.

$$g(x') = \begin{cases} \max \left\{ \max_{j \neq t} Z_j(x') - Z_t(x'), -\kappa \right\}, & \text{targeted on } t \\ \max \left\{ Z_y(x') - \max_{j \neq y} Z_j(x'), -\kappa \right\}, & \text{untargeted,} \end{cases} \quad (2.22)$$

where Z is the logits and Z_j is the j^{th} element of the logits; $\kappa \geq 0$ is a parameter that controls the strength of adversarial examples. The default value is zero for experiments in [20]. However, increasing κ can generate adversarial examples with a higher transfer success rate.

Different from equation 2.18, the following optimization problem is solved to find the minimal perturbation $\delta = x' - x$:

$$\min \|x' - x\|_2^2 + cg(x') \quad (2.23)$$

subject to $x' \in [0, 1]^n$, where $c > 0$ is a regularization parameter that controls the relative importance of the L_p -norm perturbation over g . If a large value of c is chosen, the attacker generates an adversarial example which is further away from the base example but misclassified with high confidence and vice versa. Carlini and Wagner [20] used a modified binary search to find the smallest value of c for which the solution x' to the optimization problem (2.23) satisfies the condition $g(x') \leq 0$. Alternative to the modified binary search, a line search can also be used to find the adversarial example with the minimal L_p distance from x . Instead of looking for optimal c as in L-BFGS attack, Carlini and Wagner observed that the best way to choose c is to use the smallest value of c for which $g(x') \leq 0$.

Furthermore, Carlini and Wagner considered three types of attacks, L_0 -norm attack, L_2 -norm, and L_∞ -norm attacks. The formal definition of the C&W attack is presented as follows.

Definition 6. (C&W Adversarial Attack by Carlini et al. 2017 [19]): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . Given a target class t such that $t \neq y$. The C&W attack generates an adversarial example $x' \in R^n$ by solving the optimization problem (2.23).

Carlini and Wagner [19] showed that the C&W attack is very powerful, and ten proposed defense strategies cannot withstand C&W attacks constructed by minimizing defense-specific loss functions. This shows the current limitations of detection methods. Furthermore, the C&W attack can also be used to evaluate the efficacy of potential defense strategies since it is one of the strongest adversarial attacks [19]. Though C&W’s attack is very effective, it is computationally expensive compare to other techniques. In the next two subsections, we will discuss some adversarial attacks other than L_p -norm-based attacks.

2.2.1.8 Shadow Attack

In this subsection, we introduce a class of adversarial attacks called “semantic” adversarial attacks on image datasets that usually generate adversarial examples through a larger perturbation on an image in terms of L_p -norm but are still semantically similar to its original image. A typical example of the attack is simply a small color shift, rotation, shearing, scaling, or translation of a natural image to fool a state-of-the-art image classifier [42, 43]. The attack is successful due to differences in the way the ML model recognizes an image and the human visual recognition system. A semantic adversarial example can be formally defined as follows.

Definition 7. (Semantic Adversarial Attack): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . A semantic adversarial attack is a mapping $\alpha : R^n \rightarrow R^n$ such that the adversarial example $\alpha(x) = x'$ is misclassified as class $t \neq y$ by the classifier and $\beta(x) = \beta(x')$, where β is the human visual recognition system.

The semantic adversarial attacks can be either white-box or black-box. While we discuss one of the white-box semantic adversarial attacks in detail here, next subsection will be concerned with a spatial transformation attack, a box-box semantic adversarial attack.

The shadow attack [48] is a type of “semantic” adversarial examples that is different from the attacks introduced in subsections 2.2.1.1-2.2.1.7 as it targets not only the classifier’s output label but also the certifiable defenses [54, 72]. This attack reveals that certifiable defenses are not inherently secure. Furthermore, shadow attacks can also be considered as a generalization of PGD attacks with three penalty terms that minimize the perceptibility of perturbations. Formally, shadow attacks are defined as follows, with three penalty terms added to the loss term.

Definition 8. (Shadow Attack by Ghiasi et al. 2020 [48]): The shadow attack generates the adversarial example by solving the following optimization problem:

$$\max_{\delta} L(x + \delta, \theta) - \lambda_{tv} \text{TV}(\delta) - \lambda_c \mathbf{C}(\delta) - \lambda_s \mathbf{D}(\delta),$$

where θ is the model weights, x is an arbitrary natural image, δ is the perturbation added to image x , L is a loss function, $\lambda_c > 0$, $\lambda_{tv} > 0$, and $\lambda_s > 0$ are scalar penalty weights, $\text{TV}(\delta)$ is the total variation of δ , $\mathbf{C}(\delta) = \|\text{Avg}(|\delta_R|)\|_p + \|\text{Avg}(|\delta_B|)\|_p + \|\text{Avg}(|\delta_G|)\|_p$ restricts the change in the mean of each color channel, and $\mathbf{D}(\delta) = \|\delta_R - \delta_B\|_p + \|\delta_R - \delta_G\|_p + \|\delta_G - \delta_B\|_p$ promotes even perturbation between the color channels.

Note that the total variation $\text{TV}(\delta)$ is calculated element-wise as $\text{TV}(\delta_{i,j}) = \text{anisotropic} - \text{TV}(\delta_{i,j})^2$, where $\text{anisotropic} - \text{TV}(\delta_{i,j})^2$ estimates the L_1 -norm of the gradient of $\delta_{i,j}$.

2.2.1.9 Wasserstein Attack

Different from most adversarial attacks that focus on L_p -norm, the Wasserstein attack uses the Wasserstein distance (also known as the optimal transport, Kantorovich distance, or Earth mover’s distance) to generate Wasserstein adversarial examples. L_p -norms, though used extensively in adversarial attacks, do not capture the image transforms such as rotation, translation, distortion, flipping, and reflection.

Definition 9. (The Wasserstein distance between two inputs x and y): Let $x \in R_+^n$ and $y \in R_+^n$ be two inputs that are normalized with $\sum_{i=1}^n x_i = 1$ and $\sum_{j=1}^n x_j = 1$. Furthermore, let $C \in R^{n \times n}$ be a cost matrix such that each element $C_{i,j}$ of C measures the cost of moving mass from x_i to y_j (i.e., the Euclidean distance between the pixel x_i and the pixel y_j , if x and y are images). Then, the Wasserstein

distance between two inputs x and y is:

$$d_W(x, y) = \langle T, C \rangle,$$

subject to $T1 = x, T^T 1 = y$, where $1 \in R^{n \times 1}$ is a column vector of ones, $T \in R^{n \times n}$ is a transport plan whose element $T_{i,j} > 0$ encodes how the mass moves from x_i to y_j and $\langle T, C \rangle$ denotes the sum of the matrix-matrix element-wise multiplication. That is, $\langle T, C \rangle = \sum_{i=1}^n \sum_{j=1}^n T_{ij} * C_{ij}$.

Probabilistically, you can think of a transport plan as a joint probability of x and y [115]. Although the Wasserstein distance has many theoretical advantages, calculating a Wasserstein distance between two images is computationally expensive as it requires solving a linear programming problem with $n \times n$ number of variables. Wong et al. [148] developed a fast approximate projection algorithm, called the projected Sinkhorn iteration, that modifies the standard PGD attack by projecting onto a Wasserstein ball. Formally, Wong et al. [148] proposed to solve the following objective function as follows.

Definition 10. (The Wasserstein adversarial example by Wong et al. 2019 [148]): The Wasserstein attack generates a Wasserstein adversarial example $x \in R^n$ by minimizing the following objective function:

$$\|x' - x\|_2^2 + \frac{1}{\lambda} \sum_{ij} T_{ij} \log(T_{ij}),$$

subject to the Wasserstein ball constraints, $T1 = x, T^T 1 = x'$, and $d_W(x, y) < \epsilon$, where $\lambda \in R$.

Note that an entropy-regularized term, $\sum_{ij} T_{ij} \log(T_{ij})$, on the transport plan T , is included in the objective function so that projection is approximated; however, this entropy-regularized term speeds up the computation (near-linear time). To further reduce the computational cost, the local transport plan restricts the movement of each pixel to be within the $k \times k$ region of its original location. For instance, Wong et al. [148] set $k = 5$ for all their experiments.

2.2.2 Black-box Adversarial Attacks

Recent research on the adversarial attack has shown particular interest in black-box settings. Although many attacks require the attacker to have full access to the network architecture and weight, this information is often protected in commercialized products such as the vision system in modern self-driving

vehicles. As a result, the black-box attack, in which an attacker has no knowledge of the network and the training set except the ability to query the probability/confidence score for each class (score-based black-box attack) or the label (decision-based black-box attack), is increasingly important to investigate. Furthermore, since black-box attack models do not require the knowledge of gradients as in most white-box attacks introduced in section 2.2.1, the defense strategies based on the masking gradient or non-differentiability do not work with the attacks introduced in this section.

In section 2.2.2.1, we introduce a transfer attack, a popular black box attack that utilizes the transferability of adversarial examples. This attack can even be applied to no-box settings, in which an attacker does not even have access to the output of a model. In section 2.2.2.2, we present two score-based black-box attacks, zeroth order optimization (ZOO) based attack and square attack. Then, we discuss three popular decision-based black-box attacks in section 2.2.2.3.

2.2.2.1 Transfer Attack

Szegedy et al. [52] highlight the transferability of adversarial examples through the generalization power of DNNs. That is, the same adversarial image can be misclassified by a variety of classifiers with different architectures or trained on different training data. This property is useful in black-box settings. When an attacker does not have knowledge of model architecture, they can generate a substitute model to imitate the target model and then apply white-box attack methods such as L-BGFS attack to the obtained substitute model in order to generate an adversarial example. This type of attack is called a transfer attack. To do a transfer attack, an attacker needs to have a labeled training set first. With free query access, an attacker can feed the substitute training set to the target classifier in order to obtain the label for these instances. In the extreme case of limited query access, an attacker may have to do the labeling on their own or find a substitute training set that is already labeled. However, with the power of free query access, the generated model is usually more representative of the target model. With a labeled substitute training set, an attacker can select a substitute model that has a similar structure as the target model if they have any knowledge of the underlying structure. For instance, an image classifier usually has multiple CNN layers, whereas a sequential model may have some sort of RNN layers. Then, they can train the selected substitute model with the substitute training set. This trained substitute model serves as an attack surrogate. Finally, white box attack methods introduced in section 2.2.1 can be used to generate adversarial examples. In order to have a high transfer attack success rate, an FGSM, PGD, C&W, or other attack methods with high transferability

are selected to generate adversarial examples. Furthermore, the adversarial perturbation required is usually somewhat larger in order to ensure the success of a transfer attack.

2.2.2.2 Score-based Attacks

In this section, we discuss two popular score-based attacks, ZOO and square attacks. The ZOO attack is to find adversarial examples based on zero-order-optimization, whereas the square attack generates adversarial examples by using derivative-free optimization (DFO). Below are the detailed discussions of the two attacks.

Different from the transfer attacks exploiting the transferability of adversarial images, Chen et al. [24] proposed a ZOO attack to directly approximate the gradients of the target model using confidence scores. Therefore, the ZOO attack is considered a score-based black-box attack, and it does not need to train a substitute model. The ZOO attack is as effective as the C&W attack and remarkably surpasses existing transfer attacks in terms of a success rate. Chen et al. [24] also proposes a general framework utilizing capable gradient-based white-box attacks for generating adversarial examples in the black-box setting.

The ZOO attack finds the adversarial example by also solving the optimization problem (2.23). Motivated by the attack loss functions (2.22) used in the C&W attack, a new hinge-like loss function [24] based on the log probability score vector $p = f(x')$ of the model f , instead of Z , is proposed as follows.

$$g(x') = \begin{cases} \max \left\{ \max_{j \neq t} [f(x')]_j - [f(x')]_t, -\kappa \right\}, & \text{targeted on } t \\ \max \left\{ [f(x')]_y - \max_{j \neq y} [f(x')]_j, -\kappa \right\}, & \text{untargeted,} \end{cases} \quad (2.24)$$

where $[f(x')]_j$ is the j^{th} element of the probability score vector, and the parameter $\kappa \geq 0$ ensures a constant gap between the log probability score of the adversarial example classified as class t and all remaining classes. The log probability score is used instead of a probability score since well-trained DNNs yield a significant high confidence score for a class compared to other classes and the log function lessens this dominance effect without affecting the order of confidence score. The ZOO attack is defined as follows.

Definition 11. (ZOO Adversarial Attack by Chen et al. 2017 [24]): A ZOO attack is a score-based black-box attack that generates the adversarial example $x \in R^n$ by solving the optimization problem (2.23) using the zeroth-order stochastic coordinate descent with a coordinate-wise ADAM.

In the white box setting, finding an adversarial example requires taking the gradient of the model function $\partial f(x)$. However, in the black-box setting, the gradient is inadmissible, and one can only use the function evaluation $f(x)$, which makes it a zeroth-order optimization problem. Chen et al. [24] provide a method for estimating the gradient information surrounding the victim sample x by watching variations in prediction confidence $f(x)$ when the coordinate values of x are adjusted [149]. Chen et al. [24] used the following symmetric difference quotient to approximate the gradient:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he_i) - f(x - he_i)}{2h}, \quad (2.25)$$

where h is a small constant, and e_i is a standard basis vector. For networks with a large input size n , e.g., Inception-v3 network [134] has $n = 299 \times 299 \times 3$, the number of model queries per gradient evaluate is $2 \times n$ (two function evaluation per coordinate-wise gradient estimation). This is very query inefficient.

To overcome this inefficiency, Chen et al. [24] proposed the five acceleration techniques.

1. Use zeroth-order stochastic coordinate descent with a coordinate-wise ADAM to minimize the objective function.
2. Evaluate the objective function in batches instead of one-by-one to take advantage of the GPU.
3. Reduce the attack space and perform the gradient estimation from a dimension reduced space R^m , where $m < n$, to improve computational efficiency. Specifically, a dimension reduction transformation $D : R^m \rightarrow R^n$ such as $D(\delta') = x' - x$, where $\delta' \in R^m$ is used to reduce the attack space. For instance, D can be an up-scale operator that scales up an image, such as the bilinear interpolation method, discrete cosine transformation (DCT), or an autoencoder for attack dimension reduction [143]. Although this attack space dimension reduction method reduces the computation costs for high-resolution image datasets (such as ImageNet [122]), the reduced attack space also limits the ability to find a valid attack.
4. Use hierarchical attack to overcome the limited search space issue. Instead of just one transformation, a series of transformations D_1, D_2, \dots, D_v with dimensions m_1, m_2, \dots, m_v , where $m_1 < m_2 < \dots < m_v$ are used to find the adversarial example. Start with the small attack space m_1 , an attacker tries to find an adversarial example. If they are unsuccessful, then they increase the attack space to m_2 . Otherwise, the process stops.

5. Choose which coordinates to update based on their “importance.” For instance, the pixel near the edge or corner of an image is less important than a pixel near the main object (usually near center) for an image classifier. Chen et al. [24] divided an image into 8×8 regions and defined the “importance” as the absolute values of pixel value changes in each region.

Note that acceleration techniques 3-5 are not required when n is small. For instance, Chen et al. [24] did not use techniques 3-5 for MNIST and CIFAR10 dataset. Although the attack success rate is compared to the success rate of the C&W attack, the number of required queries are large for gradient estimation despite the proposed acceleration techniques.

Andriushchenko et al. [6] proposed square attack (SA), a query-efficient attack on both L_∞ and L_2 -bounded adversarial perturbations. The SA is based on a random search (RS), an iterative technique in DFO methods. Therefore, the SA is a gradient-free attack [75] and it is resistant to gradient masking. SA improves the query efficiency and success rate by employing RS and a task-specific sampling distribution. In some cases, SA even competes with white-box attacks’ performance.

Compared to an untargeted attack, SA is to find a L_p -norm bounded adversarial example by solving the following box-constrained optimization problem:

$$\min_{x'} L(f(x', y)) \quad (2.26)$$

subject to $\|x' - x\|_p \leq \epsilon$, where $p \in \{2, \infty\}$, $x' \in [0, 1]^n$, and $L(f(x', y)) = [f(x')]_y - \max_{j \neq y} [f(x')]_j$ is a margin-based loss function measuring the level of confidence that the model f labels an adversarial input x' the ground-truth class y over other classes.

In a targeted attack with target class t , an attacker could simply minimize the loss function

$$L(f(x', t)) = \max_{j \neq t} [f(x')]_j - [f(x')]_t. \quad (2.27)$$

However, a cross-entropy loss of the target class t

$$L(f(x', t)) = -[f(x')]_t + \log \left(\sum_{i=1}^K e^{[f(x')]_i} \right). \quad (2.28)$$

is used to make the targeted attack more query efficient.

This optimization problem can be solved by using the classical RS method. In general, the RS method can be described as follows. Given an objective function g to minimize, a starting point x_0 , and a sampling distribution D , the RS algorithm outputs an estimated solution x_{out} after N iterations. For each iteration i , RS algorithm starts with a random update $\delta \sim D(x_i)$ and then adds this update to the current iteration; i.e., $x_{new} = x_i + \delta$. If the objective function g_{new} evaluated at x_{new} is less than the best g^* obtained so far, we update x_i and g^* . That is, if $g_{new} < g^*$, then $x_i = x_{new}$ and $g^* = g_{new}$. Otherwise, the update δ is discarded [35]. This process stops after a user specified number of iterations has reached. Thus, using RS to optimize g does not rely on the gradient of g .

Since Moon et al. [100] showed that when one runs the white-box PGD attacks until convergence on CIFAR10 and ImageNet dataset, the successful adversarial perturbation δ is mainly found on vertices (corners) of L_∞ -norm ball [6]. Based on this observation, the SA attack only searches over the boundaries of the L_∞ or L_2 -ball to obtain the potential adversarial perturbation of the current iterate δ_i . Hence, the perturbation for each pixel (before projection) is either $-2\epsilon, 0$, or 2ϵ for the L_∞ -norm attack, where the perturbation of zero means no perturbation for a given pixel. This critical design makes the SA mechanism different from the standard RS algorithm, and it significantly improves the query efficiency.

Another key difference is that at each step the SA attack limits the modification of the current iterate x'_i by updating only a small fraction (a square of side length v_i for each color channel) of the neighboring pixels of x'_i . This helps reduce the dimension of the search space, especially when the input space $[0, 1]^n$ is high-dimensional, e.g., $n = 290 \times 290 \times 3$ in the ImageNet dataset [122]. In summary, the SA via random search consists of the following main steps for each iteration i [35]:

1. Find the side length v_i of the square by determining the closest positive integer to $\sqrt{pw^2}$, where $p \in [0, 1]$ is the percentage of pixels of the original image x that can be perturbed and w is the width of an image. p gradually decreases with the iterations but $v_i \geq 3$ for the L_2 -norm SA. This mimics the step size reduction in gradient-based optimization. In gradient-based optimization, we begin with initial large learning rates to quickly shrink the loss value [35].
2. Find the location of the square with side length v_i for each color channel by uniformly random selection. The square denotes the set of pixels that can be modified.
3. Uniformly assign all pixels' values in the square to either -2ϵ or 2ϵ for each color channel c .

4. Add the square perturbation generated in step 3 to the current iterate x_i to obtain the new point x_{i+1} .
5. Project x_{i+1} onto the intersection of $[0, 1]^n$ and the L_∞ -norm ball of radius ϵ to obtain x_{new} .
6. If the new point x_{new} attains a lower loss than the best loss so far, the change is accepted, and the best loss is updated. Otherwise, it is discarded.

The iteration continues until the adversarial example is found. If interested, see [6] for details.

2.2.2.3 Decision-based Attack

The decision-based attack uses only the label of an ML output for generating adversarial attacks, and this makes it easily applicable to real-world ML attacks. In this section, we discuss three popular decision-based black-box attacks, a boundary attack [17], a HopSkipJump attack [23], and a spatial-transformation attack [43].

Relying neither on training data nor on the assumption of transferability, the boundary attack uses a simple rejection sampling algorithm with a constrained independent and identically distributed Gaussian distribution as a proposed distribution and a dynamic step-size adjustment inspired by Trust Region methods to generate minimal perturbation adversarial samples. The boundary attack algorithm is given as follows. First, a data point is sampled randomly from either a maximum entropy distribution (for an untargeted attack) or a set of data points belonging to the target class (for a targeted attack). The selected data point serves as a starting point. At each step of the algorithm, a random perturbation is drawn from a proposed distribution such that the perturbed data still lies within the input domain and the difference between the perturbed image and the original input is within the specified maximum allowable perturbation ϵ . Newly perturbed data is used as a new starting point if it is misclassified for an untargeted attack (or misclassified as the target class for a targeted attack). The process continues until the maximum number of steps is reached.

The boundary attack is conceptually simple, requires little hyperparameter tuning, and performs as well as the state-of-the-art gradient attacks (such as the C&W attack) in both targeted and untargeted computer vision scenarios without algorithm knowledge [17]. Furthermore, it is robust against common deceptions such as gradient obfuscation or masking, intrinsic stochasticity, or adversarial training. However, the boundary attack has two main drawbacks. First, the number of queries for generating an adversarial

sample is large, making it impractical for real-world applications [23]. Instead of a rejection sampling algorithm, Metropolis-Hastings’s sampling may be a better option since it does not simply discard the rejected sample. Secondly, it only considers L_2 -norm.

Conversely, the HopSkipJump attack is a family of query-efficient algorithms that generates both targeted and untargeted adversarial examples for both L_2 and L_∞ -norm distances. Furthermore, the HopSkipJump attack is more query efficient than the Boundary attack [152], and it is a hyperparameter-free iterative algorithm. The HopSkipJump attack is defined as follows.

Definition 12. (HopSkipJump Attack by Chen et al. 2020 [23]): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . A HopSkipJump attack is a decision-based attack that generates the adversarial example x' by solving the following optimization problem:

$$\min_{x'} \|x' - x\|_p$$

subject to the constraint $\phi_x(x') = 1$, where $p = \{2, \infty\}$

$$\phi_x(x') = \text{sign}(S_x(x')) = \begin{cases} 1 & S_x(x') > 1 \\ -1 & \text{otherwise,} \end{cases}$$

and

$$S_x(x') = \begin{cases} [f(x')]_t - \max_{j \neq t} [f(x')]_j & \text{targeted on } t \\ \max_{j \neq y} [f(x')]_j - [f(x')]_y & \text{untargeted} \end{cases}$$

Note that S_{x^*} is similar to (2.22) with $\kappa = 0$. Let us discuss HopSkipJump L_2 based target attack in detail. Interested readers can consult [23] for untargeted attack or L_2 based attacks.

The HopSkipJump L_2 based target attack is an iterative algorithm consisting mainly an initialization step (step 1) and three iterative steps (steps 2-4 are repeated until the maximum number of iterations specified by an attacker is reached.):

1. Select a random data point \tilde{x}_0 from a target class (similar to the initialization step of a boundary attack).

- Approach the decision boundary/hyperplane by using a binary search algorithm. That is, move \tilde{x}_t towards the input x and the decision hyperplane by interpolating between the input x and \tilde{x}_t to get a new data point x_t such that $\phi_x(x_t) = 1$; i.e.,

$$x_t = \alpha_t x + (1 - \alpha_t) \tilde{x}_t,$$

where α_t is obtained via a binary search between 0 and 1 and stops at x_t such that $\phi_x(x_t) = 1$.

- Estimate the gradient-direction using the method similar to FGSM, see (2.14). That is,

$$v_t = \frac{\widehat{\nabla S}(x_t, \delta_t, B - t)}{\|\widehat{\nabla S}(x_t, \delta_t, B_t)\|}$$

where $\delta_t = \frac{\|x_t - x\|_2}{n}$ depends on the distance between x_t and x and the image size n , $\{u_b\}_{b=1}^{B_t}$ is a set of independent and identically distributed uniform random noise, $B_t = B_0 \sqrt{t}$ is the batch size of $\{u_b\}_{b=1}^{B_t}$, the initial batch size B_0 is set to 100 in [23], and

- Estimate the step size ξ_t by geometric progression and then update the sample point \tilde{x}_t . That is, starting with $\xi_t = \frac{\|x_t - x\|_2}{\sqrt{t}}$, the step size is reduced by half until $\phi_x(\tilde{x}_t) = 1$, where

$$\tilde{x}_t = x_t + \xi_t v_t.$$

Spatial Transformation Attack

In contrast to the boundary attack and the HopSkipJump attack that based on L_p bounded adversarial perturbations, the spatial transformation attack [43] is a black-box ‘‘Semantic’’ adversarial attack that reveals a small random transformation, such as a small rotation on an image, can fool a state-of-the-art image classifier easily. This attack really questions the robustness of these state-of-the-art image classifiers. Furthermore, the primary disadvantage of the ZOO attack is the need to probe the classifiers thousands of times before the adversarial examples are found. This is not a case for spatial transformation attacks. Engstrom et al. [43] showed that worst-of-10 ($k=10$) is able to reduce the model accuracy significant with just 10 queries. Basically, it rotates or translates a natural image slightly to cause misclassification. This attack reveals the vulnerability of the current state-of-the-art ML models.

Definition 13. (Spatial Transformation Attack): Let $x \in R^n$ be a legitimate input data that is correctly classified as class y by an ML classifier F . A spatial transformation attack generates an adversarial example $x' \in R^n$ by finding a mapping $\alpha : R^n \rightarrow R^n$ such that the adversarial example $\alpha(x) = x'$, where the relationship between x' and x is as follows:

Each pixel (a, b, c) in x is translated $(\delta a, \delta b, 0)$ pixels to the right and rotate γ degrees counterclockwise around the origin to obtain the pixel (a', b', c) in x' , where $a' = a \cos(\gamma) - b \sin(\gamma) + \delta a$ and $b' = a \sin(\gamma) + b \cos(\gamma) + \delta b$.

To find the mapping α , an attacker needs to solve the following optimization problem:

$$\max_{a,b,c} L(\alpha(x), y) \tag{2.29}$$

Engstrom et al. [43] proposed three different methods to solve (2.29): (1) first-order method to maximize the loss function, (2) grid search on a discretized parameter space, and (3) worst-of-k (i.e., randomly sample k different set of attack parameters and choose one that model performs worst). The first order method requires gradient of the loss function which is not possible in black box setting. Even if it is possible, Engstrom et al. [4] shows that the first-order method performs worst due to the non-concavity of loss function for spatial perturbation.

Please note that we introduce a few black-box attack methods in this section. There are many other black-box attack methods. Following are two of the black-box attack techniques proposed in past few years.

- One-Pixel Attack uses differential evolution to find a pixel to modify and generate adversarial example. Success rate is 68.36% for CIFAR-10 test dataset and 16.04% for ImageNet dataset on average [131]. It shows the vulnerability of DNNs as a single pixel alternation can change the classification.
- Natural GAN uses Wasserstein Generative Adversarial Networks (WGAN) to generate adversarial examples that look natural for tasks such as image classification, textual entailment, and machine translation [158]. Similarly, the Domain name GAN (DnGAN) [18] uses GAN to generate adversarial DGA examples and tests its performance on Bayes, J48, decision tree, and random forest classifier. Moreover, the authors [18] also used these generated adversarial examples along with normal training data to train a long short-term memory (LSTM) model and

show a 2.4% increase in accuracy compared to the LSTM model trained with a normal training set alone. As another application, Chen *et al.* [25] use WGAN to denoise cell images.

2.2.3 Data Poisoning Attacks

While adversarial attacks cannot change the training process of a model and can only modify the test instance, the data poisoning attacks, on the contrary, can manipulate the training process. Specifically, in data poisoning attacks, attackers aim to manipulate the training data (e.g., poisoning features, flipping labels, manipulating the model configuration settings, and altering the model weights) in order to influence the learning model. It is assumed that attackers have the capability to contribute to the training data or have control over the training data itself. The main objective of injecting poison data is to influence the model’s learning outcome.

Recent studies on adversarial ML have demonstrated particular interest in data poisoning attack settings. This section discusses few data poisoning attack models. We start with briefly going over label flipping attacks in subsection 2.2.3.1 and then focus on clean label data poisoning attacks in subsection 2.2.3.2 since they are stealthy. To the end, we introduce backdoor attacks.

2.2.3.1 Label Flipping Attacks

A simple and effective way to attack a training process is to simply flip labels of some training instances. This type of data poisoning attacks is called label flipping attacks. Mislabeling can be done easily in crowdsourcing, where an attacker is one of the annotators, for example. In this subsection, we discuss some common label flip attacks against ML models. These label flip attacks could be either model-independent or model-dependent. For instance, a random label flipping attack is model-independent. It simply selects a subset of training instances and flips their labels. The attacker does not need to have any knowledge of the underlying target ML model. Even though this random strategy looks simple, it is capable of reducing classification accuracy significantly depending on the type of dataset under attack, the training set size, and the portion of the training labels that are flipped. The random label flip attack can be mathematically defined as follows.

Definition 14. (Random Label Flipping Attack): Given a training set $D = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in X$ and $y_i \in \{-1, 1\}$, it is called a random label flipping attack if an attacker with the ability to

change $p \in (0, 1)$ fraction of the training label can randomly select $\lfloor np \rfloor$ training labels and flip the labels.

This random label flipping attack can further be divided into two groups: targeted and untargeted. In an untargeted random label flipping attack, an attacker may select some instances from class 1 to misclassify as class -1 and some instances from class -1 to misclassify as class 1. In contrast, an attacker misclassifies one class as another consistently in a targeted random label flipping attack. The targeted random label flipping attack is more severe compared to the untargeted one as the targeted attack consistently misleads the learning algorithm to classify a specific class of instances as another specific class.

Rather than random label flipping, an attacker can also utilize label flip attacks that are model-dependent. For instance, we show that SVM constructs a decision hyperplane using only the support vectors in subsection 2.1.1. Existing studies presented a few label flipping attacks based on this characteristic. For example, Farfirst attack [146] is one of such label flipping attacks, where a training instance far away from the margin of an SVM is flipped. This attack effectively changes many non-support vector training instances (training instances that are far from the margin and correctly classified by an untainted SVM) to support vectors, and significantly alter the decision boundary consequently.

Formally, an optimal label flipping attack can be considered as a bi-level optimization problem defined as follows.

Definition 15. (Optimal Label Flipping Attack): Suppose that training set $D = \{(x_i, y_i)\}_{i=1}^n$ and a test set $T = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$, where $x, \tilde{x} \in X$ and $y_i, \tilde{y}_i \in \{-1, 1\}$. Let l be the number of training labels that an attacker has the ability to modify, and let $I \in \{-1, 1\}^n$ be an indicator vector such that its i^{th} element I_i equals to -1 if the i^{th} training label is flipped and 1 otherwise. Then, $|I| = l$ and the poisoned training set $D' = \{x_i, y'_i\}_{i=1}^n$, where $y'_i = y_i I_i$. It is called an optimal label flipping attack if an attacker can find an optimal I by solving the following optimization problem:

$$O(\min_{\Theta} L(D', \Theta), T),$$

where O is an objective function specified by the attacker and Θ is model weights. A common objective function is to maximize the test error rate, in which case $O(\Theta^*, T) = \frac{1}{m} \sum_{i=1}^m J[C(\tilde{x}_i) = \tilde{y}_i]$, where J is an indicator function such that

$$J[C(\tilde{x}_i) = \tilde{y}_i] = \begin{cases} 1 & C(\tilde{x}_i) = \tilde{y}_i \\ 0 & otherwise. \end{cases}$$

However, solving such a bi-level optimization problem is NP-hard. Here, we introduce a simple greedy algorithm that is suboptimal but tractable. At iteration $t = 1$, an attacker first flips the label of the first training instance to obtain a poisoned training set D' . Then, the attacker trains an ML model using D' and evaluates the performance p_1 of the poisoned model. Next, the attacker flips the label of the second training instance while keeping other labels untainted. Similarly, the attacker trains the ML model using the poisoned dataset and evaluates its performance p_2 on the test set. The attacker repeats this process for each training label to obtain $\{p_1, p_2, \dots, p_n\}$. Now, the attacker can determine which training label is actually flipped by simply finding the training label corresponding to the worst performance. After that label is flipped, the attacker can select the next training label to flip out of the remaining $n - 1$ training labels by following the same process as iteration 1. For each iteration, one additional training label is flipped. The process stops until l training labels have been flipped.

2.2.3.2 Clean Label Data Poisoning Attack

Security of DL networks can be degraded by the emergence of clean label data poisoning attacks. This is achieved by injecting legitimately labeled poison samples into a training set. Although poison samples seem normal to a user, they indeed comprise illegitimate characteristics to trigger a targeted misclassification during the inference process.

In this subsection, an attacker deploys “clean-labels” data poisoning attacks without the need to control the labeling process. Furthermore, such adversarial attacks are often “targeted” such that they allow for controlling the classifier’s behavior on a particular test instance while maintaining the overall performance of the classifier [127]. For instance, an attacker can insert a seemingly legitimate image, i.e., a correctly labeled image, into a training dataset for a facial identification engine and thus control the chosen person’s identity during the test time. This attack is severe since the poison instance can be placed online and awaits to be scraped by a bot for data collection. The poison instance is then labeled correctly by a trusted annotator but still able to subvert the learning algorithm.

Here, we introduce two clean label data poisoning attacks against both transfer learning and an end-to-end training.

Feature Collision Attack

The first one is the feature collision attack introduced by Shafahi et al. [127] who assumed that an attacker has

- no knowledge of the training data,
- no ability to influence the labeling process,
- no ability to modify the target instance during inference, and
- knowledge of the feature extractor’s architecture and its parameters.

An attacker tries to find a poison instance that collides with a given target instance in a feature space while maintaining its indistinguishability with a base instance from class c other than target class $t \neq c$. Hence, the generated poison instance looks like a base instance and an annotator labels it as an instance from class c . However, that poison instance is close to the target instance in a feature space, and the ML model is likely classifying it as an instance from class t . This causes the targeted misclassification and only one such poison instance is needed to poison a transfer learning model. That is why this attack is sometimes called a one-shot attack as well, and its formal definition is shown as follows.

Definition 16. (Feature Collision Attack by Shafahi et al. 2018 [127]): Given a feature extractor f , a target instance x_t from class t , and a base instance x_b that belongs to a targeted class b such that $b \neq t$, it is called a feature collision attack if an attacker finds a poison instance x_p as follows:

$$x_p = \operatorname{argmin}_x \|f(x) - f(x_t)\|_2^2 + \beta \|x - x_t\|_2^2,$$

where β is a similarity parameter.

The base instance can be selected randomly from any classes other than the target class. However, some base instances may be easier for an attacker to find a poison instance than others. The coefficient β must be tuned by the attacker in order for them to make the poison instance seem indistinguishable from a base instance. Shafahi et al. [127] solved the optimization problem by using a forward-backward-splitting iterative procedure [49].

This attack has a remarkable attack success rate (e.g., 100% in one experiment presented in [127]) against transfer learning. Nevertheless, we want to point out that such an impressive attack success rate

reported in [127] is due to the overfitting of the victim model to the poison instance. The data poisoning attack success rate drops significantly on an end-to-end training and in black-box settings, and Shafahi et al. [127] proposed to use a watermark and multiple poison instances to increase the attack success rate on an end-to-end training. However, one obvious drawback is that the pattern of the target instance shows up in the poison instances sometimes.

Convex Polytope Attack and Bullseye Polytope Attack

Although in many ML-based attacks, an attacker is required to obtain full access to the network architecture and weight of a model (e.g., a feature collision attack), such a privilege is likely protected in commercialized products such as perception systems in smart cars. In this subsection, we introduce a transferable and scalable clean label targeted data poisoning attack against transfer learning and an end-to-end training in a black-box setting, where attackers do not have access to the target model. Aghakhani et al. [2] proposed such a clean label data poisoning attack model, named the bullseye polytope attack, to create poison instances that are similar to a given target instance in a feature space. The bullseye polytope attack achieves a higher attack success rate compared to a feature collision attack in a black-box setting and increases the speed and success rate of the targeted data poisoning attack on the end-to-end training compared to the convex polytope attack.

The convex polytope attack [159] is very similar to the bullseye polytope attack; both of them are the extension of a feature collision attack into a black-box setting. Let us define the convex polytope attack formally as follows before we discuss the bullseye polytope attack, a scalable alternative to the convex polytope attack.

Definition 17. (Convex Polytope Attack by Zhu et al. 2019 [159]): Given a feature extractor $\{f^i\}_{i=1}^m$, a target instance x_t from class t and a set of base instances $\{x_b^j\}_{j=1}^k$ that belongs to a targeted class b such that $b \neq t$, it is called a convex polytope attack if an attacker finds a set of poison instances $\{x_p^j\}_{j=1}^k$ by solving the following optimization problem:

$$\min_{\{x_p^j\}_{j=1}^k, \{c^i\}_{i=1}^m} \frac{1}{2} \sum_{i=1}^m \frac{\|f^i(c_t) - \sum_{j=1}^k c_j^i f^i(x_p^j)\|^2}{\|f^i(x_t)\|^2}, \quad (2.30)$$

subject to the constraints $\sum_{j=1}^k c_j^i = 1, c_j^i \geq 0, \forall i = 1, 2, \dots, m, \forall j = 1, 2, \dots, k$, and $\|x_p^j - x_b^j\|_\infty \leq \epsilon, \forall j = 1, 2, \dots, k$, where $c^i \in R^k$ is a coefficient vector consisting of elements c_j^i .

Without the knowledge of a victim model, an attacker can utilize the attack transferability to attack the victim model by substituting feature extractors $\{f^i\}_{i=1}^m$ (similar to transfer attacks in section 2.2.2.1). This is the main idea behind both the convex polytope attack and the bullseye polytope attack. The main difference between the two attacks is that there is no incentive to refine the target deeper inside the convex hull of poisons (“attack zone”) for the convex polytope attack, whereas the bullseye polytope attack pushes the target into the center of convex hull by solving the optimization problem (2.31) instead. Consequently, the target is often near the convex hull boundary in the convex polytope attack. The optimization problem (2.30) requires finding a set of coefficient vectors $\{c^i\}_{i=1}^m$ in addition to find a set of poisons $\{x_p^j\}_{j=1}^k$ that minimizes its objective function. Hence, the convex polytope attack has a much higher computation cost than that of the bullseye polytope attack. As shown below, the bullseye polytope attack simply sets $c_j^i = \frac{1}{k}, \forall i = 1, 2, \dots, m, \forall j = 1, 2, \dots, k$, to assure that the target remains near the center of the poison samples.

Definition 18. (Single-target Mode Bullseye Polytope Attack by Aghakhani et al. 2020 [2]): Given a feature extractor $\{f^i\}_{i=1}^m$, a target instance x_t from class t and a set of base instances $\{x_b^j\}_{j=1}^k$ that belongs to a targeted class b such that $b \neq t$, it is called a single-target mode bullseye polytope attack if an attacker finds a set of poison instances $\{x_p^j\}_{j=1}^k$ by solving the following optimization problem:

$$\min_{\{x_p^j\}_{j=1}^k} \frac{1}{2} \sum_{i=1}^m \frac{\|f^i(c_t) - \frac{1}{k} \sum_{j=1}^k f^i(x_p^j)\|^2}{\|f^i(x_t)\|^2}, \quad (2.31)$$

subject to the constraints $\|x_p^j - x_b^j\|_\infty \leq \epsilon, \forall j = 1, 2, \dots, k$.

The bullseye polytope attack can further be extended to a more sophisticated target mode (called the multi-target mode) by constructing a set of poison images that targets multiple instances enclosing the same object from different angles or lighting conditions. Specifically, if n_t target instances of the same object is used in a multi-target mode attack, $f^i(x_t)$ in (2.31) is replaced by the average target feature vectors $\sum_{v=1}^{n_t} f_v^i(x_t)$. Such a model extension can remarkably enhance the transferability of the attack to unseen instances comprising the same object without having to use additional poison samples [2].

2.2.3.3 Backdoor Attack

In this last subsection, we introduce backdoor attacks, a type of data poisoning attacks/causative attacks that contains a backdoor trigger (an adversary’s embedded pattern). Specifically, we consider backdoor attacks on an outsourcing scenario [58], a transfer learning scenario [58], and a federated learning scenario [8]. Here are the three attack scenarios:

- Outsourced training: In this attack scenario, a user aims to train the parameters Θ of a network f_{Θ} by using a training dataset. For this purpose, the user sends the model description to the trainer who will then return trained parameters Θ' . The user herein verifies the trained model’s parameters by checking the classification accuracy on a “held-out” validation dataset, D_{valid} , as the trainer cannot be fully trusted. The model will be accepted only if its accuracy fulfills a target accuracy, a^* , on the validation set. Building upon the above setting, a malicious trainer will return an illegitimate backdoored model $f_{\Theta_{adv}}$ to the user such that $f_{\Theta_{adv}}$ has met the target accuracy requirement while $f_{\Theta_{adv}}$ misclassifies the backdoored instances.
- Transfer learning: In this attack scenario, a user unintentionally downloads an illegitimately trained model $f_{\Theta_{adv}}$ and the corresponding training and validation set D_{valid} from an online repository. Then, the user develops a transfer learning model $f_{\Theta_{adv},TL}$ based on it. The main attacker’s goal is similar to his/her goal in the adversarial outsourced training scenario. However, both $f_{\Theta_{adv}}$ and $f_{\Theta_{adv},TL}$ must have the reasonably high accuracy on D_{valid} and the user’s private validation set for the new domain of application.
- Federated learning: This attack scenario is especially vulnerable to a backdoor attack since, by design, the central server has no access to the participants’ private local data and training. A malicious participant or a compromised participant can manipulate the joint model by providing a malicious interned update to the central server based on the techniques such as contrain-and-scale [8]. As a result, the joint model would behave in accordance with the attacker’s goal as long as the input encloses the backdoor features while the model can maintain the classification accuracy on the clean test instances.

An attacker can generate an illegitimate trained DL network, also known as a backdoored neural network (BadNet), for both targeted or untargeted misclassification on backdoor instances by modifying

the training procedure, such as injecting poisoned instances with backdoor trigger superimposed and label altered, manipulating the model configuration settings, or altering model parameters directly. Subsequently, the malicious learning network behaves wrongly on backdoor instances whereas has high classification accuracy on the user's private validation set.

Although various detection mechanisms have been elaborated on adversarial backdoors detection, e.g., using statistical differences in latent representations between clean input data and attacker's input samples in a poisoned model, we introduce an adversarial backdoor embedding attack [137] that can bypass several detection mechanisms (e.g., Feature Pruning [144] and Dataset Filtering by Spectral Signatures [141]) simultaneously in this section. Notably, Shokri et al. [137] developed an adversarial training mechanism, which creates a poisoned model that can mitigate either a specific defense or multiple defenses at the same time by utilizing an objective function that penalizing the difference in latent representations between the clean inputs and backdoored inputs.

The proposed mechanism maximizes the indistinguishability of both clean and poisoned data's hidden representations by utilizing the idea similar to a generative adversarial network. The poisoned model decouples the adversarial inputs comprising the backdoor triggers from clean data by developing a discriminative network. The conventional backdoor detection mechanisms may succeed in recognizing the backdoor triggers only if the attacker naively trains the poisoned model in a way that leaves a remarkable difference in the distribution of latent representations between the clean data and backdoor instances. Therefore, a knowledgeable attacker can make the model more robust against existing backdoor detection mechanisms by minimizing the difference in latent representations of two. As a result, the attacker can attain high accuracy of classification by the poisoned model (the behavior of the poisoned model remains unmodified on clean data), while fooling the backdoor detection mechanism.

2.2.4 Summary

The performance of deep learning is remarkable in a variety of domains such as image classification, natural language processing, and machine translation. With such outstanding performance, ML models are widely used today. However, recent studies have shown that machine learning algorithms can be easily fooled by techniques such as Fast Gradient Sign Method (FGSM) [52] and PGD attack [94]. This is a serious problem as many deep learning techniques are used in security-sensitive and/or safety-critical applications

such as medical diagnosis, malware detection [151], self-driving cars [97], as well as digital assistants like Google Assistant, Alexa and Siri.

Adversarial example/sample \mathbf{x}' is a generated sample that is close to natural sample \mathbf{x} to human eyes but misclassified by a DNN model [133]. That is, the modification is so subtle that a human observer does not even notice it, but a DNN model misclassifies it. Formally, an adversarial sample x' satisfies the conditions that

- $\|\mathbf{x} - \mathbf{x}'\|_2^2 < \epsilon$, where ϵ is the small perturbation that is not noticeable by human eyes.
- $\mathbf{x}' \in [0, 1]$.
- $F(\mathbf{x}') = y'$, $F(\mathbf{x}) = y$, and $y \neq y'$, where y and y' are class labels.

In general, there are two types of adversarial attacks for DNN models: untargeted and targeted. In a untargeted attack, attacker does not have a specific target label in mind when trying to fool a DNN model. In contrary, attacker tries to mislead a DNN model to classifies an adversarial sample \mathbf{x}' as a specific target label t in a targeted attack. In 2013, Szegedy *et al.* [133] first generated such adversarial example using Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) attack. Given a natural sample \mathbf{x} and a target label $t \neq F(\mathbf{x})$, they used L-BFGS method to find an adversarial sample \mathbf{x}' that satisfies the following box-constrained optimization problem

$$\min c \|\mathbf{x} - \mathbf{x}'\|_2^2 + J(\mathbf{x}', t), \quad (2.32)$$

subject to $\mathbf{x}' \in [0, 1]^m$ and $F(\mathbf{x}') = t$, where c is a constant that can be find by linear searching, and $m = hw$ if the image is gray scaled and $m = 3hw$ if the image is colored. Because of this computational intensive linear searching method for optimal c , L-BFGS Attack is time consuming and impractical. In the course of the next few years, many other gradient-based attacks are proposed such as

- Fast Gradient Sign Method (FGSM) is a one-step algorithm that generates perturbations in the direction of the loss gradient;
- Basic Iterative Method (BIM) is an iterative application of FGSM in which a finer perturbation is obtained at each iteration. This method is introduced by Kurakin *et al.* [81] for generating adversarial example in the physical world.

- DeepFool searches for the shortest distance to cross the decision boundary using an iterative linear approximation of the classifier and orthogonal projection of the sample point onto it [101].
- Carlini and Wagner (C&W)'s Attack [20] is introduced as a targeted attack against defensive distillation, a defense method against adversarial example proposed by Papernot *et al.* [111]. To ensure that the box-constraint $x' \in [0, 1]^n$ is satisfied, they proposed three methods, projected gradient descent, clipped gradient descent, and change of variable, to avoid box-constraint. C&W's attack is not only effective for defensive distillation but also effective for most of existing adversarial detecting defenses. In [19], Carlini and Wagner uses C&W's attack against ten detection methods.

There are other attack techniques that assume a less powerful attacker who has no access to the model. These are called black-box attacks. For instance, zeroth-order Optimization (ZOO)-based Attack estimates the gradient and Hessian using the quotient difference [24]. Though this eliminates the needed for direct access to gradient, it requires high computation cost. As another popular example, transfer attack generates adversarial samples for a substitution model and uses these generated adversarial samples to attack the targeted model. This is often possible due to the transferability of DNN models.

Researchers have developed many adversarial attack and data poisoning attack approaches over the years, but we are not able to cover all of them due to the space limit. For instance, Biggio *et al.* [14] proposed a set of poisoning attacks against SVMs, where attackers aim to increase the SVM's test error in 2012. In that study, Biggio *et al.* also demonstrated that an attacker could predict the SVM's decision function change, which can then be used to construct malicious training data.

2.3 Active Learning Methods

This section provides a brief description of existing active learning methods. Though gathering large quantities of unlabeled data is easy and cheap, obtaining the label for them is usually time-consuming and expensive, even using both high-performance and distributed computing systems. For instance, there are more text data available than what a researcher with a limited labeling budget can afford. Active learning reduces the number of labeled instances needed for training a model effectively. Active learning is an iterative process that consists of training and querying. First, a model is trained based on an initially labeled training set. After the initial model is trained, a query strategy is used to select a subset of data for label

querying. Then, an oracle, e.g., a human annotator, labels these data, and the model is retrained with currently available labeled data. The process repeats until a stopping criterion is met, such as the reach of the desired accuracy. See Fig. 2 for an overview of the general active learning framework. Various active learning algorithms have developed over the years. They can be categorized as uncertainty-based active learning, diversity-based active learning, and adversarial active learning.

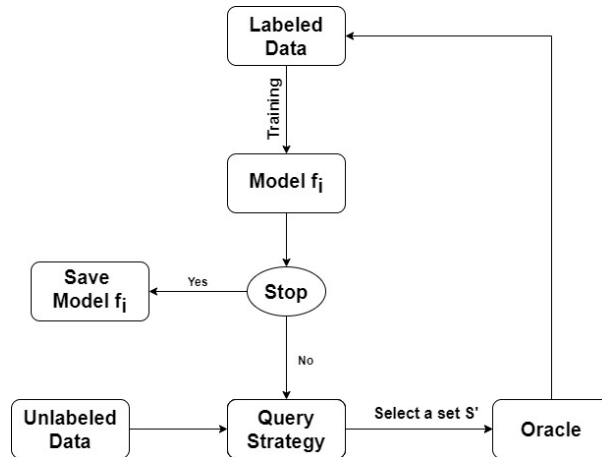


Figure 2. Active Learning Framework. Reprinted from [85] © 2021 IEEE. Used with permission as a senior author.

The uncertainty based active learning algorithms focus on selecting the instances based on the uncertainty principle [107], [126], [71]. The uncertainty principle is the idea that a training instance that is most uncertain to a model contains the most new information for improving the model. There are many ways of measuring uncertainty. For instance, the distance between an instance and the classification boundary can be used to measure the uncertainty. As the instance is closer to the classification boundary, it is more difficult for a classifier to make high confidence classification. This provides a measure of uncertainty. Another measure uncertainty is the final confidence provided by a trained model, as the lower confidence provided by a model indicates a relatively high uncertainty in its performance. An alternative to this idea is to measure the uncertainty as the difference between the two highest confidences provided by a model. The idea behind this is that if a model is sure about its prediction, it should provide a very high confidence to only one class and low confidence values for other classes. On the contrary, if a model is unsure of its prediction, it generally provides relatively high confidence values for two or more classes. For instance, if a classifier sees a blurry image and it is uncertain whether it is an image of a dog or a cat, the classifier may provide close to 50% confidence for both dog class and cat class, in which case the difference in confidence

is close to 0. On the contrary, if a classifier is certain that it is an image of a dog, it will assign close to 100% confidence for the dog class and near 0% for the cat class. In this way, the difference between the two confidence values reveals uncertainty. Other measures of the uncertainty include entropy [73], smallest predicted class probability [145], etc. See, e.g., [126] and [9] for an overview of classic active learning methods.

However, recent work on data poisoning attack, e.g., [41], [142], [159], has shown that uncertainty sampling may help attackers in achieving their goals. As the poisoning instances are usually laid around the decision boundary, these poison instances are more likely to be selected by the uncertainty sampling approaches.

Another class of active learning algorithms focuses on selecting the most diverse and representative dataset. For instance, Yang et al. [132] proposed an optimization-based active learning algorithm that imposed a diversity constraint objective function. The core-Set approach [125] is a diversity-based approach using core-set selection. Batch Active learning by Diverse Gradient Embeddings (BADGE) algorithm [7] selects a subset of instances whose gradients with respect to the parameters in the output layer are most diverse.

The adversarial active learning algorithms, e.g., [160] and [95], are a relatively new idea and are usually based on the Generative Adversarial Networks (GAN) [50], Variational Auto-Encoders (VAE) [77], or adversarial examples (AE) [133], [131], [52], [101]. The first active learning work using GAN is proposed by Zhu et al. [160]. GAN enables the synthesis of additional instances from the labeled training instances. However, its performance is not as good as active learning based on the random sampling strategy. The first active learning algorithm for multi-classification using GAN is Adversarial Sampling for Active Learning [95].

Chapter 3

Secure Machine Learning Against Adversarial Samples at Test Time

Deep Neural Networks (DNN) are widely used to handle many difficult tasks, such as image classification, malware detection, medical diagnosis, self-driving cars, as well as digital assistants like Google Assistant, Alexa, and Siri, and achieve outstanding performance. For instance, the state-of-the-art performance of image classification on the ImageNet dataset is 73.8% in 2011 but 98.7% (Top 5 Accuracy) in 2020. This outstanding performance surpassed human annotators. However, recent studies on adversarial examples, which have maliciously undetectable perturbed inputs that are similar to their original samples to human eyes but misled by machine learning approaches, show that machine learning models are vulnerable to security attacks. Many researchers have attempted to secure neural networks through techniques such as defensive distillation, classifier robustifying, input reconstruction, network verification, and adversarial detection. Nevertheless, many of these techniques are ineffective to new or stronger adversarial attacks such as the Carlini and Wagner (C&W)'s attacks. Although various adversarial retraining techniques have also been developed in the past few years, none of them are scalable. In this chapter¹, we propose a new iterative adversarial retraining approach to robustifying the model and to reducing the effectiveness of adversarial inputs on DNN models. The results from our extensive experiments demonstrate that the proposed approach increases the robustness of the DNN model against various adversarial attacks, specifically, fast gradient sign attack, Carlini and Wagner (C&W)'s attack, Projected Gradient Descent (PGD) attack, and DeepFool attack. To be precise, the classifier obtained by the proposed approach can maintain the performance accuracy of 99% on average. Moreover, the experimental evaluation on the runtime of two of the most effective adversarial attacks, i.e., C&W's attack and PGD attack showed that C&W's attack can utilize GPU for faster adversarial example generation than the PGD attack does. For this reason, a parallel implementation of the

¹Portions of this chapter have been previously published by the author of this dissertation in [87, 88, 89], where no formal reuse license is needed for the IEEE © [2020] publication [87] and Springer Open © The Authors 2022 [88] and remixing and adapting arXiv [89] CC BY-NC-SA 2021 is permitted for their senior authors.

proposed approach is developed. This parallel implementation makes the proposed approach scalable for large datasets and more complex models.

Contrary to some existing studies that attempt to detect the adversarial examples at the cost of reducing the classifier’s accuracy, we aim to maintain the classifier’s prediction accuracy by increasing model capacity and generating additional training data. Moreover, this proposed approach can be applied to any classifiers since it is classifier-independent. Particularly, the proposed approach is useful for safety- and security-critical systems, such as machine learning classifiers for malware detection [118] and autonomous vehicles [15].

3.1 Introduction

Various proactive and reactive defense methods against adversarial examples have been proposed over years [154]. Examples of proactive defenses include adversarial retraining [52], [68], defensive distillation [111], and classifier robustifying [16], [1]. Examples of reactive defenses are adversarial detection [45], [55], [62], [13], [84], [150], input reconstruction [56], and network verification [74], [53]. However, all defenses are later shown to be either ineffective for stronger adversarial attacks such as C&W attack or cannot be applied to large networks. For instance, Reluplex [74] is only feasible for a network with only a few hundreds of neurons.

Our proposed iterative approach can be considered as an adversarial (re)training technique. First, we train the model with normal images and normal images with random Gaussian noises added (the later ones are simply called noise images in this chapter). Next, we generate adversarial images using attack techniques, such as PGD, DeepFool, and C&W attacks. Then, we retrain the model based on the adversarial images generated in the previous step, normal images, and noise images. This step can be considered as a combination of adversarial (re)training [133] and Gaussian data augmentation (GDA) [156]. However, we use soft labels instead of hard labels. Repeat this retraining process until the acceptable robustness or maximum iteration number is reached. The resulting classifier has n classes. The first $n - 1$ classes are normal image classes, and the last one is the adversary class.

After the model is fully trained, we can test it. At the testing time, a small random Gaussian noise is added to a given test image before it is inputted into the model. Since our model is trained with Gaussian noise, it does not affect the classification of the normal images. However, if the test image is adversarial, it is likely to be generated from optimization algorithms that introduce well-designed minimal perturbation to a

normal image. We can improve the likelihood of distinguishing the normal image from the adversarial image by disturbing this well-calculated perturbation with noises. Additionally, we propose an ensemble model in which a given test image without random noise is also input to the model, and its output is compared with the output of the test image with random noise added. If two outputs are the same, that is the final output of the model. If two outputs are different, a given test image is marked as adversarial to alert the system for further examination.

The key contributions of this paper are in the following:

1. The iterative process is highly parallelizable. Since the iterative model at time t is not much different from the iterative model at time $t - 1$, the older model for generating adversarial examples should work as well as the current model due to the transferability of the adversarial examples. Some GPU nodes can be used to generate adversarial examples based on the previous iterative model instead of waiting for a new iterative model. Therefore, the adversarial example for the next iteration t can be generated before the new iterative model is produced.
2. The trained model based on our proposed methodology will not only be robust against adversarial examples but also maintains the accuracy of the original classifier. Some existing methods strengthen the model with respect to some attacks but reduce the accuracy of the classifier at the same time. Our model, through multiple data generating methods and larger network capacity, refines the data representation at each iteration; therefore, we can maintain our robustness and accuracy at the same time.
3. We proposed an ensemble model that outputs the final label based on the labels provided by two tests, one with added small random noise and one without. Small random noise is aimed to distort the optimal perturbation injected by the adversary. We have trained our model against random noise at training time; hence, the small random noise does not affect the classification accuracy of normal images. However, it may disturb the optimal adversarial example generated by an adversary. If the original image and image with added test noise produce different outputs from the model, the input image is likely to be adversarial.

The remainder of this chapter is organized as follows. We introduce the threat model in section 3.2. In section 3.3, we present related work. In section 3.4, we present the proposed approach, followed

by an evaluation in section 3.5. In section 3.6, we discuss the implications and limitation of the proposed approach. Finally, we give conclusions and future work in section 3.7.

3.2 Threat Model

Before discussing the related work, we define the threat model formally. In this work, we consider evasion attacks in both white-box settings (assume an adversary has full knowledge of the trained model F) and grey box settings (assume an adversary has no knowledge of the trained model F but has knowledge of the training set used). The ultimate goal of an adversary is to generate adversarial examples \mathbf{x}' that misled the trained model. That is, for each input (\mathbf{x}, y) , the adversary's goal is to solve the following optimization problem

$$\begin{aligned} \min_{\delta} \quad & \|\mathbf{x} - \mathbf{x}'\|_2^2 \\ \text{s. t.} \quad & \|\mathbf{x} - \mathbf{x}'\|_2^2 < \epsilon, \mathbf{x}' \in [0, 1]^n, F(\mathbf{x}') = y', \end{aligned} \quad (3.1)$$

where $y \neq y'$ are class labels and ϵ is a maximum allowable perturbation that is undetectable by human eyes.

In this chapter, we consider the DNN models for image classification. A grey scaled image \mathbf{x} has $h \times w$ pixels; i.e., $\mathbf{x} \in R^{hw}$, where each component x_i is normalized so that $x_i \in [0, 1]$. Similarly, for a colored image \mathbf{x} with a RGB channel, $\mathbf{x} \in R^{3hw}$. In the following subsection, we will consider the attack approaches for generating adversarial image \mathbf{x}' from the natural or original image \mathbf{x} , where \mathbf{x} can be either gray scaled or colored.

3.3 Related Work

Machine learning automates the tasks of writing rules for a computer to follows. That is, giving the input and the desired outcome, machines learning can find a set of rules needed automatically. As ML systems have been dramatically integrated into a broad range of decision-making-sensitive applications for the past years, adversarial attacks and data poisoning attacks have posed a considerable threat against these systems. For this reason, section 2.2 focuses on the two important areas of ML security: adversarial attacks and data poisoning attacks. Specifically, it has comprehensively described, discussed, and scrutinized the

adversarial attacks and the data poisoning attacks with regard to their adversarial capabilities. The main goal of section 2.2 is to gain insights and implications of existing adversarial attacks and data poisoning attacks, as well as to increase the awareness of potential adversarial threats when ones develop learning algorithms and apply ML methods to various applications in the future. For instance, PDFrate and Hidost are two machine learning classifiers that are based on the random forest and support vector machine (SVM), respectively. Xu *et al.* [151] used genetic programming to generate adversarial examples that can bypass those malware classifiers at 100% of the time. That is, they can modify the malware slightly so that it can evade these classifiers with a 100% success rate. The modification makes those PDF malware classifiers useless. Similarly, Morgulis *et al.* [103] have demonstrated that traffic sign recognition systems of a car can be easily fooled by adversarial traffic signs and cause the car to take unwanted actions. Likewise, an adversarial example can be generated against the semantic segmentation task of a self-driving car by hiding pedestrians. Without noticing the pedestrians on the crosswalk, a self-driving car will not stop and can cause a deadly accident [97]. This is a serious safety problem, which must be addressed.

Over time, many defense mechanisms have been proposed as well. In the published survey paper [154], researchers recently summarized three main proactive countermeasures for adversarial examples: network/defensive distillation [111], adversarial (re)training [52], [68], and classifier robustifying. Network distillation used distillation, a technique usually used to reduce the dimensionality, to decrease the success rate of adversarial sample crafting [111]. A DNN model is trained and its softmax output is smoothed by a temperature parameter $T > 0$ using the equation

$$q_i = \frac{e^{\frac{z_i}{T}}}{\sum_{j=1}^n e^{\frac{z_j}{T}}},$$

where q_i is the smoothed probability of class i that will be input to second DNN, z_i is the probability of class i output by first DNN, and n is the total number of classes. Then, the original hard labels are replaced by the soft labels $\{q_i\}_{i=1}^n$ when training the second DNN. This way the loss function is smoother with soft targets, and so it robustify the model. Defensive distillation is shown to reduce the success rate of JSMA attack by 98.6% and 81.36% on the MNIST and CIFAR-10 dataset, respectively. Alternative method of network distillation is label smoothing [134]. Instead of using the first DNN to generate soft labels, they proposed just to assign a high probability, like 0.9, to the right class and then distribute rest probabilities among other $n - 1$ classes if there is a total of n classes. For our proposed approach, we also use soft labels. However,

we use soft labels not only to reduce adversarial gradients and sensitivity to input perturbation but also to prevent the network from becoming over-confident and improve generalization and model calibration [104].

Adversarial (re)training is another countermeasure, where an adversarial example is generated and injected into the training set for retraining [52], [68]. Usually, a base model is used to generate some adversarial samples using techniques such as FGSM, JSMA, BIM, etc. Then, these adversarial samples with the correct labels and the natural samples are mixed and used to retrain the model. However, Grosses [55] proposed a variate adversarial retraining by classifying the adversarial examples as the $n + 1^{th}$ classes. Nevertheless, Tramèr et al. [139] showed that an adversarially trained model using single-step attack methods such as FGSM is still vulnerable to other simple and powerful first-step attacks and transfer attacks. They showed that a two-step attack can easily bypass the classifier trained using adversarial examples by first adding a small noise to the system and then performing any classical attack technique such as FGSM, DeepFool, etc. Instead of injecting adversarial examples to the training set, Zantedeschi et al. [157] suggested to add small noises to the normal images to generate additional training examples. They compared their methods with other defense methods such as adversarial training, label smoothing, etc and showed that their approach is robust and does not compromise the performance of the original classifier. The last type of proactive countermeasures is classifier robustifying. The goal is to build more robust neural networks [16], [1]. For instance, in [16], Bradshaw combined DNN with Gaussian processes (GP) to make scalable GPDNN and showed that it is less susceptible to the FGSM attack.

Furthermore, Yuan, et al. [154] introduced three reactive countermeasures for adversarial examples: adversarial detection [45], [55], [62], [13], [84], [150], input reconstruction [56], [130], and network verification [74], [53]. Adversarial detection consists of many techniques for adversarial detecting. For instance, Feinman *et al.* [45] assumed that the distribution of adversarial sample is different from the distribution of natural sample and proposed a detection method based on the kernel density estimates in the subspace of the last hidden layer and the Bayesian neural network uncertainty estimates with dropout randomization. However, Carlini and Wagner [19] showed that the kernel density estimation, which is most effective defense technique among ten defenses considered by Carlini and Wagner on MNIST, is completely ineffective on CIFAR-10. Grosse *et al.* [55] used Fisher's permutation test with Maximum Mean Discrepancy (MMD) to check where a sample is adversarial or natural. Though MMD is a powerful statistical test, it fails to detect C&W's attack [19]. This indicates there may not be significant difference between the distribution of

adversarial sample and the distribution of natural sample when the adversarial perturbation is subtle. Xu et al. [150] used feature squeezing techniques such as color bit depth reduction and spatial smoothing to detect adversarial examples, and the proposed technique can be combined with other defenses such as defensive distillation for defense against adversarial example.

Input reconstruction is another category of reactive defense, where a model is used to find the distribution of natural sample and input is projected onto data manifold. In [56], a denoising contractive autoencoder network (CAE) is trained to transform an adversarial example to the corresponding natural one by removing the added perturbation. Song et al. [130] used PixelCNN, which provides the discrete probability of the raw pixel values in the image, to calculate the probabilities of all training images. At the test time, a test instance is inputted and its probability is computed and ranked. Then a permutation test is used to detect adversarial perturbation. In addition, they proposed PixelDefend to purify adversarial example by solving the following optimization problem:

$$\max_{\mathbf{x}'} P(\mathbf{x}') \text{ subject to } \|\mathbf{x}' - \mathbf{x}\|_{\infty} \leq \epsilon.$$

Last but not the least, network verification formally proves whether a given property of a DNN model is violated. For instance, Reluplex [74] used satisfiability modulo theory solver to verify whether there exists an adversarial example within a specified distance of some input for DNN model with ReLU activation function. Later, Carlini *et al.* [22] showed that Reluplex can also support max-pooling by encode max operators as follows.

$$\max(x, y) = \text{ReLU}(x - y) + y. \tag{3.2}$$

Initially, Reluplex could only handle the L_{∞} norm as a distance metric. Using 3.2, carlini *et al.* [22] encoded absolute value as

$$\begin{aligned} |x| &= \max(x, -x) = \text{ReLU}(x - (-x)) + (-x) \\ &= \text{ReLU}(2x) - x. \end{aligned}$$

In this way, Reluplex can handle the L_1 norm as a distance metric as well. However, Reluplex is computationally infeasible for large networks. More specifically, it can only handle networks with a few hundred neurons [22]. Using Reluplex and k-means clustering, Gopinath et al. [53] proposed DeepSafe to provide

safe regions of a DNN. On the other hand Reluplex can be used by an attacker as well. For instance, Carlini *et al.* [22] proposed Ground-Truth Attack that uses C&W attack as initial step for a binary search to find an adversarial example with the smallest perturbation by invoking Reluplex iteratively. In addition, Carlini *et al.* [22] also proposed a defense evaluation using Reluplex to find a provably minimally distorted example. Since it is based on Reluplex, it is computationally expensive and only works on small networks. However, Yuan et al. [154] concluded that almost all defenses have limited effectiveness and are vulnerable to unseen attacks. For instance, Carlini and Wagner (C&W)’s Attack is effective for most of existing adversarial detection methods though it is computationally expensive [19]. In [19], authors showed that ten proposed detection methods cannot withstand white-box attack and/or black-box attack constructed by minimizing defense-specific loss functions.

3.4 Methodology

In this section, we present our iterative approach for generating more data from the original dataset to train our proposed model. The training process can be summarized in the following steps:

1. Generate adversarial examples using existing *adversarial example crafting techniques*. In the evaluation section, we only use the PGD and C&W attacks to generate adversarial examples since they are two the strongest attacks in existence and we have limited computation resource.
2. Generate more noisy images by adding the small random noises to the regular images. This is needed to make sure that the final model is not skewed toward the current methods for generating adversarial examples, and we want our final model to be robust against random noise.
3. Combine the regular training set with the adversarial images and noisy images generated in 1 and 2, respectively, to train the model. Instead of hard labels for these images, soft labels are used instead. For instance, hand-written “7” and “1” are similar in that both have a long vertical line. Hence, rather than label a hand-written digit as 100% “7” or 100% “1”. We may say it is 80% “7” and 20% “1”. This shows the structure similarity between “7” and “1”. More precisely, the soft labels for an adversarial image, a random image, and a clean image is defined as follows, respectively. Let τ be a hyperparameter that measures the acceptable size

of perturbation. Let us first define the soft label, $p_i(x_j)$, for an adversarial image based on the value of τ in the following two cases.

- (a) If $\eta < \tau$, then we define the soft label $p_i(x_j) = \alpha + \frac{1-\alpha}{n}$ for the adversarial image x'_j generated using the real image x_j that belongs to class i , and $\frac{1-\alpha}{n}$ otherwise, where $0 < \alpha < 1$ is close to 1 and n is the number of classes.
- (b) If $\eta \geq \tau$, we define the soft label $p_i(x_j) = \beta$ for the adversarial image x'_j generated using the real image x_j that belongs to class i , the soft label $p_n(x_j) = \gamma$ for the adversarial image x'_j generated using the real image x_j that belongs to the adversarial class, and the soft label $p_k(x_j) = \frac{1-\beta-\gamma}{n-2}$ for adversarial image x'_j generated using the real image x_j that belongs to class k , where $k \neq i, n$, $0 < \beta, \gamma < 1$, and $0 < \beta + \gamma < 1$.

The soft label for a random-noise image is defined similarly. The α and τ values used for soft label calculation of adversarial images and random noise images are shown in Table 2. For simplicity, the correct class is assigned a soft label of 0.95, whereas other classes are assigned a soft label of 0.05/9 for the clean images.

4. Check robustness of the model if it is used as a stopping criterion. In [157], the robustness of a model is defined in terms of the expected amount of L_2 perturbation that is required to fool the classifier:

$$\rho = E_{(x,y)} \frac{\eta}{\|x\|_2 + \delta}, \quad (3.3)$$

where η is the amount of L_2 perturbation that an adversary added a perturbation to a normal image x , and δ is an extremely small constant allowing for the division to be defined, say $\delta = 10^{-10}$. We use this definition of robustness in this paper since this definition captures the intuition that larger perturbation is required to fool the classifier indicates a more robust classifier.

5. While $k < k_{max}$ and/or the robustness of the model $\rho < \rho_0$, where ρ_0 is an acceptable level of robustness selected by an user and k_{max} is the maximum number of iterations allowed, this step repeatedly generates and accumulates a large amount of data for training a robust model.

- (a) Sample a mini-batch of N stratified random images from a real images set and generate additional images using *adversarial example generating techniques* (shown in step 1) and

random perturbation (shown in step 2). This combined sample is then used in the next step to retrain the model.

- (b) Retrain the model. Update the model weights by minimizing the following cross-entropy loss:

$$J_D = -\frac{1}{N} \sum_{j=1}^N \sum_{i=1}^n p_i(x_j) \log(q_i(x'_j)), \quad (3.4)$$

where n is the number of classes. The n^{th} class is the adversarial class, whereas classes 1 to $n - 1$ are regular image classes.

The implications of the above proposed approach are discussed in section 3.6, where we have showed that our proposed approach is robust when a small noise is injected to the test instance before the classification is performed, and it is also robust against both white-box attacks and black-box attacks. However, we have not studied whether or not our proposed approach is robust against adaptive attacks. A further study is needed in the future.

3.5 Evaluation

In this section, we empirically evaluate our proposed approach on MNIST [82] dataset and CIFAR-10 [79], canonical datasets used by most papers for attack and defense evaluation [150], [111]. The performance metric considered is the accuracy of the classifier.

3.5.1 Datasets

The MNIST handwritten digit recognition dataset [38] is a subset of a larger set available from NIST. It is normalized, where each pixel is range between 0 and 1 and each image has 28×28 pixels. There are 10 classes: the digits 0 through 9. All the images are black and white. The samples are split between a training set of 60,000 and a testing set of 10,000. Out of 60,000 training instances, 10,000 are reserved for validation, and 50,000 are used for actual training. To evaluate the scalability of the proposed robust classifier, we consider CIFAR-10, which is a $32 \times 32 \times 3$ colored image dataset consisting of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The samples are split between a training set of 50,000 and a testing set of 10,000.

3.5.2 Network Architectures

The network architecture for the MNIST dataset consists of two ReLU convolutional layers, one with 32 filters of size 3 by 3 and follows by another with 64 filters of size 3 by 3. Then, a 2 by 2 max-pooling layer and a dropout with a rate of 0.25 is applied before a ReLU fully connected layer with 1024 units. The last layer is another fully connected layer with 11 units and a softmax activation function for classification in 10 regular classes plus an adversarial class. The total number of the parameters is 6,616,970 for the original model with 10 regular classes only. The proposed model with 11 classes has 6,617,995 parameters. The accuracy of the model is 99%, which is comparable to the accuracy of the state-of-the-art DNN. The network architecture for the CIFAR-10 model is a ResNet-based model provided by the IBM [108]. The hyperparameters used for the training and the corresponding values are shown in the following table. For instance, we set the acceptable level of robustness ρ_0 to 0.1.

Table 1 Training Parameters. Reprinted from [88] © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Parameter	MNIST
$\alpha_{NormalNoise}$	0.44
α_{BIM}	0.44
α_{CW}	0.54
α_{DF}	0.54
$\tau_{NormalNoise}$	0.1
τ_{BIM}	0.1
τ_{CW}	0.0265
τ_{DF}	0.0265
β	$\frac{\alpha}{2} + \frac{1-\alpha}{11}$
γ	β
ρ_0	0.1
Batch Size	60
k_{max}	3000
Steps	100

3.5.3 Adversarial Crafting

We use Adversarial Robustness Toolbox (ART v0.10.0) [108] to generate adversarial examples for training and testing. The training set is used to generate the adversarial examples for training, and the test set is used to generate the adversarial examples for testing. Due to the limited computing resource, we only use BIM and C&W attacks and Gaussian noise to generate additional images for training. The BIM and C&W

attacks are selected over others because they generate the strongest attacks [94], and Gaussian noise is added to the images to take care of other potential attacks and for better data representation. The soft labels for those images are based on the perturbation introduced. For instance, under the C&W attack, we use a soft label of 0.44 if the perturbation limit is less than 0.026. 0.026 is selected based on the observation that the perturbation within this limit is hardly noticeable to human eyes. For a similar reason, the perturbation limit of BIM and Gaussian noise is set to 0.15 and 0.25 with a soft label value of 0.44.

At the testing time, we consider the strong white-box attack against our proposed model. The adversarial images are generated using FGSM, DeepFool, and C&W attack with the assumption that an adversary has feature knowledge, algorithm knowledge, and the ability to inject adversarial images. FGSM attack is selected because it is a popular attack technique using by many papers for evaluation. BIM, DeepFool, and C&W attacks are selected because they are the three currently strongest existing attacks. For the FGSM attack, the perturbation of $\epsilon = 0.1$ is considered. This is a reasonable limit because a larger perturbation can be detected by human and/or anomaly detection systems. Similarly, the maximum perturbation for the BIM attack is also set to 0.1 and the attack step size is set to $\frac{0.1}{3}$. The maximum number of iterations for BIM is 40. The settings of the C&W and DeepFool attacks are left as default in ART [108]. Random noise added to train images follows a Gaussian distribution with mean 0 and variance η_{max} for each batch.

3.5.4 Results

We consider the accuracies of the classifiers on the normal MNIST and CIFAR-10 test images and the accuracies of the classifier under FGSM, C&W, BIM, and DeepFool attacks, respectively. The prediction accuracy of the original classifier on the normal MNIST test images is 99%. The prediction accuracy of the robust classifier on the normal MNIST test images is also 99% after retraining. Furthermore, the accuracies of the robust classifier under FGSM, C&W, BIM, and DeepFool attacks are shown in Table 2. These results have shown that after retraining, the model performance dramatically improved under these attacks. As shown, the original model’s classification accuracy drops significantly under the attacks. On the contrary, our robust classifier maintains the classification accuracy even under the attacks. Compared to the adversarial retraining method implemented by IBM’s adversarial robustness toolbox, our robust classifier performs better under DeepFool and C&W attacks.

To evaluate the performance of the classifiers under grey-box attacks, we assume that an attacker has no knowledge of a neural architecture. In this case, an attacker builds his/her own approximated CNN model

and conducts the transfer attacks. We assume that an attacker develops a simple CNN model consisting a CNN layer with 4 filters of size 5 by 5, followed by a 2 by 2 max-pooling layer and a fully connected layer with 10 units and a softmax activation function for classification. The accuracy of the model is 99%, which is comparable to the accuracy of the state-of-the-art [155]. Table 3 shows that the original model performs poorly under black-box attack. In fact, it is worsen than what is under the white box attack. However, the black-box attacks have little affected under the adversarially retrained models.

Table 2 Classification Accuracy on the MNIST Test Dataset Under the White-box Attack. Reprinted from [88] © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Attack	FGSM	C&W	BIM	DeepFool
Original	29%	7%	7%	29%
Adversarial Training (ART)	98%	49%	98%	42%
Robust Classifier	100%	98%	99%	99%

Table 3 Classification Accuracy on the MNIST Test Dataset Under the Grey-box Attack. Reprinted from [88] © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Attack	FGSM	C&W	BIM	DeepFool
Original	11%	9%	4%	44%
Adversarial Training (ART)	98%	99%	98%	99%
Robust Classifier	99%	99%	99%	99%

Moreover, a similar experiment has been performed on the CIFAR-10 dataset for the evaluation of the scalability of the proposed robust classifier. The experimental results are shown in Table 5. Note that we do not conduct any hyperparameter tuning due to the constraints of HPC cluster resource and time. That is, we have used the same hyperparameter settings, as shown in Table 2. The robust classifier performs better than the original model, though the improvement is not as good as for the MNIST dataset. As shown in this table, the accuracy has been improved by 31%, 32%, 72%, and 23% under FGSM, CW, BIM, and Deep Fool attack, respectively.

3.6 Discussion

In this section, the implications of the mechanism are discussed.

Table 4 Classification Accuracy on the CIFAR-10 Test Dataset Under White-box Attacks. Reprinted from [88] © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Attack	FGSM	C&W	BIM	DeepFool
Original	16%	46%	8%	13%
Robust Classifier	47%	78%	80%	36%

3.6.1 Accuracy vs. Resilience Against Adversarial Examples

One common problem of adversarial training and other defense methods is the trade-off between the accuracy and resilience against adversarial examples. This trade-off exists because the network architectures of an original system and its proposed one are similar and/or the dataset size is fixed. However, our proposed approach has a larger network capacity with a larger dataset generated by multiple techniques. Hence, our proposed approach does not have this trade off as shown in section 3.5. This solves the problem of trade off between the accuracy and the robustness against an adversarial example and makes the model not only robust against a strong adversarial attack but also maintains the performance for classification. See section 3.5 for detail.

3.6.2 Training Time

Another consideration is the training time. To check the number of epochs needed for the training, the model is trained for ten epochs, as shown in Figure 3. The total training time for ten epochs is about 90 seconds on Google Colab (with 12GB NVIDIA Tesla K80 GPU). To prevent overfitting, three epochs are selected for training the original model. A similar procedure is used to determine that ten epochs are needed for the proposed model. However, the proposed approach is highly parallelizable. Due to the transferability of adversarial examples, the adversarial examples do not have to be generated using the current model. Hence, adversarial crafting and adversarial training can be performed simultaneously. That is, at iteration t , an adversarial example can be generated based on the model generated at iteration $t' < t$, and adversarial training can be performed using the adversarial example generated previously as well. Depending on the memory and computation resource, we can store the generated adversarial example at each iteration and sample from it when performing the adversarial training. The sampling probability can be based on the performance of the model at previous iterations. For instance, initially, we assign a probability of $\frac{1}{n}$ to each

adversarial example and increases its probability for the next iteration if the model misclassifies it or it is not selected for the current iteration of training and decreases its probability if the model correctly classifies it. Furthermore, we can parallelize the adversarial crafting step (or the fake image generation step) by assigning a CPU (or GPU) for each adversarial crafting technique. Similarly, data parallelization can be performed during training.

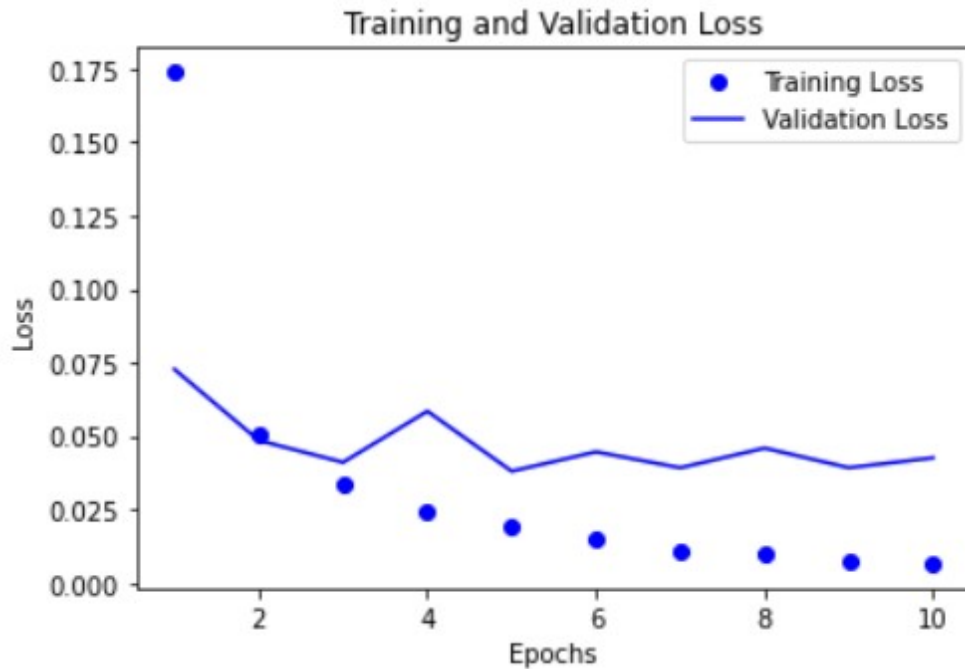


Figure 3. Training and Validation Loss. The training loss decreases with every epoch, whereas the validation loss does not decrease much after the third epoch. Hence, the training process can be stopped after three epochs to prevent overfitting. Reprinted from Lin, J., Njilla, L.L. & Xiong, K. Secure machine learning against adversarial samples at test time. EURASIP J. on Info. Security 2022, 1, doi: 10.1186/s13635-021-00125-2. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

The framework for parallelization is shown in Figure 4. Initially, the original model and a random sample of the original images are used to generate fake images. The original model is needed because we want to generate adversarial images by just adding an application-specific imperceptible perturbation to the original image and make it misclassified by the original model. There are various ways to obtain such an application-specific imperceptible perturbation. For instance, the FGSM attack generates such a perturbation by using the model's loss gradient as described in section 2.2.1.2. After the fake images are generated, it is saved to the fake image storage folder. During the retraining step, a random sample of fake

images and original images as well as the copies of the original model are sent to the multiple-GPUs for retraining. After the retraining, the updated model is checked to see if it is robust against adversarial attacks. If so, the updated model is saved, and the process ends. Otherwise, the updated model is saved as the new model for the next stage of fake image generation and retraining.

To demonstrate the importance of selecting right the resource for different *adversarial example crafting techniques*, we conduct an experiment on a standard alone personal computer (12 thread CPUs). First, we independently run each attacks on the same computer and measure the run time for generating a batch of 64, 128, 256, and 512 adversarial examples without GPU resource. To reduce the variation, we repeat the same experiment for 30 times for each attack. Then, we install K40C, TitanV, and both K40C and TitanV, respectively and repeat the experiments. The result is shown in Figures 5 and 6. The BIM attack run faster on CPU than on GPUs. In fact, when an attacker tries to utilize the GPUs, the adversarial image generation speed goes down. As shown in Figure 5, though we utilize different types of GPU, the adversarial image generation speed does not change among the GPU types. On the contrary, C&W attack takes advantage of the GPU and runs faster on GPU than on CPU. Furthermore, we see that C&W attack runs faster on Titan V than on K40c. Table 5 is obtained by averaging over batch sizes. As shown, the BIM attack runs twice as fast on CPU than GPU whereas C&W attack runs faster on GPU. The variation among the runs is smaller under BIM attacks on average, as shown with smaller standard deviations. Moreover, we create a row called Titan V + K40c (average), which takes the average speed of Titan V and K40c. Comparing the values on this row with the value in the last row, we see that actual speed with two GPUs is not equal to the average speeds if GPU is utilized (as in C&W attack).

Table 5 GPUs vs. CPU. Reprinted from [88] © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Node	Average		Standard Deviation	
	BIM	C&W	BIM	C&W
CPU	0.752	0.477	0.0088	0.01485
K40c	1.431	0.172	0.02139	0.01193
Titan V	1.453	0.098	0.00129	0.02398
Titan V + K40c (average)	1.442	0.135	0.01134	0.01795
Titan V + K40c (actual)	1.447	0.097	0.00151	0.01995

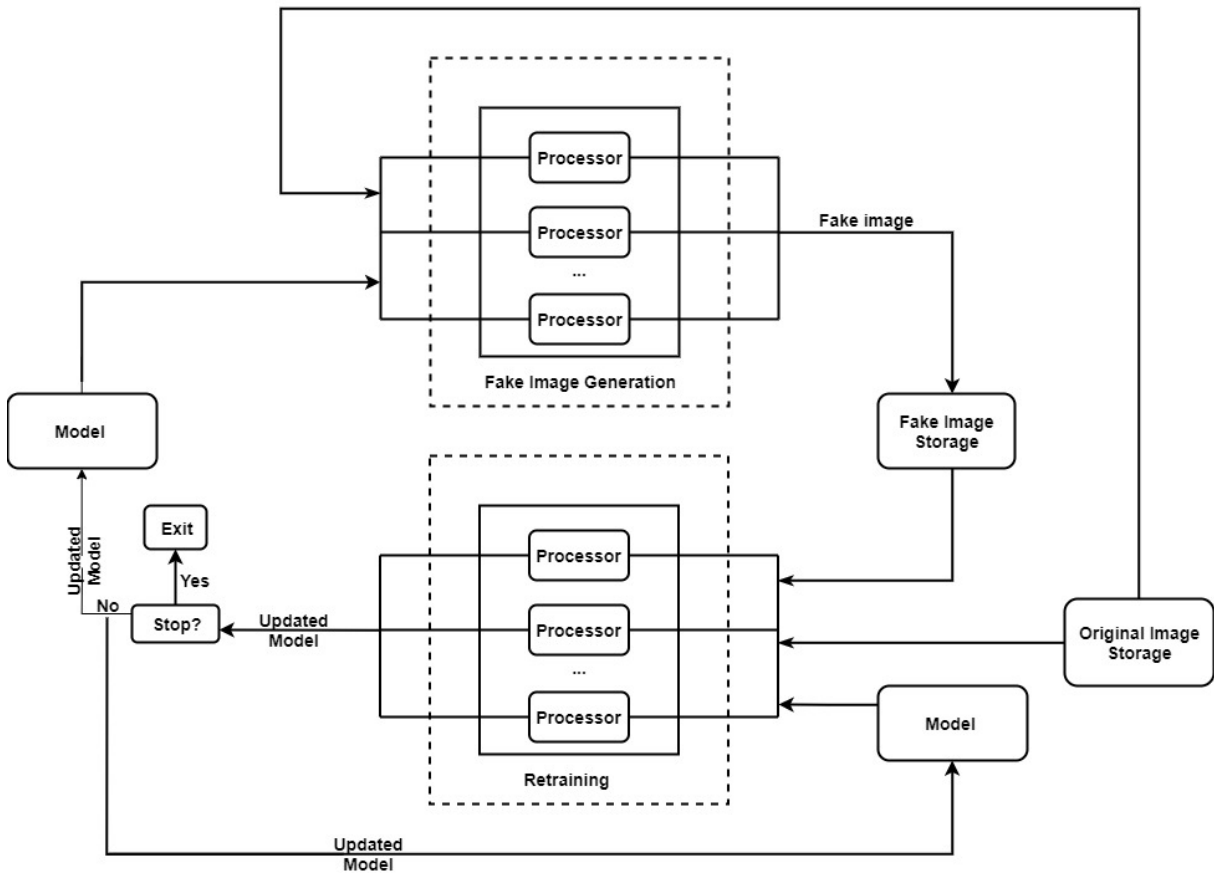


Figure 4. Parallelization Framework. We decouple the fake image generation step from the retraining step by only updating the model used for the fake image generation as a new model is obtained. This way, the fake image generation step can continuously generate adversarial images while the retraining continues. Furthermore, the fake image generation step is parallelized by using multi-processing. Note that the processors can be GPUs or CPUs. GPUs are better in term of computation efficiency, and CPUs can store more images. Reprinted from Lin, J., Njilla, L.L. & Xiong, K. Secure machine learning against adversarial samples at test time. EURASIP J. on Info. Security 2022, 1, doi: 10.1186/s13635-021-00125-2. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

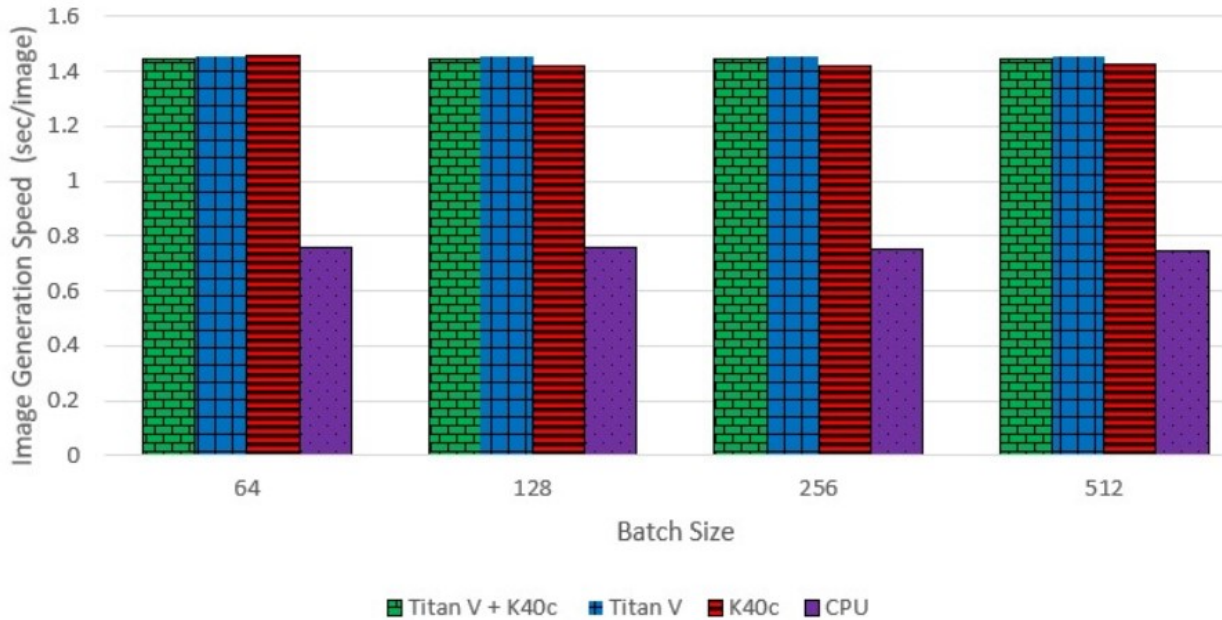


Figure 5. Adversarial Image Generation Speed for BIM Attacks. Image generation speeds under the BIM attack. Note that CPU has a much faster adversarial image generation speed than K40C, TitanV, and both K40C and TitanV. Reprinted from [88]. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

3.6.3 Ensembling

At the testing time, a small random noise is injected to the test instance before performing the classification. This small noise is aimed to distort the intentional perturbation injected by the adversarial. An experiment is performed to illustrate it as follows. First, 100 adversarial images are generated using two popular adversarial attack algorithms, BIM and FGSM. Then, using NumPy’s [59] normal number generator with a mean of 0.01 and variance of 0.01, a small random normal noise is generated for each adversarial image. This small random normal noise is injected into an adversarial image. Next, the original classifier classifies these noisy adversarial images. If the predicted label is different from the ground truth, then the attack is successful (we considered an untargeted attack). Otherwise, it is unsuccessful. This process is repeated 30 times. The average result is shown in Figure 7. Even without robust adversarial training, the classifier (original) can reduce the attack success rate by 14% for the FGSM attack and 20% for the BIM attack when the perturbation injected by an attacker is 0.01. Even when the perturbation increases to 0.05, the injected random noise can reduce the attack success rate of FGSM and BIM by 6% and 14%, respectively.

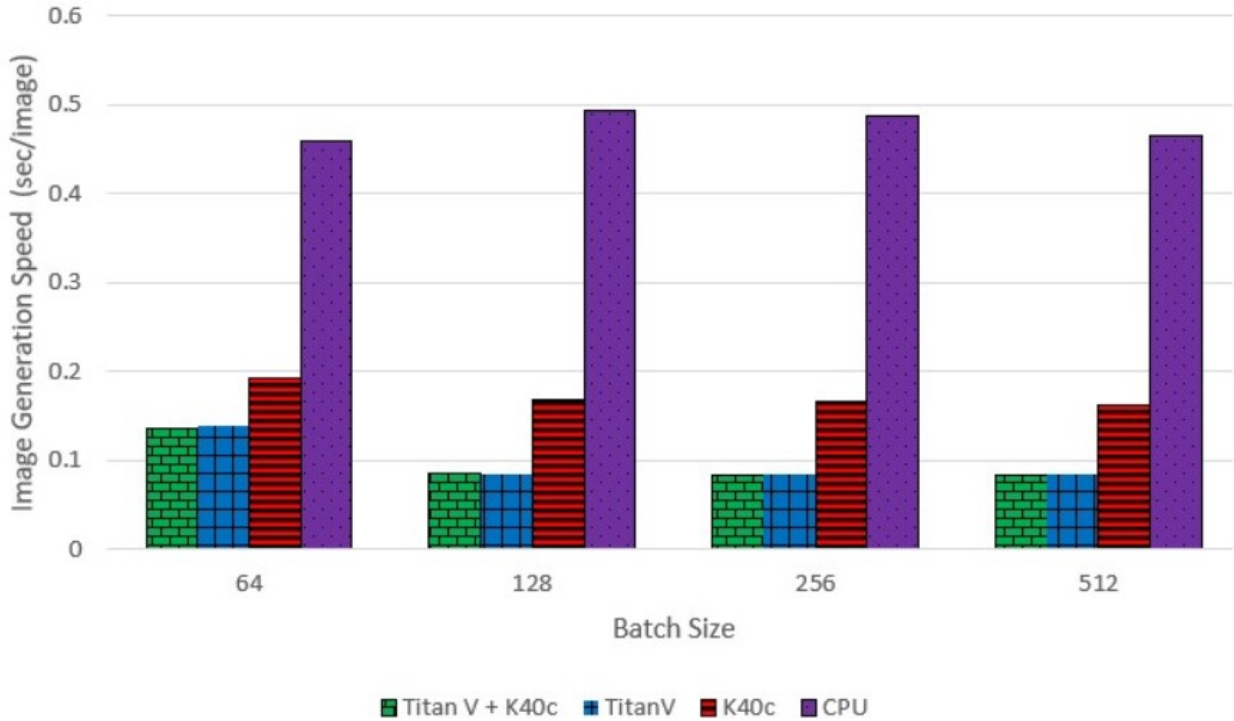


Figure 6. Adversarial Image Generation Speed for C&W Attacks. Note that CPU has a much slower adversarial image generation speed than K40C, TitanV, and both K40C and TitanV in the attack. That is, the C&W attack has an opposite result than the BIM attack. Reprinted from [88]. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Since our proposed model is trained with random noise, it is robust against it. The small noise added to the natural image is not likely to change the label of the image since we have trained our model with small Gaussian noise. However, if the test instance is adversarial, then it is likely produced by an optimization algorithm that searches for a minimal perturbation to a clean image. However, the injected random noise to such a clean image will mess up such well-calculated minimal perturbation. Therefore, if the label of a given test image without added random noise is different from that of with added random noise, this is likely an adversarial image and the system is alerted.

3.6.4 Limitations

The Evaluation section considered both white-box attacks (the attacker knows model architecture) and black-box attacks (the attacker does not know the model architecture). The proposed model is robust against both types of attacks, as shown in section 5. However, those evaluations are based on the MNIST

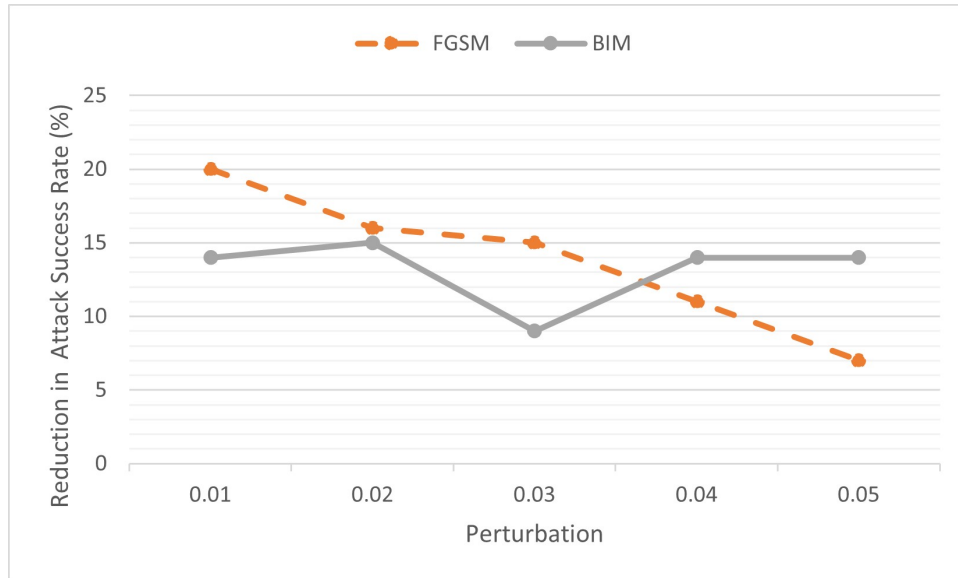


Figure 7. Reduction in Attack Success Rate (%) as a Function of Perturbation Injected by the FGSM and BIM Attacks, Respectively. Reprinted from [88]. © 2022. Used with permission as a senior author. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

dataset. More experimentation on larger datasets are encouraged if the computational resource is available. Furthermore, it is unclear whether or not our proposed approach is still robust when crafting adversarial samples are done by someone who has no idea what toolboxes like ART are used. We will answer this question in our further studies.

In section 6.2, we discussed the training time and proposed a parallelization framework for adversarial training. Then, to show the importance of selecting the right resource, we experimented on different types of GPUs. In section 6.3, we discussed the idea of ensembling and performed an experiment to show the effectiveness of random noise in reducing the attack success rate on the original model (without adversarial training) under FGSM and BIM attacks. Note that all attacks are generated based on the Adversarial Robustness Toolbox. However, similar results can be obtained using other toolboxes such as Foolbox [119] or advtorch [40]. Furthermore, it is encouraged to test the proposed defense method on a tailored adaptive attack that might generate successful perturbations on randomly perturbed images instead of clean images.

3.7 Conclusions and Future Work

Many recent researchers have utilized various machine learning techniques, such as DNN, for security-critical applications. For instance, Morgulis *et al.* [103] have demonstrated that traffic sign recognition systems of a car can be easily fooled by adversarial traffic signs and cause the car to take unwanted actions. This result has shown the vulnerability of machine learning techniques. In this work, we have presented a distributed adversarial retraining approach against adversarial examples. The proposed methodology is based on the idea that with enough datasets, sufficient complex neural network, and computational resource, we can obtain a DNN model that is robust against these adversarial attacks. The proposed approach is different from the existing adversarial retraining approach in several aspects. First, we have used soft label instead of hard label to prevent overfitting. Secondly, we have proposed to increase model complexity to overcome the tradeoff between the model accuracy and the resilience against adversarial examples. Furthermore, we have utilized the transferability property of adversarial instance to develop a distributive adversarial retraining framework that can save runtime when multiple GPUs are available. In addition, we have robustly trained the DNN model against random noise. Therefore, the obtained final classifier can provide the correct labeling to a normal instance, even though that the instance has random Gaussian noise added. By utilizing this robustness against random noise, we have added random noise to all test instances before performing classification in order to break the careful calculated adversarial perturbation. Moreover, we have compared our proposed approach against the current start-of-the-art approach and demonstrated that our proposed approach can effectively defend against stronger adversarial attacks, i.e., C&W attack and Deepfool. Future work will explore black-box attacks and formal guarantee for performance.

Chapter 4

Active Learning Under Malicious Mislabeling and Poisoning Attacks

Deep neural networks usually require large labeled datasets for training to achieve the start-of-the-art performance in many tasks, such as image classification and natural language processing. Though a lot of data is created each day by active Internet users through various distributed systems across the world, most of these data are unlabeled and are vulnerable to data poisoning attacks. In this chapter¹, an efficient active learning method that requires fewer labeled instances and incorporates the technique of adversarial retraining in which additional labeled artificial data are generated without increasing the labeling budget is presented. The generated adversarial examples also provide a way to measure the vulnerability of the model. To check the performance of the proposed method under an adversarial setting, i.e., malicious mislabeling and data poisoning attacks, an extensive evaluation on the reduced CIFAR-10 dataset, which contains only two classes: ‘airplane’ and ‘frog’ by using the private cloud on campus is performed. The experimental results demonstrate that the proposed active learning method is efficient for defending against malicious mislabeling and data poisoning attacks. Specifically, whereas the baseline active learning method based on the random sampling strategy performs poorly (about 50%) under a malicious mislabeling attack, the proposed active learning method can achieve the desired accuracy of 89% using only one-third of the dataset, on average.

4.1 Introduction

Despite state-of-the-art performance, deep learning models are easily fooled by data poisoning attacks and evasion attacks [155]. The main difference between data poisoning attacks and evasion attacks is the assumption of an attacker’s ability. For a triggerless clean-label data poisoning attack, an adversary tries to manipulate some training data in order to change a decision boundary and cause the misclassification of a

¹Portions of this chapter have been previously published by the author of this dissertation in [85], where no formal reuse license is needed for the IEEE © [2021] publication [85].

specified test instance on a targeted attack. Contrarily, an adversary tries to generate an adversarial example by introducing a subtle perturbation to a specified test instance in order to mislead a learning algorithm and evade detection in an evasion attack. In this dissertation, we propose an effective active learning method that mitigates the effect of data poisoning attacks by utilizing adversarial examples.

4.2 Threat Model

There is a large amount of data generated every day. However, these data need to be properly labeled before we can apply supervised machine learning algorithms. This is not always feasible with a restricted labeling budget. Crowdsourcing is not reliable as humans make mistakes also, and the adversary can intentionally provide misleading labels. Furthermore, the data obtained could be poisoned. In this research, we consider the adversarial setting by assuming that

- it is not feasible for an individual attacker to control a significant portion of the training data. However, an attacker knows the model and its parameters and can inject a poison instance into the training set. The poisoning attack is defined in equation 4.2.
- there is some probability p_r that a human labeler makes a mistake when annotating.
- a malicious labeler will consistently give a biased label for a target class of the instance. For instance, consider the MNIST data set of handwritten digits and a target class of handwritten numbers ‘1’. A malicious labeler can consistently mislabel ‘1’ as ‘7’ for each query for labeling that involves handwritten numbers ‘1’ whereas providing correct labels for other classes. As shown in [44], this type of biased mislabeling is more severe than the unbiased mislabeling considered in other papers.

Different from most deep learning literature, our assumption is that the class labels provided by a human labeler is not perfect. That is, we consider the effect of human errors. We assume each benign labeler mislabels a data $p_r \in (0, 1)$ of time and this mislabel is uniformly distributed. That is, it is equally likely that a class c is mislabel as any other class $\bar{c} \in C - c$, where C is the set of classes. Each independent labeler will provide the true label with probability $1 - p_r$ and the remaining p_r is evenly distributed to other class labels for the data collected in distributed systems. That is, if an instance x belongs to class c , then a human

labeler will label it

$$O(x) = \begin{cases} \bar{c} & \text{if } u < p_r \\ c & \text{if } u \geq p_r, \end{cases}$$

where u is a uniform random number between 0 and 1 and $\bar{c} \in C - \{c\}$ is any class but c . However, under a malicious mislabeling attack, an attacker consistently mislabels the instances of a target class c_t as if it belongs to a benign class $c'_t \in C - \{c_t\}$. For instance, considering the target class ‘spam email’ and the benign class ‘normal email’ for a spam detection model, an adversary may mislead the spam detector by mislabeling a ‘spam email’ as a ‘normal email.’ As shown in [44], the biased mislabeling usually degrades the classification accuracy of the target class severely but has little effect on the non-target classes.

Furthermore, we also consider a data poisoning attack. In general, the attacker’s objective is to inject a data poisoning instance (p, y) such that

$$p = \operatorname{argmax}_{p \in D_i} \mathbf{C}(F(\theta_i^*), D_t), \quad (4.1)$$

where \mathbf{C} is a cost function that measures the performance of a model F on a test set D_t and the parameters θ_i^* of the model F at iteration i is determined by minimizing the loss function L used to train a poisoned training set D_i as follows:

$$\theta_i^* = \operatorname{argmin}_{\theta_i} L(F(\theta_i, D_i)).$$

The poison instance p can be generated by solving Equation 4.1 based on [127]. The targeted poisoning attack (also known as feature collision attack) proposed in [127] can achieve a 100% attack success rate for transfer learning models and 53% for end-to-end learning models. The targeted poisoning attack assumes an attacker has no knowledge of the training data but has knowledge of the model architecture and its parameters. In this research, we extend this poisoning attack to an active learning setting, where the attacker can inject poison instances to the data pool as well as knows the model architecture and its parameters. Specifically, the attacker can generate a poison instance p by solving the following optimization problem:

$$p = \operatorname{argmin}_x \|f_i(x) - f_i(t)\|_2^2 + \beta \|x - b\|_2^2, \quad (4.2)$$

where f_i is the output of the penultimate layer of a network obtained at iteration i of the active learning process, $\beta \in [0, 1]$ is a similarity parameter that indicates the relative importance of the first component $\|f_i(x) - f_i(t)\|_2^2$ to the second component $\|x - b\|_2^2$, t is the target instance, and b is a base instance.

Consider spam detection task again. A target instance t would be a spam email, and a base instance b is a benign email. The goal of an attacker is to have a spam email detector to misclassify a targeted spam email as a benign email. An attacker tries to craft a poison instance by adding small adversarial perturbation that is not noticeable to a human labeler (i.e., $\|x - b\|_2^2$ is small). Meanwhile, the generated poison instance looks like the target instance in the feature space defined by f_i (i.e., $\|f_i(x) - f_i(t)\|_2^2$ is small). That is, a poison instance looks like a base instance to a human, so a human labeler labels that poison instance as a benign. However, a poison instance looks like a target instance for a classifier (close in feature space defined by f_i). While the network tries to adjust the decision boundary so that they can classify the poison instance as a benign, the ultimate effect is that the target instance is classified as a benign email at the test time as well since the feature representation of the target instance is similar to the feature representation of the poison instance. In this way, the classifier gets misled into classifying a target spam email as a benign email.

Notation

The commonly used notation for this chapter is shown in Table 1. The subscript i indicates the i -th iteration. For instance, F_i indicates the neural network model obtained at i -th iteration of active learning. Furthermore, $|*|$ indicates the cardinality of a set $*$. For instance, $|C|$ indicates the total number of classes as C is the set of class labels. Last, $\|x\|_2^2$ represents L_2 -norm of x .

Table 6: Notation.

Symbol	Description
p	a poison instance
b	a base instance
t	a target instance
x	an instance
S	a secured set of labeled data. It is secured and the adversary has no knowledge of it.
S'	a newly selected unlabeled dataset that is sending to human labelers for labeling

Table 6: Notation (Continued)

Symbol	Description
U	unlabeled training set
D	labeled training set
D_t	labeled test set
q	an application-specific parameter that depends on the dynamic of the dataset
m	the size of the dataset used in the initial training stage
n	the number of clusters
N_c	the number of adversarial examples generated for center c
C	the set of class labels
τ	a threshold that determines whether a minimal perturbation is small enough to stop active learning
ϵ	a threshold that determines whether to further examine the newly selected dataset S'
a	an amount of reduction in the trustworthy of O
$O(x)$	the label provided by an oracle/human labeler for an instance x .
F	the neural network model
f	the output of the penultimate layer of a network
p_r	the probability that an independent labeler mislabel a data point unintentionally
c	the true class an instance belongs to
\bar{c}	an misleading class $\bar{c} \in C - c$
c_t	targeted class
\mathbf{C}	a cost function that measures the performance of a model F on a test set D_t
θ_i^*	weight parameters of a neural network model F at iteration i
β	a similarity parameter for optimization problem (2)
q	the number of iterations passed before the secured set S is updated again
$d(*, *)$	a distance metric in the feature space

4.3 Related Work

This section provides a brief description of existing active learning methods. Though gathering large quantities of unlabeled data is easy and cheap, obtaining the label for them is usually time-consuming and expensive. For instance, there are more text data available than what a researcher with a limited labeling budget can afford. Active learning reduces the number of labeled instances needed for training a model effectively. See Fig. 8 for an overview of the general active learning framework. Various active learning algorithms have developed over the years. They can be categorized as uncertainty-based active learning, diversity-based active learning, and adversarial active learning.

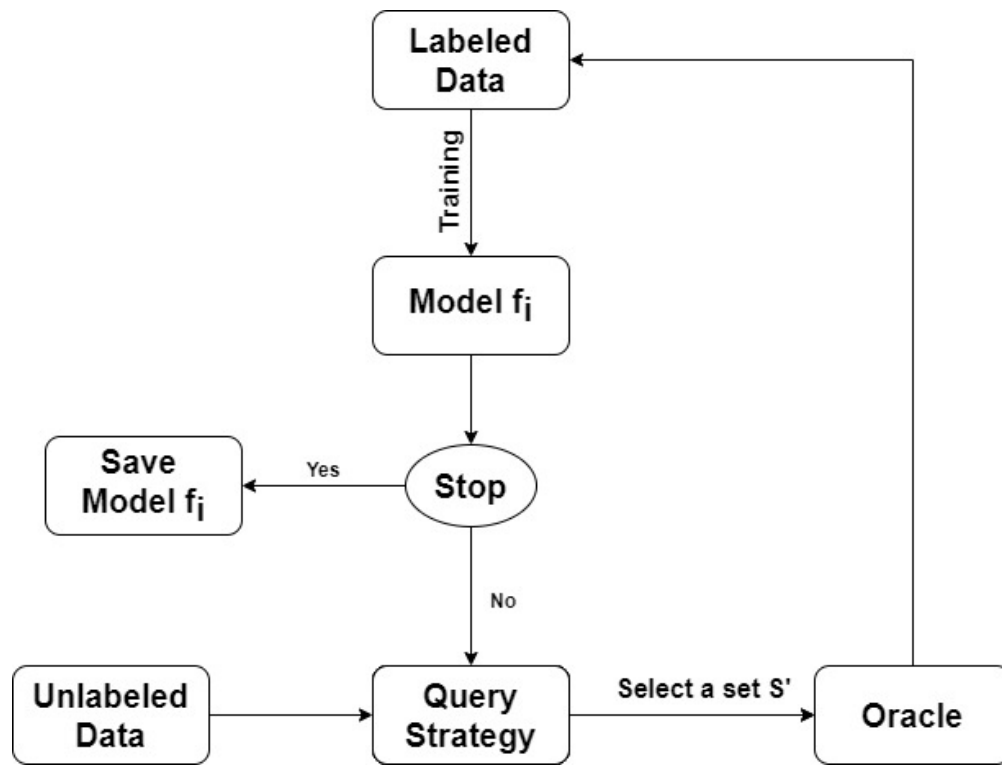


Figure 8. Active Learning Framework

The uncertainty based active learning algorithms focus on selecting the instances based on the uncertainty principle [107], [126], [71]. The uncertainty principle is the idea that an training instance that is most uncertain to a model contains the most new information for improving the model. However, recent work on data poisoning attack, e.g., [41], [142], [159], has shown that uncertainty sampling may help at-

tackers in achieving their goals. As the poisoning instances are usually laid around the decision boundary, these poison instances are more likely to be selected by the uncertainty sampling approaches.

Another class of active learning algorithms focuses on selecting the most diverse and representative dataset. For instance, Yang et al. [132] proposed an optimization-based active learning algorithm that imposed a diversity constraint objective function. The core-Set approach [125] is a diversity-based approach using core-set selection. Batch Active learning by Diverse Gradient Embeddings (BADGE) algorithm [7] selects a subset of instances whose gradients with respect to the parameters in the output layer are most diverse.

The adversarial active learning algorithms, e.g., [160] and [95], are a relatively new idea and are usually based on the Generative Adversarial Networks (GAN) [50], Variational Auto-Encoders (VAE) [77], or adversarial examples (AE) [133], [131], [52], [101]. The first active learning work using GAN is proposed by Zhu et al. [160]. GAN enables the synthesis of additional instances from the labeled training instances. However, its performance is not as good as active learning based on the random sampling strategy. The first active learning algorithm for multi-classification using GAN is Adversarial Sampling for Active Learning [95].

4.4 Methodology

Fig. 9 shows the framework of our proposed active learning method. Four main components of this framework are training (1), performance checking (2), querying (3), and artificial data crafting (4). The training step is the same as regular training on the labeled dataset. Initially, a set of m data points D is randomly selected to query for labels. Then, the model is trained using the labeled dataset D . To check the performance of the model and the existence of malicious instances, a representative subset S of data with labels is set aside and secured so that adversary does not know of it. This set S is updated every q iterations to make sure it represents the diversity of the current dataset, where q is an application-specific parameter that depends on the dynamic of the dataset. If the underlying distribution of the data does not change over time and security of it is not in question, $q = \infty$ and no update is needed.

Besides, for each queried instance, we can generate both adversarial examples and data augmentation. In this way, the training set is increased without incurring an extra query budget. The various adversarial crafting techniques and data argumentation techniques are useful in this case. For each center c , N_c instances are selected. There is a total of $\sum_{c=1}^n N_c$ data points S' . Then, a group of human labelers

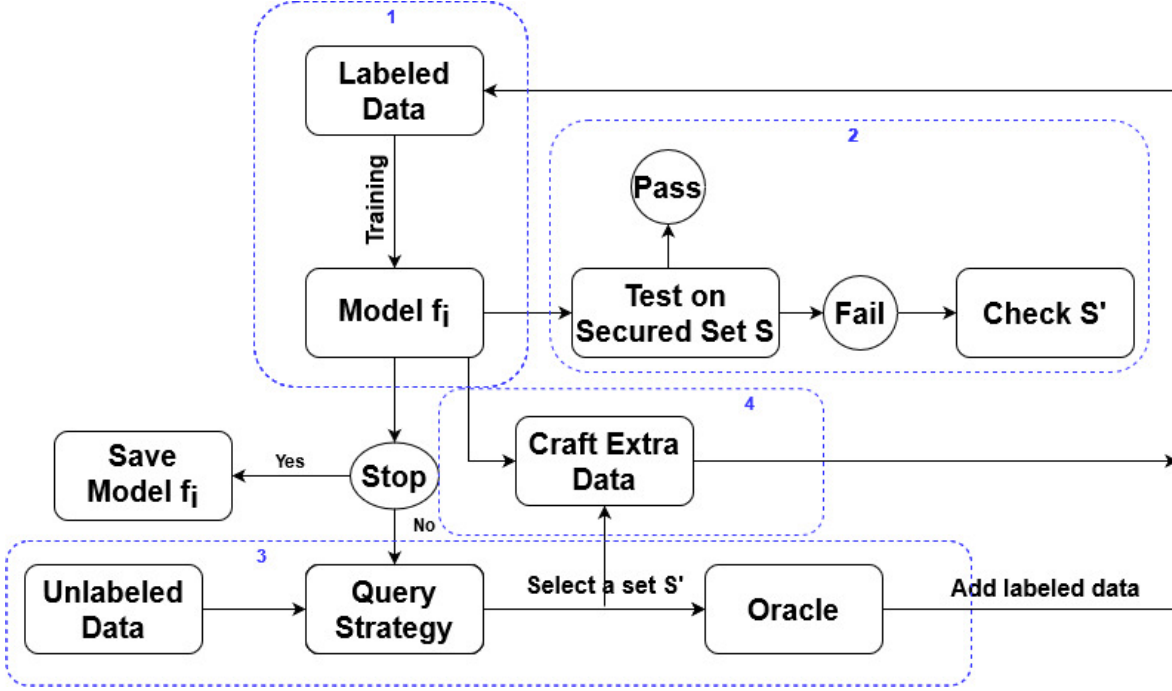


Figure 9. The Proposed Active Learning Framework. Reprinted from [85] © 2021 IEEE. Used with permission as a senior author.

provides the labels for these data points. Next, this newly labeled dataset

$$D' = \{(x, O(x)) | x \in S'\},$$

where $O(x)$ is the label provided by an oracle/human labeler for an instance x , and D' is added to the previously obtained labeled dataset D ; i.e.,

$$D = D \cup D'.$$

Next, we train the model F_i on the new dataset D . The performance of the updated model is tested on a secured set S . If the performance degrades by a threshold ϵ , the newly obtained dataset D' is examined by h ($h > 0$) independent labelers to see if any instance is poisoned or mislabeled. The probability P_h that at least one labeler out of h ($h > 0$) independent labelers provides correct annotation is

$$P_h = 1 - (1 - P_1)^h, \quad (4.3)$$

where P_1 is the average performance of a human labeler. To find the number h of independent labelers to ensure that P_h is large enough, saying 0.9995, we solve the equation 4.3 and obtain

$$h = \left\lceil \frac{\log(1 - P_h)}{\log(1 - P_1)} \right\rceil \quad (4.4)$$

For instance, [116] showed the human annotation performance for the CIFAR10 dataset is 93.91%. Hence, three independent labelers are enough to ensure at least one labeler provides correct annotation.

If the performance does not degrade more than ϵ , the model is accepted and the stopping criteria are checked. If the stopping criterion is met, the process stops. Otherwise, the process repeats. The stop criteria can be

- the minimal perturbation of all adversarial examples is larger than a threshold τ ,
- the desired classification accuracy is reached, or
- the labeling budget is exhausted.

In the following sections, we provide the details for performance checking, querying, and artificial data crafting.

4.4.1 Performance Analysis

The ultimate goal of malicious labeling attacks is to reduce the accuracy of a model. Hence, the best way to detect it is by monitoring the classification accuracy on a secured subset S . If the performance (i.e., classification accuracy) of a newly trained model F_i on S drops over a threshold ϵ compared to F_{i-1} , then the newly queried dataset used to train F_i is likely malicious. They need to be further examined. Otherwise, accept F_i .

For further examination, three independent labelers are assigned to relabel the newly queried data point(s). For each data point, if there is any inconsistency in the label, the data point is discarded. However, if the consistent label is provided and it is different from the original label provided by the original labeler O , then the confidence in O 's label is reduced (Fig. 10). The degree of reduction in confidence depends on whether F_i provides high confidence or low-confidence misclassification. If F_i provides a low-confidence value for this misclassification, then it may be a mild drift and we should reduce the confidence in O 's label by a amount, where a is inversely proportional to the confidence value provided by F_i . Nevertheless, If F_i

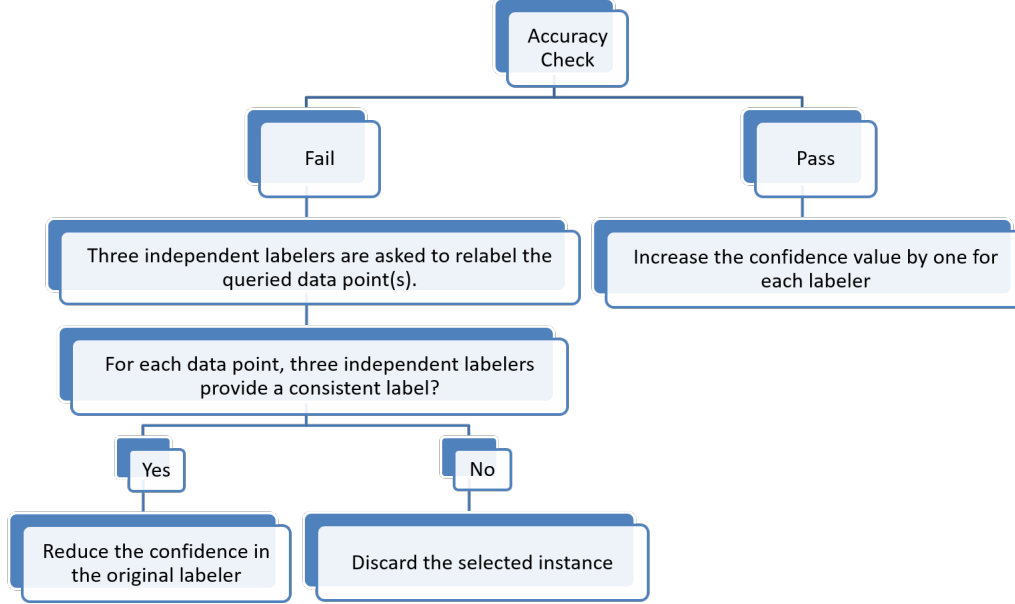


Figure 10. Confidence Score Updating. Reprinted from [85] © 2021 IEEE. Used with permission as a senior author.

provides a high-confidence value for this misclassification, then we should be alert and the trustworthiness of O needs to be checked immediately.

4.4.2 Querying Strategy

The proposed querying strategy looks for unlabeled instances that are different from the most labeled ones. In order to do that, a cluster algorithm, such as k-mean, k-medoids (PAM), and hierarchical clustering, can be used to find n centers. n can be determined by various methods such as elbow, silhouette, or gap statistic methods. For each center c , N_c unlabeled instances whose corresponding distance (in feature space) from the currently labeled instances are the largest are selected; i.e., select $x \in A$ such that solves the following optimization problem:

$$\max_{x \in A_c} \min_{y \in D} d(x, y), \quad (4.5)$$

where A_c is the collection of unlabeled instances that belong to the cluster c , and d is a distance metric in the feature space. This ensures the diverse instances being selected.

The number of instances N_c being selected at each iteration depends on

- the average perturbation of adversarial examples generated in the previous iteration,

- the performance of the model on the given cluster, and
- the labeling budget.

4.4.3 Artificial Data Crafting

For the past few years, a lot of attention is given to the adversarial example. A popular example of an adversarial example in security is adversarial eyeglass frames against deep neural network-based face detection systems [128].

Adversarial examples and poison instances generated by the feature collision attack [127] are similar in the sense that they are both indistinguishable to the original data. For instance, Fig. 11 shows an example of a data poisoning attack. An attacker tries to mislead an image classifier to misclassify the targeted frog image as an airplane. In order to do it, the attacker selects a random image of an airplane as a base instance. Then, using a forward-backward-splitting iterative procedure [49], the attacker solves equation 4.2 to obtain the poison instance shown in Fig. 11. Clearly, this poison instance looks indistinguishable from the base instance. However, the poison instance looks like the targeted instance in the feature space based on solving the optimization given in equation 4.2. Hence, by injecting this poison instance into the training set, the trained classifier misclassifies the targeted instance as an “airplane” due to the similarity in the feature space.



Figure 11. An Example of Targeted Poisoning Attacks

To check the reason behind such misclassification and the success rate of such a poisoning attack, we develop a binary classifier based on [127]. We find that when the number of training instances is small, the success rate of the poisoning attacks is high, just like it is claimed in the paper [127]. However, when the number of training instances is large, the success rate drops significantly. Furthermore, the poisoning attack

on a network with end-to-end training is much harder than an attack on a network with transfer learning [127]. This indicates that the increase in the training set can be beneficial. However, a bigger dataset requires a larger labeling budget.

To address the issue, we take advantage of the “indistinguishable property” of adversarial examples. We utilize various adversarial crafting techniques to generate extra data for training. Using the labeled instances, we can generate additional instances that are similar to these labeled instances without incurring an extra labeling budget. Since the minimal adversarial perturbation obtained through adversarial crafting techniques can misclassify these instances, they point at the vulnerable area of the classifier and are a great place to get more data points for the next iteration of the active learning.

For each center c , N_c instances selected using equation 4.5 are utilized to generate adversarial examples. An adversarial example can be generated using multiple existing techniques such as DeepFool, C&W, and BIM. PGD, DeepFool, and C&W attacks are good choices because they are the strongest attacks that currently exist, and the generated perturbation is a good measurement of the performance of the trained model. Besides, the artificial instance can also be generated using Generative Adversarial Network (GAN). If the dataset is images, data augmentation such as rotation, scaling, and change colors, can be used as well. Furthermore, another simple way to obtain additional noise samples is by adding random noise to the N_c selected instances. The noise added should be subtle so that labels are not changed. These noise samples can be used in addition to the adversarial examples, as adding noises is usually less time consuming than generating adversarial examples.

Real-world large data may be often collected in different location, so a distributed system should be a right candidate for our proposed methodology in this case. Specially, the distributed system consists of multiple computing nodes that are distributed at different locations, where the proposed active learning framework is used to train the data collected in one location and an orchestrator is implemented to automate the management, coordination and organization of the proposed active learning in these nodes of the distributed computing systems. The orchestration supports the efficient delivery of distributed computing resources for the active learning under malicious mislabeling and poisoning attacks.

4.5 Evaluation

We empirically evaluate the proposed defense method on a subset of the CIFAR-10 [79] dataset. The baseline model for comparison is active learning based on random selection [153]. Our experiments

were conducted in the high-performance computing system within the campus private cloud. In the set of experiments, we wish to address the following:

1. Does our proposed method perform as good as the standard active learning method when there is no attack? We try to address this question as a few papers (e.g., [57], [91], [4], [110]) have shown the trade-off between the performance of the active learning algorithms with attacks and their performance without attacks.
2. Is our proposed method effective under a malicious mislabeling attack? and
3. Is our proposed method effective under a poison attack? That is, are certain vulnerabilities (e.g., to specific inputs) learned as we perform the adversarial active retraining?

The evaluation metric for question 1) is the labeling budget used to achieve desired classification accuracy. The evaluation metrics for question 2) are the overall classification accuracy on the clean test data and the required labeling budget to achieve desired classification accuracy. The evaluation metrics for the last question are the overall classification accuracy on the clean test data, the poisoning attack success rate, and the required labeling budget to achieve desired classification accuracy.

4.5.1 Dataset and Model

The CIFAR-10 dataset is an object classification dataset that consists of ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image is a 32×32 tiny color image and each pixel has three values corresponding to the RGB intensities. The samples are split between a training set of 50,000 with 5,000 samples for each class and a testing set of 10,000. Table II summarizes the dataset. Real-world data may be collected through a computing system whose devices are distributed in different

Table 7 CIFAR-10 vs. CIFAR-2

Dataset	CIFAR-10	CIFAR-2
# Training Example	50,000	3,000
# Testing Example	10,000	2,000
# Classes	10	2
Image Size	32×32	32×32

location. However, due to the limited computation resource and the slowness of the data poisoning attack

for the DNN model proposed in [127], we reduce the CIFAR-10 dataset to the CIFAR-2 dataset, only considering two classes, airplane and frog. To further speed up the data poisoning attack, we reduce the training set to 3,000 training instances instead of 10,000 samples. As you can see in the evaluation section, the proposed method does not even need to have 3,000 training instances to reach the desired accuracy. Thus, this limitation speeds up the evaluation time without affecting the performance. The testing set is 2,000, and each testing instance is considered as a targeted instance during the poison instance generation process.

The network architectures for the CIFAR-10 dataset is based on transfer learning. More precisely, ResNet50 [60] is used for feature extraction, and then a dropout rate of 0.2 is applied to it. Last, a fully connected layer with the sigmoid activation function is used for binary classification.

4.5.2 Artificial Data Generation and Data Poisoning Attack

Adversarial examples and poison instances are generated using the Adversarial Robustness Toolbox (ART v1.3) [108]. C&W, PGD, and DeepFool attacks are used to generate adversarial images in addition to Gaussian noise. The PGD attack is selected over others because the PGD attack generates the strongest first order adversary [94]. However, due to the fact that the PGD attack implemented in ART does not take advantage of GPU power, the runtime for the PGD attack is much higher than the C&W attack and so the PGD attack is not included during the training. The maximum number of iterations for the C&W attack and DeepFool are 20 and 100 (default in ART), respectively. Random noise added to train image follows a Gaussian distribution with mean 0 and variance 0.3 for each batch. This is a reasonable limit on what is permissible because a larger perturbation would distort the image too much.

Assume that the adversary knows the model architecture and can inject poison instances. We generate the data poisoning attack based on [127] to evaluate the robustness of the proposed active learning method. This is the only data poisoning attack on DNN model available in ART. We consider each test instance as a targeted instance. For each targeted poisoning attack, we use first 50 training instances from a difference class than the targeted instance as base instances for generating the poison instances. A small amount of watermark (0.1-0.3) is added to increase the attack success rate while the poison instances are indistinguishable from their corresponding base instances. The exact amount of the watermarks added depends on the difficulty of each targeted attack scenario. After a set of poison instances are generated for each targeted attack, we check whether the attack is successful. If a targeted poisoning attack is not successful, the corresponding poison instances are discarded. In short, we only save the set of poisoning instances that

an attacker successfully attacks the baseline model trained with all training instances and the poison instances. In the end, we generate 67 sets of successful data poisoning attacks, which can be used to evaluate the performance of the proposed active learning method.

4.5.3 Mislabeling

For most deep learning tasks, we assume that the labels provided are true (correct). However, in the real world, humans make mistakes and adversaries can inject malicious labeling [114, 135]. In this paper, we consider both cases. First, we assume a human labeler has a 5% chance to make a mistake when annotating. The 5% is used since the dataset used is tiny, low-resolution images that are difficult to classify correctly sometimes. Furthermore, with a total of eleven labelers for the experiments, one of them is malicious. More precisely, a malicious labeler will intentionally misclassify an image in the ‘airplane’ class as the ‘frog’ class.

4.5.4 Results

In this section, we evaluate the performance of the proposed active learning method by addressing the three questions posed at the beginning of the evaluation section.

4.5.4.1 Does our proposed active learning method perform as good as the standard active learning method when not under an attack?

Table 8 Statistics of Labeling Budgets. Reprinted from Lin et al. [2021b] © 2021 IEEE. Used with permission.

Model	Baseline	Proposed
Mean	1186	572.3
Standard Deviation	182.6	51.7
Minimum	608	462
25%	1128	533
50%	1184	563
75%	1272	610
Maximum	1536	672

Here, we consider only the benign mislabeling caused by human errors. The prediction accuracy of the classifier trained with the full training set is 89%. Using it as the desired accuracy, we find the labeling

budget required to achieve such accuracy. We compare our proposed active learning method with the active learning based on a random sampling strategy (as baseline). The experiment is repeated 30 times, and the summary statistics are provided in Table III. As shown in the table, the proposed active learning method only needs about 48% of the labeling budget required by the baseline active learning method when there are no attacks. Furthermore, the variation in performance is much lower for our proposed method compared to the baseline model. The maximum number of labeling budgets required to achieve the desired accuracy is halved if using our proposed method.

4.5.4.2 Is our defense method effective under a malicious mislabeling attack?

First, we evaluate the performance of the baseline model under a malicious mislabeling attack (see subsection 4.5.3 for the implementation of malicious mislabeling attack). Similar to the previous experiment, we repeated our experiments 30 times. However, the baseline model is not able to achieve the desired accuracy in all the experiments. As shown in Fig. 12, the malicious mislabeling attack is very effective against the baseline model. The test accuracy just fluctuates between 0.44 and 0.52. Even when we use all 3,000 training instances, the test accuracy is still close to 0.5.

Nevertheless, our proposed active learning method is robust against malicious mislabeling attacks. As shown in Fig. 13, we can still obtain the desired accuracy even under a malicious labeling attack. Furthermore, Table 9 shows the summary statistics for the labeling budget required to achieve the desired accuracy. On average, we only need about one-third of the original training set size to achieve the same accuracy (89%) even under a malicious mislabeling attack.

Table 9 Summary Statistics for Labeling Budget Under a Malicious Mislabeling Attack

Model	Proposed
Mean	925.3
Standard Deviation	315.9
Minimum	580
25%	701.5
50%	875
75%	1013
Maximum	1939

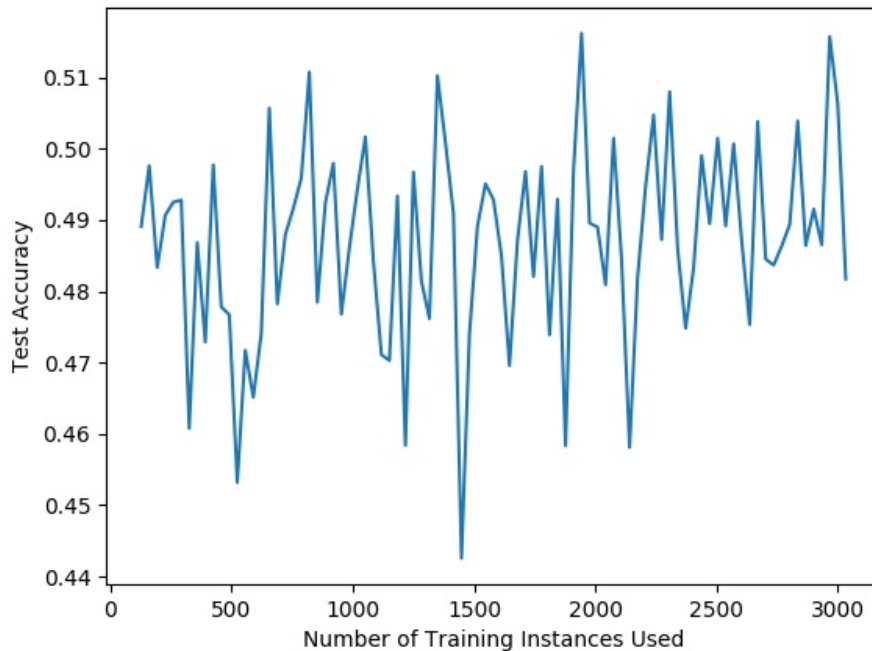


Figure 12. Baseline Model Performance Under a Malicious Labeling Attack. Reprinted from Lin et al. [2021b] © 2021 IEEE. Used with permission as a senior author.

4.5.4.3 Is our proposed method effective under a poison attack?

Here, we consider a targeted clean label data poisoning [127] and evaluate the performance of the baseline and our proposed active learning method under such an attack. The targeted clean label data poisoning attack tries to reduce the performance of the classifier on the targeted instance while maintaining the performance of the classifier on the clean data. This makes it very difficult to detect as the classification accuracy on the clean data is not changed under the attack. However, the classifier makes misclassification on the targeted instance.

Since most uncertainty-based active learning strategies select unlabeled instances that are closer to the decision boundary, this makes these uncertainty-based active learning strategies more vulnerable to the data poisoning attack. On the contrary, simple active learning based on a random sampling strategy is more robust against such an attack. Hence, we will compare our proposed active learning method with active learning based on a random sampling strategy (baseline). As shown in Fig. 14 and Fig. 15, the test accuracies of the two active learning strategies under the poisoning attack are similar though the labeling

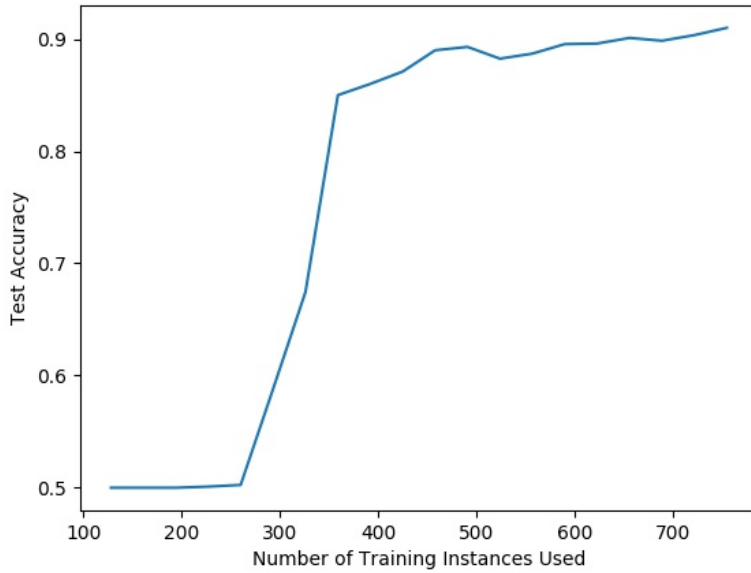


Figure 13. Robust Active Learning Model Performance Under a Malicious Labeling Attack

budget required is halved if our proposed method is used. Furthermore, the proposed method has a much lower variation in labeling budgets than the baseline method.

Table 10 Summary Statistics for Labeling Budget Under a Poison Attack

Model	Baseline	Proposed
Mean	1221	622
Standard Deviation	603	88.9
Minimum	672	484
25%	896	574
50%	976	599.5
75%	1216	698
Maximum	2816	751

4.5.5 Computation Cost and Limitation

The artificial crafting step lowers the labeling budget by generating labeled data without labelers, but it also requires additional computation cost. For instance, it takes about 0.19 seconds to generate an adversarial image using a C&W method (with a batch size of 64) on an NVIDIA Tesla K40c GPU. However,

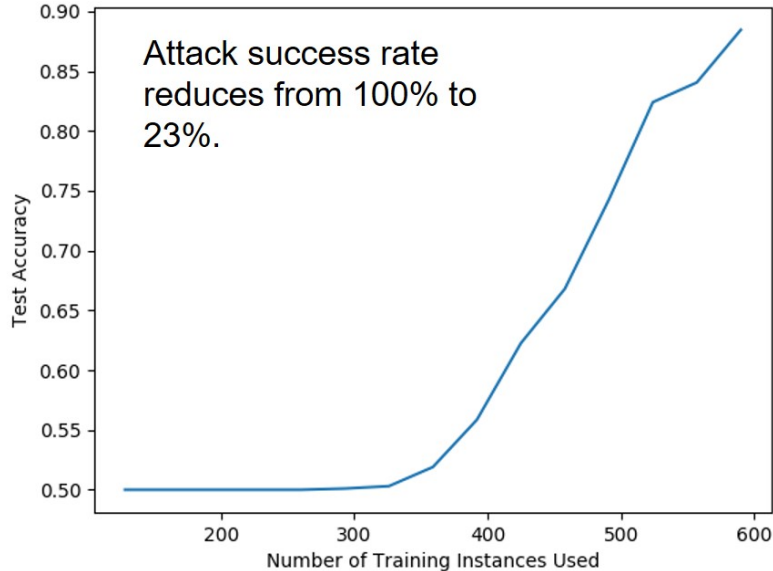


Figure 14. Baseline Model Performance Under a Poisoning Attack. Reprinted from Lin et al. [2021b] © 2021 IEEE. Used with permission as a senior author.

it takes less than 0.1 seconds to generate an adversarial image using a C&W method (with a batch size of 512) on the same computer with an NVIDIA TITAN V GPU. Hence, the type of GPU and batch size can affect the computation time. Furthermore, different methods may require different computation resources. For instance, BIM attacks mainly use CPU resources whereas a C&W method mainly uses GPU resources based on IBM’s adversarial robustness toolbox [108]. For the evaluation, we generate an artificial image for each corresponding newly queried image. However, this ratio can be adjusted based on the computational resource and the labeling budget limitation. Furthermore, the evaluation is based on a single dataset. It would be interesting to check the effectiveness of the approach on other datasets, which will be in our further study.

4.6 Conclusions and Future Work

Recent work has shown that machine learning models, such as DNNs, are highly vulnerable to various attacks, such as malicious mislabeling attack, adversarial examples, and data poisoning attacks. In this work, we utilized these adversarial examples to generate artificial data for training without incurring an extra labeling budget. Furthermore, we proposed a confidence score updating system to check the trust-

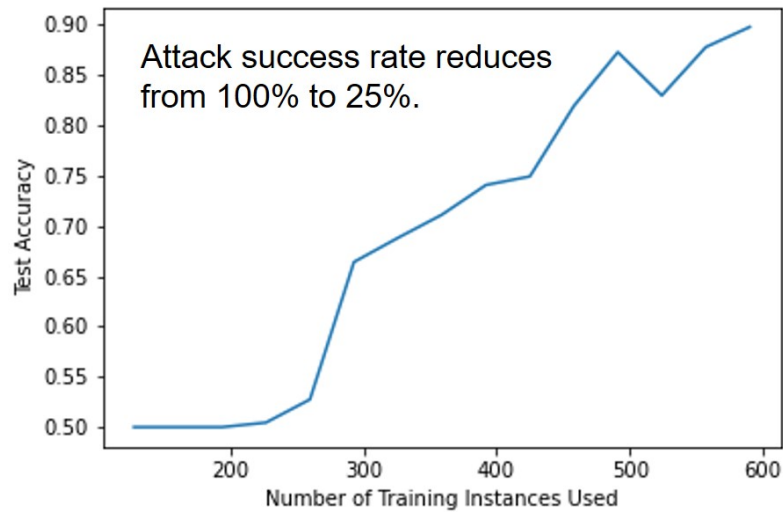


Figure 15. Robust Active Learning Model Performance Under a Poisoning Attack. Reprinted from Lin et al. [2021b] © 2021 IEEE. Used with permission as a senior author.

worthiness of each labeler and a performance check on a secured set to monitor the potential mislabeling attack. The experiment results showed that our proposed active learning method is robust against malicious mislabeling and data poisoning attacks. Though we only evaluated the performance of the proposed active learning method on binary classification in the campus private cloud, the method can be applied to a multi-classification task in a distributed system of a public cloud as well.

Although most classic single-query active learning strategies do not work as well as a random sampling strategy on deep neural network and many uncertainty-based active learning strategies also perform poorly under data poisoning attacks, it would be interesting to compare the proposed method with the state-of-the-art batch active learning method for CNNs in the future. Furthermore, our evaluation is based on a single dataset. It would be interesting to check the effectiveness of the method on other datasets. Last, the artificial crafting step introduces additional computation cost. Hence, future work will also investigate the scalability of the proposed method. We can parallelize the process and utilize the scalability in cloud computing for this purpose. Besides a private cloud, we can also deploy our proposed method on the public cloud to not only achieve scalability but also to increase reliability and flexibility.

Chapter 5

An Adversarial Attack Method for Poisoning a Transfer Learning Model

Despite the wide-ranging applicability of machine learning, they are vulnerable to security attacks, such as evasion attacks and triggerless data poisoning attacks. An evasion attack occurs at the inference time when an attacker feeds in an adversarial example, a malicious perturbed input that looks the same as its untampered copy to a human oracle, that fools a state-of-the-art classifier. In contrast, a triggerless data poisoning attack occurs at training time. An attacker tries to subvert learning with injected poisoned instances. Moreover, an attacker is allowed to manipulate the test instance in an evasion attack but not in a triggerless clean-label data poisoning attack. In this chapter¹, we focus on a special sub-category of data poisoning attacks, namely triggerless clean-label targeted data poisoning attacks. This type of attacks is more realistic in the sense that it does not require an attacker to have the ability to change the label of any training instance. That is, attackers can successfully attack the training process with a poison instance that is correctly labeled by a human oracle. We propose a simple but effective way to alter an adversarial attack method into a triggerless clean-label targeted data poisoning attack method with a remarkable attack success rate of up to 100%. Furthermore, our proposed method only requires the injection of a single poison instance in order to manipulate a transfer learning model to misclassify an untampered targeted instance. We compare our method with a popular one-shot attack and show that our method is easier to use as we do not need to tune for a hyperparameter such as a similarity coefficient.

5.1 Introduction

The state-of-the-art performance of Deep Neural Networks (DNNs) mainly drives by the use of large labeled datasets and GPUs. Recently, the idea of transfer learning is utilized often for image classification and natural language process tasks with a limited labeled dataset. However, similar to deep learning models, they are vulnerable to security attacks, such as evasion attacks and data poisoning attacks.

¹Received and approved for public release by AFRL on 26 October 2021, case number: AFRL-2021-3751. Portions of this chapter will be published by the author of this dissertation in [86].

An evasion attack occurs when an adversarial example, an artificial instance generated by injecting a small indistinguishable perturbation to its natural copy, is fed into a trained model at test time. Though the generated adversarial example looks just like its genuine copy, a trained classifier mislabels the adversarial example while correctly classifying its genuine copy. The mislabeling can be problematic for security- and/or safety-critical applications. For instance, an attacker can make a small unnoticeable change to the speed limit sign of highways and trick a self-driving car into believing the speed limit sign is a stop sign. This trick may cause the self-driving car to brake suddenly on the road and cause a car accident.

In this research, we develop a simple but effective way to poison a transfer learning model with just a single poison instance by altering the use of an evasion attack method. Specifically, we utilize an evasion attack method, called feature adversaries (FA), to generate a data poisoning instance for a triggerless clean-label targeted data poisoning attack against a transfer learning model that is pre-trained on a feature extractor f (e.g., ResNet-50 [60], VGG-16 [129], and Inception-v3 [134]). The triggerless clean-label targeted data poisoning attack is stealthy and more powerful as it does not require an attacker to have control over the labeling process and only one poison instance is needed per targeted attack. Furthermore, since there is no need to control the labeling process, the poison instance can be uploaded to the Internet and then wait for a web scraping bot to retrieve it. Lastly, since this targeted data poisoning attack maintains the overall model performance (i.e., it only misclassifies the targeted instance), it is difficult to detect such an attack through the monitoring of the overall model accuracy compared to other types of data poisoning attacks.

The contributions of this research are:

- We propose a simple but effective way to alter an evasion attack method into a clean-label data poisoning attack method with a remarkable attack success rate of up to 100%.
- We design a set of extensive experiments to show the statistically significant three-way interaction among different datasets, pre-trained models, and training set sizes on attack success rate using Aligned Ranks Transformation ANOVA.
- We compare the FA attack with the popular one-shot attack [127] and show our method is simple to use since there is no need to tune the similarity coefficient and can incorporate different optimizers or adversarial attack methods, such as Adam [76], L-BFGS-B [133], and PGD [94].

The rest of this chapter is organized as follows. In section 5.2, we present a threat model, and in section 5.3, we discuss its related work. In section 5.4, we describe our proposed approach, In section 5.5, we present our evaluation result, followed by conclusions and future work in section 5.6.

5.2 Threat Model

In this chapter, we consider a clean-label targeted data poisoning attack on a transfer learning model. Similar to [127], we assume that an attacker knows the transfer model used for training but not the training set. The attacker’s goal is to misclassify a targeted instance without degrading the model accuracy on other instances. In this way, it can easily bypass any detection algorithms that use model performance as the key criteria for detecting the data poisoning attack. Moreover, we consider a single poison instance per targeted attack instead of multiple poison instances (often 1% of the training set) per targeted attack [2, 69, 159]. Mathematically, we try to solve an optimization problem presented in [127] to obtain a poison instance p :

$$\begin{aligned}
 p = \operatorname{argmin}_x \quad & \|f(x) - f(t)\|_2 + \beta \|x - b\|_2 \\
 \text{s. t.} \quad & x \in [0, 1]^n,
 \end{aligned} \tag{5.1}$$

where n is the dimension of input x , f is a feature extractor, $\beta \in [0, 1]$ is a similarity coefficient, t and b are a target instance and a base instance, respectively.

5.3 Related Work

In this section, we provide a brief introduction to transfer learning and data poisoning attacks.

5.3.1 Transfer Learning

Supervised learning is extensively used on various applications today, and it focuses on training a model for a specific task. For instance, a simple model with two convolutional layers can be trained for a handwritten digit classification task, whereas a VGG-16 [129] can be trained for an ImageNet classification task. However, obtaining a large number of labeled instances may be complex or impossible [136]. For instance, the ImageNet dataset has more than one million training images with 1000 classes and 100 thousand test images. It takes years to hand-label them.

What happens when a new task T_2 , related to another previously learned task T_1 , does not have enough labeled instances to achieve high performance? Transfer learning can be utilized here to transfer knowledge (i.e., features and model weights) from T_1 to T_2 . This transferred knowledge overcomes the data scarcity issue presented. Even when the labeled data exist in large quantity, we can still use the weights in re-used layers as weight initialization for the new related task T_2 to reduce training time. The idea of transfer learning was first proposed in 1995 [138]. Currently, there are many high-performing pre-trained models shared online for direct prediction tasks or transfer learning tasks. For instance, Keras [28] has 26 pre-trained models available for its users. Those models can be downloaded and used for image classification directly. Even if the new task T_2 is somewhat different from the original task that the model is trained for, these pre-trained models can still be used as a feature extractor for a new task T_2 . To act as a feature extractor, the model weights of a pretrained model are frozen (unchanged). In this chapter, we focus on this type of transfer learning mainly. For a detailed or more comprehensive introduction to transfer learning, please see [109].

According to Andrew Ng, a co-founder and head of Google Brain, “transfer learning will be the next driver of machine learning commercial success after supervised learning” [106]. The clean label data poisoning attack on a transfer learning model can affect its commercial use, especially in high stakes, safety- and security-critical applications. In this work, we will study the impact of a clean label data poisoning attack on various transfer learning models.

5.3.2 Data Poisoning Attacks

The goal of a triggerless data poisoning attack is usually to (i) degrade the model performance [14, 46, 105] or (ii) misclassify a specific instance (target attack) [2, 127, 159]. The data poisoning attack that degrades the model performance can be easily detected. Hence, we focus on the targeted data poisoning attacks. More specifically, this chapter is about the clean-label targeted data poisoning attack against a transfer learning model (using only a single poison instance per targeted attack). Shafahi et al. [127] first proposed one such data poisoning attack against the transfer learning models by solving equation (5.1) using a forward-backward-splitting iterative procedure [49]. They called it the one-shot attack as it only requires a single poison instance to misclassify a target instance.

Later, Zhu et al. [159] modified the objective function (5.1) in order to find a set of k poison instances $\{p^{(j)}\}_{j=1}^k$ from the same class (base class) for targeted transfer attacks. Based on transferability [140], they

showed that targeted misclassification can be achieved as long as the target instance’s feature vector is located within the convex polytope formed by the set of poison instances $\{p^{(j)}\}_{j=1}^k$ in feature spaces of m substitute models. That is,

$$\begin{aligned} \{p^{(j)}\}_{j=1}^k = \min_{\{c^{(i)}\}, \{p^{(j)}\}} & \frac{1}{2} \sum_{i=1}^m \frac{\|f_i(t) - \sum_{j=1}^k c_j^{(i)} f_i(p^{(j)})\|_2^2}{\|f_i(t)\|_2^2} \\ \text{s. t. } & \sum_{j=1}^k c_j^{(i)} = 1, \forall i, c_j^{(i)} \geq 0, \forall i, j, \\ & \|p^{(j)} - b^{(j)}\|_\infty \leq \epsilon, p^{(j)} \in [0, 1]^n, \forall j, \end{aligned} \quad (5.2)$$

where m is the number of feature extractors, k is the number of poison instances generated, $c_j^{(i)}$ is the convex combination coefficient for poison instance $p^{(j)}$ in the feature space generated by substitute model f_i , $b^{(j)}$ is the base image (an arbitrarily selected clean image) used to generate the j -th poison $p^{(j)}$, and t is the target instance. Similar to [127], Zhu et al. [159] (i) use the forward-backward splitting algorithm to find an optimal set of convex combination coefficients, (ii) use Adam [36] to optimize $p^{(j)}$ for a given set of convex combination coefficients, and (iii) finally clip the value of $p^{(j)}$ to ensure that it is within ϵ -ball of the base images. The steps (i)-(iii) are repeated until convergence.

Aghakhani et al. [2] proposed the Bullseye Polytope Attack that improves the scalability and transferability of the Convex Polytope Attack developed by Zhu et al. [159]. Bullseye Polytope simplifies the objective function by directly assigning $c_j^{(i)} = \frac{1}{k}$, which not only reduces the computation time significantly but also improves the performance by directly moving the target to the center of the attack zone. Furthermore, they introduce the idea of multi-draw dropout and multi-target mode to improve transferability. In this chapter, we focus on the constraint case of one poison instance per targeted attack and compare the performance of the proposed method with the one-shot attack.

5.4 Methodology

Figure 16 shows the relationship between an evasion attack and a triggerless data poisoning attack. During an evasion attack, an attack feeds an adversarial example to a DNN classifier at inference time. That classifier misclassifies the adversarial example as a ‘bird’ even though it classifies its natural copy correctly (as a ‘horse’). On the contrary, an attacker cannot inject any modified test instance into the DNN classifier

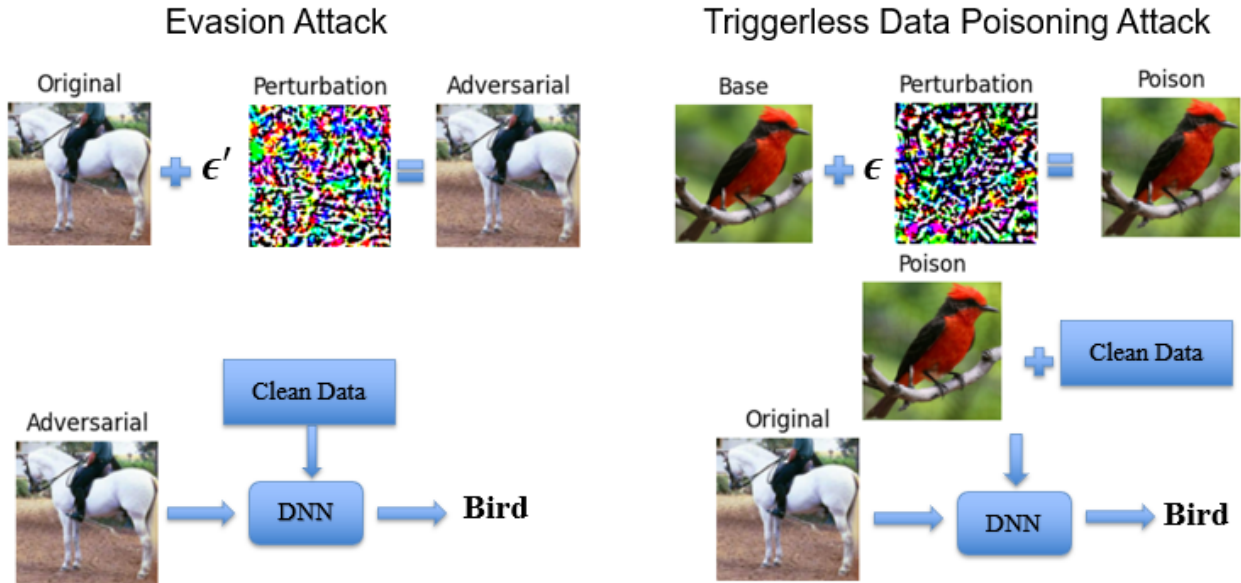


Figure 16. Evasion Attack vs. Triggerless Clean Label Data Poisoning Attack. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

at inference time for a triggerless clean-label data poisoning attack. In this case, they can still cause the targeted misclassification by misleading the DNN classifier during its training process. In this chapter, we consider a stealthy data poison attack in which the attacker has neither control over the labeling process nor the ability to inject multiple poison instances. We present a simple but efficient way to use an evasion attack method to generate a clean label data poison instance that misleads a victim model to misclassify a target instance while maintaining its overall model performance. The right side of Figure 16 shows the basic idea behind the approach. We try to generate a poison instance that looks like its natural copy (called a base instance or a source instance) to a human, but its feature representation resembles a target instance (horse). That misleads the classifier to consider the target instance (horse) and the poison instance (a bird) as if they are from the same class, although this is not the case.

Before formally describing the approach, we begin by introducing the FA attack [123]. In 2015, Sabour et al. [123] developed a novel approach to generate adversarial examples at the testing time. The generated adversarial example x appears perceptually similar to a given natural image (they called it source image s), but its deep representations (i.e., feature representations) have similar characteristics as another image (target image or guide image g) from a different class. Because of the similarity between the feature representation of the adversarial example and the feature representation of the guide image, the trained

model gives the same label for both the adversarial image and guide image, although the adversarial image is more resemble a source image (whose class is different from that of the guide image) to a human. Basically, this approach solves the following optimization problem:

$$\begin{aligned}
 \min_x \quad & \|f(x) - f(g)\|_2^2 \\
 \text{s.t.} \quad & x \in [0, 1]^n \\
 & \|x - s\|_2 < \epsilon.
 \end{aligned} \tag{5.3}$$

Table 11 Adversarial Image Generation vs. Triggerless Clean-Label Data Poisoning. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Attack Type	Evasion	Triggerless Clean-Label Data Poisoning
When to Attack	Inference Time	Training Time
$g \iff t$	An Arbitrary Guide Image That Belongs to a Class Other Than the Base Class (Class of the Base Image b)	A Target Instance That an Attacker Wants a Classifier To Misclassify
$x \iff p$	An Adversarial Example	A Poison Instance Used for Training
$s \iff b$	An Arbitrary Natural Image That an Attacker Used to Generate x or p .	

In this chapter, we use the FA attack to generate a poison instance for a stealthy triggerless clean-label data poisoning attack. We can do it since equation (5.1) and equation (5.3) are equivalent in the sense that the solutions are identical with the appropriately chosen similarity coefficient β . Specifically, Table 11 sums up this adversarial image generation method and its alternative use as a data poisoning method. Basically, the adversarial example x is injected into the training set as a poison instance p in order to misguide a trained model to misclassify a target instance (denoted as t in equation (5.1) and is equivalent to g in equation (5.3)).

To generate a poison instance for a target instance t (whose true/clean label is l), we first craft an adversarial example x by applying the FA attack to the pre-trained model f and a source image s . Then, we inject x into the training set. A human labeler will label x as y (the label of the source image s since obtained x looks the same as its natural copy s to a human). This produces a clean label for x . Next, x and its clean-label along with other natural images and their labels are used to train a transfer learning model. (Note that x is used for the training rather than as an adversarial example here.) Since the feature representation $f(x)$ of x resembles the feature representation $f(t)$ of the target t by construction, a DNN model considers that they are the same and provides the same label for t as for x . Since the poison instance x and its label y

are used during training, the trained model likely labels g as y , although its true label is $l \neq y$. Hence, the model is poisoned, and the target instance t is misclassified while only one poison instance is injected.

Originally, Sabour et al. [123] solved equation (5.3) using the Limited Memory Broyden-Fletcher-Goldfarb-Shanno method with boundary constraints (L-BFGS-B), a widely used Quasi-Newton method. However, we can use other optimization algorithms, such as gradient descent [121] and Adam [76], that do not natively support box constraints by clipping all components of x on the ϵ -ball of the base instance after each gradient step. Alternatively, we can use the project gradient descent (PGD) method as well. Hence, plenty of existing algorithms are readily applicable for solving equation (5.3) and take advantage of GPU power. This makes our method more flexible and scalable. In the next section, we will validate that this simple method is effective and demonstrates that it is easier to use than the popular one-shot attack.

5.5 Evaluation

In this section, we describe our design of experiments and analyze our experimental results. To test the performance of the clean-label data poisoning attack on the transfer learning models, we have developed various transfer learning models based on VGG-16 [129], VGG-19 [129], and ResNet-50 [60] pre-trained on ImageNet [37] for CIFAR-10 [79], STL-10 [30], CIFAR-2s (two classes: automobile vs. frog, airplane vs. frog, and bird vs. horse), and STL-2 (a reduced STL-10 dataset with only two classes: bird vs. horse). Table 12 summarizes the datasets. Furthermore, in order to see the effect of dataset size on attack success rate, we consider fourteen different dataset sizes from 500 to 7,000 with an incremental of 500. These training sets are selected from the original training set. To satisfy the ANOVA assumption of normality, we repeat for each combination of treatment levels 30 times to ensure that the normality assumption is approximately satisfied. (We will do the formal analysis to check it later.) This generates a large amount of experimental data for formal statistical analysis.

Note that we use the L_2 norm for measuring the perturbation for all experiments, and the standard test accuracy refers to the test accuracy on the clean test data. We say an attack is a success if the target instance is misclassified. The training set, base set, validation set, and test set are mutually exclusive. Unless stated otherwise, we select the first 30 test instances that are correctly classified by the unpoisoned model (standard model) as target instances. All our evaluations are based on the IBM implementation [108].

In this research, we propose a systematic evaluation of the efficiency of the FA attack by focusing on the following questions:

Table 12 Dataset Summary: CIFAR vs. STL

Dataset	CIFAR-10	CIFAR-2	STL-10	STL-2
Number of Training Instances	50,000	10,000	5,000	1,000
Number of Testing Instances	10,000	2,000	8,000	1,600
Number of Classes	10	2	10	2
Image Resolution	$32 \times 32 \times 3$		$96 \times 96 \times 3$	

- A. Shafahi et al. [127] also proposed a highly cited clean label data poisoning attack that used the forward-backward-splitting iterative procedure [49] to solve the optimization problem in (5.1). How is it compared to an FA attack?
- B. What is the effect of the dropout layer and optimizer on attack success rate?
- C. How likely does a poison instance generated from dataset D successfully attack a model built for classifying images in a different dataset $D' \neq D$ with different resolutions?
- D. Do different dataset types, dataset sizes, and pre-trained models affect attack success rate?

5.5.1 Comparison to Existing Studies

As discussed in section 5.4, our proposed data poisoning attack method solves an equivalent optimization problem as the popular one-shot attack; however, our approach is easier to use. To generate a data poison instance, we just need to specify the maximum perturbation allowed. FA attack will generate a poison instance to meet that criterion. For instance, Figure 17 is obtained by simply setting the maximum allowable perturbation δ to 0.05. (Note that we used the PGD optimizer to solve equation (5.3). As discussed in the Methodology section, the PGD optimizer utilizes the GPU resource to speed up the poison crafting process. In the next subsection, we will show that the PGD optimizer also generates the poison instance with a higher attack success rate than the original L-BFGS-B algorithm proposed in [123].)

In contrast, we have to specify the similarity coefficient to generate the poison instance and check whether the perturbation is within the maximum allowable value individually using the one-shot attack method. Table 13 shows the average, minimum, and maximum perturbations of 30 poison instances generated by the one-shot attack with a learning rate of 3 and various similarity coefficients. The similarity coefficients shown in Table 13 are obtained after we try a set of values $\{0.128, 0.25, 0.5, 1, 2, \dots, 1024\}$.

Similarly, the learning rate is obtained after we try a set of possible values $\{0.01, 0.03, 0.1, 0.3, 1, 3, 9\}$. It takes time to find the optimal range of the similarity coefficients and learning rates for the desired level of perturbation. However, even after we get the proper values for similarity coefficient and learning rate, it does not guarantee that the generated poison instances have perturbation within the maximum allowable value. For instance, even after we specify the similarity coefficient to 320 that has an average perturbation value of 0.042, the perturbation of the generated poison instances still varies from 0.031 to 0.053 depending on the target instance and base instance pair used. Hence, it is easier to use an FA attack than a one-shot attack to generate a similar quality poison instance that meets the maximum allowable perturbation limit. As an illustration, Figure 18 shows the poison instances generated by a one-shot attack with a similarity coefficient of 256 and a learning rate of 3.

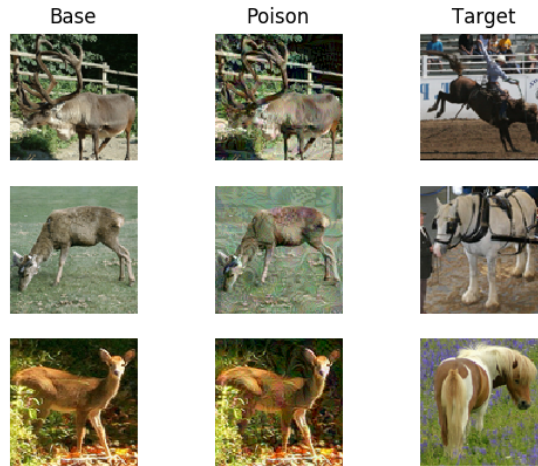


Figure 17. Poison Instances Created by the FA Attack with Maximum Allowable Perturbation Set to 0.05. The actual perturbation amounts are 0.0470, 0.0466, and 0.0474 (from row 1 to row 3).

Table 13 Similarity Coefficients. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Similarity Coefficient	Perturbation		
	Average	Minimum	Maximum
320	0.042	0.031	0.053
256	0.051	0.038	0.066
192	0.065	0.050	0.087
128	0.091	0.073	0.127

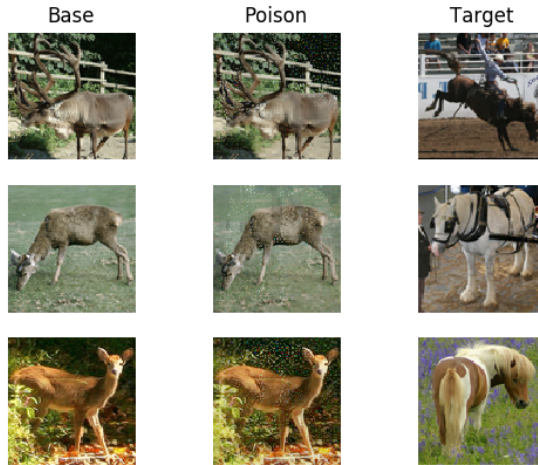


Figure 18. Poison Instances Created by the One-shot Attack with a Similarity Coefficient of 256 and a Learning Rate of 3. The average perturbation for 30 generated poison instances is 0.051. The actual perturbation values are 0.0559, 0.0458, and 0.0624 for the first three poison instances (from row 1 to row 3).

5.5.2 Effect of a Dropout Layer and Optimizer

In this subsection, we add a dropout layer before a penultimate layer to check the effect of the dropout layer with various dropout rates on the attack success rate. We consider the dropout rate of 0 (no dropout), 0.1, 0.25, and 0.5. These values are selected as they are commonly used in practice. Another usual value is 0.3. However, since it is close to 0.25, we skip it. We randomly select the airplane-frog class pair from the CIFAR-10 dataset as our prototypical example. We use the VGG-16 pretrained on ImageNet to build a classifier for this binary classification. The standard test accuracy without poisoned instances is 88%. We select the first 30 test instances that are correctly classified by the unpoisoned model as target instances. For each target instance, we craft a corresponding poison instance using either L-BFGS-B or PGD optimizer and evaluate the performance of these poison instances on victim models with various dropout rates. To make a fair comparison, we fix the $\delta = 0.1$ for both optimizers. Tables 14 and 15 show the result of this experiment. As shown, the PGD optimizer has a higher attack success rate than the original L-BFGS-B proposed in [123], and it takes less time to craft the poison instances using the PGD optimizer. This result is not surprising as the PGD attack is one of the best first-order adversarial attack methods.

Tables 14 and 15 show the target instances that are misclassified do not vary much with the dropout rate. As shown in Table 14, target instances # 5, 11, 12, and 15 are misclassified by their corresponding poisoned model (without a dropout layer) trained on the poison instance crafted using L-BFGS-B. Similarly, target instances # 11, 12, and 15 are misclassified by the corresponding poisoned model with a dropout layer. It looks like if a poison instance, which successfully misleads a victim model (without a dropout layer) to a target misclassification, likely also succeeds in fooling a model with a dropout layer to misclassify the same targeted instance disregard the dropout rate. Likewise, Table 15 shows target instances # 0, 1, 22, 25, and 28 are misclassified regardless of the value of the dropout rate, and target instances # 4, 12, 18, 26, and 29 are misclassified 3 out of 4 dropout rates.

Table 14 Effect of Dropout Rate on Attack Success Rate (L-BFGS-B). Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Dropout Rate	Attack Success on Target Instance #
0	5, 11, 12, 15
0.1	11, 12, 15
0.25	11, 12, 15
0.5	11, 12, 15

Table 15 Effect of Dropout Rate on Attack Success Rate (PGD). Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Dropout Rate	Attack Success on Target Instance #
0	0, 1, 5, 8, 11, 12, 18, 22, 25, 26, 28, 29
0.1	0, 1, 4, 11, 12, 18, 22, 24, 25, 26, 28, 29
0.25	0, 1, 4, 8, 21, 22, 24, 25, 26, 28, 29
0.5	0, 1, 4, 12, 18, 22, 25, 28

5.5.3 Effect of Upsampling and Downsampling on Attack Success Rate

In this subsection, we investigate the effect of image resolution on attack success rates. We consider two image recognition datasets, CIFAR-10 and STL-10. STL-10 has a higher resolution but few labeled training instances than CIFAR-10. We randomly select two classes, bird and horse, that are in both datasets. We consider both upsampling and downsampling of poisoned images when the victim model requires higher and lower image resolutions, respectively.

The experiment is performed using the VGG-16 model that is pretrained on either the selected CIFAR or the selected STL dataset. First, we consider the effect of upsampling (the first two rows of Table 16)

on an attack success rate. We use a CIFAR image with a resolution of $32 \times 32 \times 3$ as a source/base image for generating a poison instance (using the PGD optimizer) that targets an STL image whose resolution is $96 \times 96 \times 3$. We can generate such a poison instance either with the knowledge of the underlying pretrained model (row 2 of Table 16) or using a substitute model (row 1 of Table 16) built from the source dataset. A similar setup with the role of CIFAR and STL switch is used to study the effect of downsampling on the attack success rate. The experimental result shows that the attack success rate is higher if we are provided with the pretrained model weight for both upsampling and downsampling cases.

Table 16 Effect of Upsampling and Downsampling on Attack Success Rate

Transfer Model	Source Resolution	Target Resolution	Attack Success Rate (%)
Substitute	$32 \times 32 \times 3$	$96 \times 96 \times 3$	42
Pretrained	$32 \times 32 \times 3$	$96 \times 96 \times 3$	100
Substitute	$96 \times 96 \times 3$	$32 \times 32 \times 3$	27
Pretrained	$96 \times 96 \times 3$	$32 \times 32 \times 3$	87

5.5.4 Effect of Dataset Type, Dataset Size, and Pre-trained Models

To show the effect of dataset type, dataset size, and pre-trained models on standard test accuracy and attack success rates, we design an extensive experiment to attack transfer learning models built with the following combinations of treatment levels:

- Dataset: CIFAR-2 (Airplane vs. Frog) or CIFAR-10:
- Pre-trained Models: ResNet-50 or VGG-19
- Training Set Sizes: 500-7,000 (with an increment of 500)

With these combinations, we build $2 \times 2 \times 14 = 56$ victim models per repeated run and repeated it 30 times with different random seeds for a total of 1680 victim models. For each victim model, we record the standard test accuracy on clean test instances. Then, we also generate one data poisoning instance per model and evaluate the attack success rate afterward.

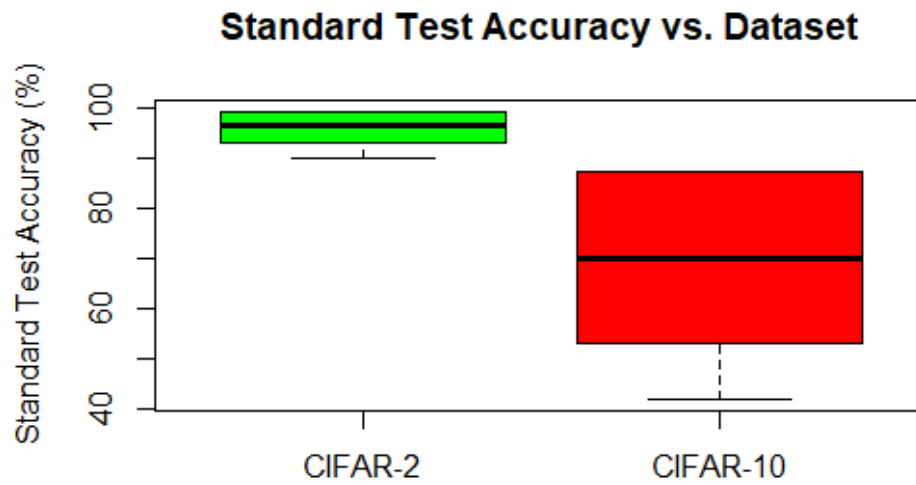


Figure 19. Distribution of Standard Test Accuracy for CIFAR-2 and CIFAR-10. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

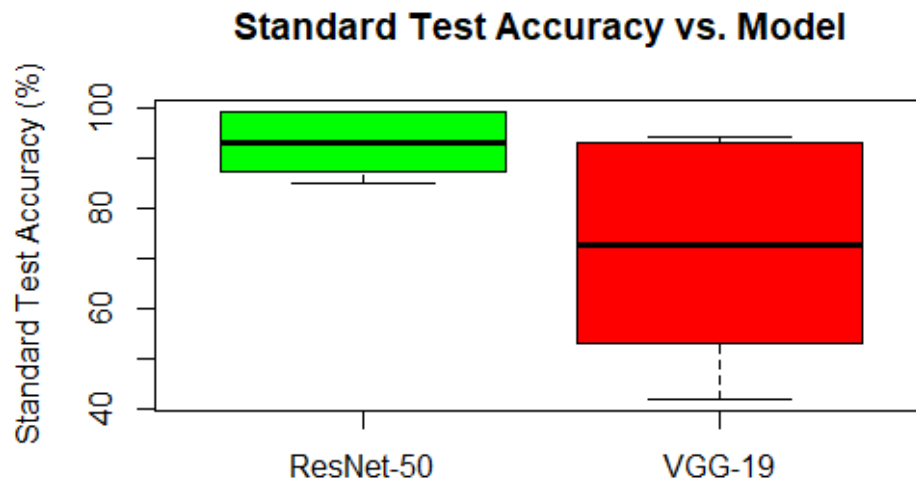


Figure 20. Distribution of Standard Test Accuracy for ResNet-50 and VGG-19

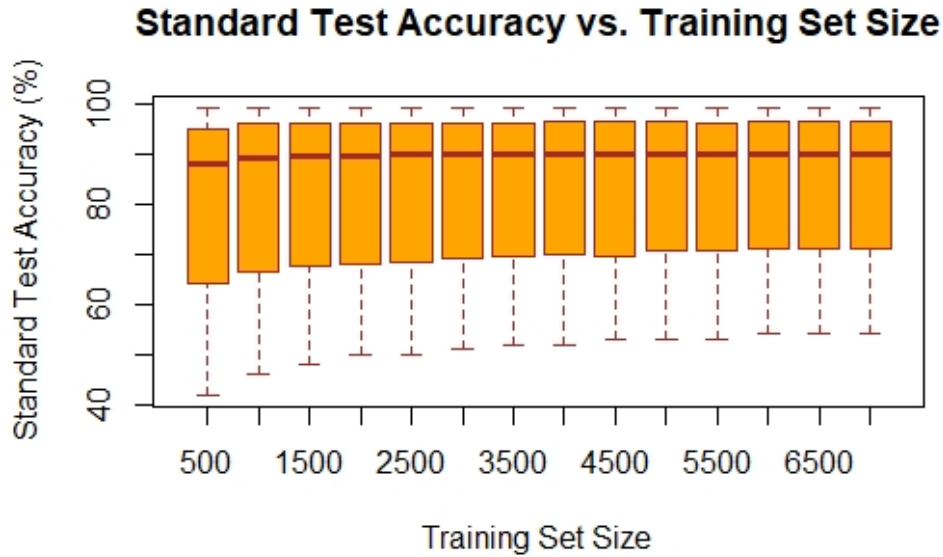


Figure 21. Distribution of Standard Test Accuracy with Different Training Set Size

Figure 19 shows the distribution of the standard test accuracy for two datasets, CIFAR-2 and CIFAR-10. As shown, it is easier to do binary classification since the training set range (500-7,000) is relatively small for the ten-class classification. Furthermore, the standard test accuracy for CIFAR-10 varies widely compared to that of CIFAR-2. As the number of instances per class ranges from 50 to 700 for CIFAR-10 and the performances of VGG-19 and ResNet-50 models are very different (as shown in Figure 20, ResNet-50 models usually have higher test accuracy than VGG-19 models do across the dataset and training set sizes), there is a wide variation in standard test accuracy. Similarly, Figure 21 shows a wide variation in the standard test accuracy for different training set sizes.

Besides whisker-and-box plots, we do a formal statistical analysis to check the relationship among the standard test accuracy, dataset, pre-trained model type, and training set sizes. Analysis of variance (ANOVA) has shown the statistically significant three-way interaction among the dataset types, dataset size, and model types on the standard test accuracy. However, while checking the assumptions of ANOVA, we notice that the homogeneity of variance is not satisfied. Particularly, Bartlett’s test, Levene’s test, and Fligner-Killeen’s test of homogeneity of variances all show a significant deviation from the constant variance assumption. Hence, we do a non-parametric version of ANOVA, Aligned Ranks Transformation ANOVA [147], which also shows the statistically significant three-way interaction between the dataset types,

Table 17 Average Reduction in Test Accuracy. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Model	Dataset	Average Reduction in Accuracy(%)
VGG-19	CIFAR-2	1.33
VGG-19	CIFAR-10	3.00
ResNet-50	CIFAR-2	0.06
ResNet-50	CIFAR-10	1.02

dataset size, and pretrained model types on the standard test accuracy (P-value $< 2.22 * 10^{-16}$ for three-way interaction term). Likewise, Aligned Ranks Transformation ANOVA shows the statistically significant three-way interaction between the dataset types, dataset size, and pretrained model types on the standard test accuracy after a data poison attack. However, there is not much difference in standard test accuracy between a poisoned model and an unpoisoned model except for the case of the VGG-19 models for classifying the CIFAR-10 dataset (see Table 17).

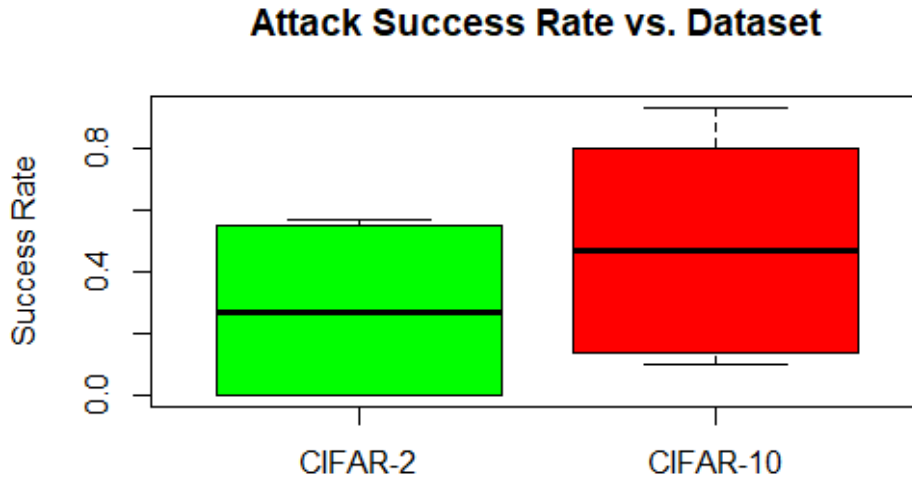


Figure 22. Distribution of Attack Success Rate for CIFAR-2 and CIFAR-10. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

Next, we consider the average attack success rate. The average attack success rate is calculated by averaging the 30 repeated runs for each combination of dataset types, dataset sizes, and pretrained models.

Target instances are the first 30 test instances that are correctly classified by the corresponding unpoisoned model. Since we take the average of 30 repeated runs, there are only four (two models and two datasets) data points per training set. Hence, we do not show the box plot for it. Figure 22 shows the average attack success rate on the CIFAR-2 dataset is lower than that of the CIFAR-10 dataset. Moreover, VGG-19 models are easier to attack than ResNet-50 models, as shown in Figure 23.

Besides whisker-and-box plots, we also want to do a formal statistical analysis. However, the data points do not follow the normal distribution as the QQ-plot of the residuals, as shown in Figure 24, indicates the normality assumption is violated. Hence, we utilized Aligned Ranks Transformation ANOVA instead of ANOVA. It reveals a statistically significant two-way interaction between the dataset and model types on the average attack success rate (P-value = $1.1045 * 10^{-12}$ for two-way interaction term). That is, the average attack success rate differs for different datasets and the pre-trained model selected for transfer learning. For instance, a victim model with the ResNet-50 pre-trained on ImageNet has a much lower attack success rate compared to a victim model trained with the VGG-19 pre-trained on ImageNet.

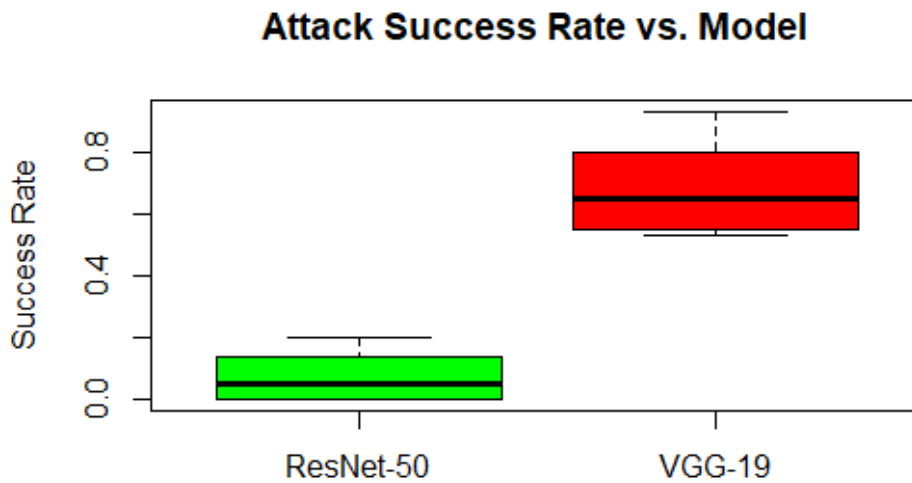


Figure 23. Distribution of the attack success rate for ResNet-50 and VGG-19. Reprinted from [86] © 2022 IEEE. Reused with permission as a senior author.

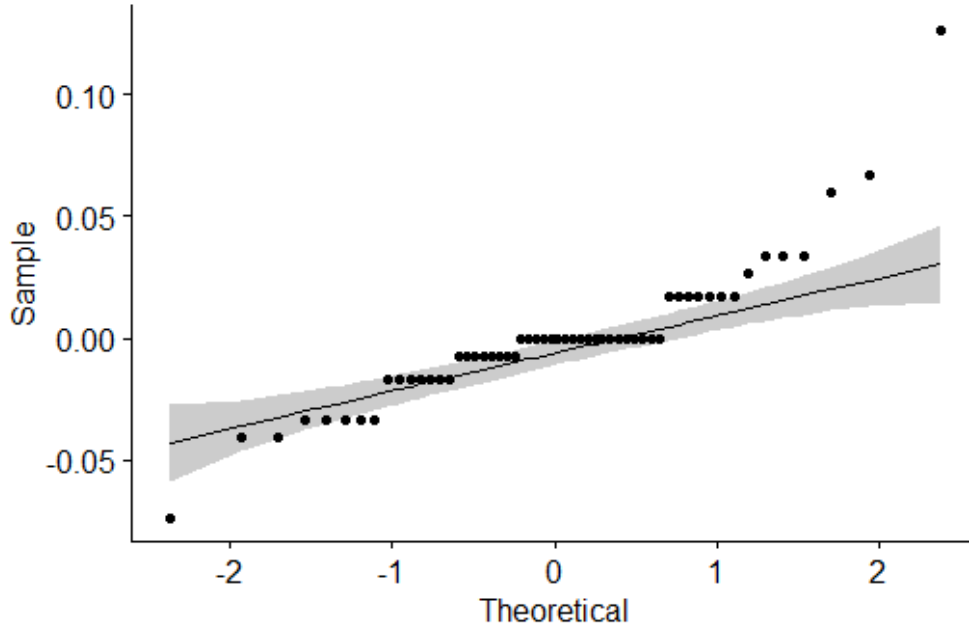


Figure 24. QQ Plot: Check Normality

5.6 Conclusions and Future Work

Although many existing studies focus on evasion attacks, a recent survey has found that data poisoning attack is the most concerning threat to machine learning practitioners [80]. Particularly, the data curation process is becoming more and more automated but without any mechanisms to filter malicious data scraped from insecure sources. In this paper, we have adapted FA attack, an adversarial attack method, for data poisoning attack against a transfer model to control its behavior and lead to targeted misclassification with just one injected poison instance. Different from the data poisoning attack on classical machine learning such as SVM [14], our studies have focused on the transfer learning models as these models are increasingly used to improve classification accuracy, reduce training time, and overcome data scarcity. We have compared the FA attack against the popular one-shot attack and showed its effectiveness. We have also studied the effects of these poisons in various scenarios and demonstrated their outstanding performance on the numerous combinations of training set sizes, datasets, and pre-trained models.

This work has the same disadvantage as [127] since both methods generate the data poisoning instance by solving equation (5.1), assuming that the model can learn to classify the poison instance correctly, which is not always the case. That is why the attack success rate is not always 100%. We will address this

issue in future work. Furthermore, this research has focused on the transfer models with freezing layers. Future studies will extend the FA attack to poison an end-to-end training process and improve the scalability of the proposed methodology. Moreover, this work focuses on adapting an adversarial attack method into a triggerless clean-label data poison attack method. In the future, we will evaluate the current defense strategies against such an attack.

Bibliography

- [1] Mahdieh Abbasi and Christian Gagné. “Robustness to adversarial examples through an ensemble of specialists”. In: arXiv preprint arXiv:1702.06856 (2017).
- [2] Hojjat Aghakhani et al. “Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability”. In: arXiv preprint arXiv:2005.00191 (2020).
- [3] Kashif Ahmad et al. “Developing Future Human-Centered Smart Cities: Critical Analysis of Smart City Security, Interpretability, and Ethical Challenges”. In: CoRR abs/2012.09110 (2020). <https://arxiv.org/abs/2012.09110>.
- [4] Naveed Akhtar and Ajmal Mian. “Threat of adversarial attacks on deep learning in computer vision: A survey”. In: IEEE Access 6 (2018), pp. 14410–14430.
- [5] Ethem Alpaydin. Introduction to machine learning. MIT press, 2020.
- [6] Maksym Andriushchenko et al. “Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search”. In: Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII. Ed. by Andrea Vedaldi et al. Vol. 12368. Lecture Notes in Computer Science. Springer, 2020, pp. 484–501.
- [7] Jordan T Ash et al. “Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds.” In: ICLR. 2020.
- [8] Eugene Bagdasaryan et al. “How To Backdoor Federated Learning”. In: The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2938–2948.
- [9] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. “Margin based active learning”. In: International Conference on Computational Learning Theory. Springer. 2007, pp. 35–50.

- [10] Andrew R Barron. “Approximation and estimation bounds for artificial neural networks”. In: *Machine learning* 14.1 (1994), pp. 115–133.
- [11] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.
- [12] James Bergstra et al. “Theano: Deep learning on gpus with python”. In: *NIPS 2011, BigLearning Workshop*, Granada, Spain. Vol. 3. Citeseer. 2011, pp. 1–48.
- [13] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. “Dimensionality reduction as a defense against evasion attacks on machine learning classifiers”. In: *arXiv preprint arXiv:1704.02654* (2017).
- [14] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning Attacks against Support Vector Machines”. In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, Edinburgh, Scotland, UK, June 26 - July 1, 2012. icml.cc / Omnipress, 2012.
- [15] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).
- [16] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. “Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks”. In: *arXiv preprint arXiv:1707.02476* (2017).
- [17] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models”. In: *6th International Conference on Learning Representations, ICLR 2018*, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. [OpenReview.net](https://openreview.net), 2018. <https://openreview.net/forum?id=SyZi0GWCZ>.
- [18] Heng Cao et al. “Adversarial DGA Domain Examples Generation and Detection”. In: *2020 International Conference on Control, Robotics and Intelligent System*. 2020, pp. 202–206.
- [19] Nicholas Carlini and David Wagner. “Adversarial examples are not easily detected: Bypassing ten detection methods”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM. 2017, pp. 3–14.
- [20] Nicholas Carlini and David Wagner. “Towards evaluating the robustness of neural networks”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 39–57.

- [21] Nicholas Carlini and David A. Wagner. “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. IEEE Computer Society, 2018, pp. 1–7. 10.1109/SPW.2018.00009. <https://doi.org/10.1109/SPW.2018.00009>.
- [22] Nicholas Carlini et al. “Provably minimally-distorted adversarial examples”. In: arXiv preprint arXiv:1709.10207 (2017).
- [23] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. “HopSkipJumpAttack: A Query-Efficient Decision-Based Attack”. In: 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020. IEEE, 2020, pp. 1277–1294. 10.1109/SP40000.2020.00045. <https://doi.org/10.1109/SP40000.2020.00045>.
- [24] Pin-Yu Chen et al. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models”. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. ACM, 2017, pp. 15–26.
- [25] Songkui Chen et al. “Image Denoising With Generative Adversarial Networks and its Application to Cell Image Enhancement”. In: IEEE Access 8 (2020), pp. 82819–82831. 10.1109/ACCESS.2020.2988284.
- [26] Chung-Cheng Chiu et al. “State-of-the-art speech recognition with sequence-to-sequence models”. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 4774–4778.
- [27] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, 2014, pp. 1724–1734. 10.3115/v1/d14-1179. <https://doi.org/10.3115/v1/d14-1179>.
- [28] François Chollet et al. Keras. 2015.
- [29] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: arXiv preprint arXiv:1511.07289 (2015).

- [30] Adam Coates, Andrew Y. Ng, and Honglak Lee. “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. Ed. by Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 215–223. <http://proceedings.mlr.press/v15/coates11a/coates11a.pdf>.
- [31] Iginio Corona, Giorgio Giacinto, and Fabio Roli. “Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues”. In: Inf. Sci. 239 (2013), pp. 201–225. 10.1016/j.ins.2013.03.022. <https://doi.org/10.1016/j.ins.2013.03.022>.
- [32] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: Machine learning 20.3 (1995), pp. 273–297.
- [33] Francesco Croce and Matthias Hein. “Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack”. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2196–2205. <http://proceedings.mlr.press/v119/croce20a.html>.
- [34] Francesco Croce and Matthias Hein. “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks”. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2206–2216. <http://proceedings.mlr.press/v119/croce20b.html>.
- [35] Francesco Croce et al. “Sparse-RS: a versatile framework for query-efficient sparse black-box adversarial attacks”. In: CoRR abs/2006.12834 (2020). arXiv: 2006.12834. <https://arxiv.org/abs/2006.12834>.
- [36] Kingma Da. “A method for stochastic optimization”. In: arXiv preprint arXiv:1412.6980 (2014).
- [37] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. IEEE Computer Society, 2009, pp. 248–255. 10.1109/CVPR.2009.5206848. <https://doi.org/10.1109/CVPR.2009.5206848>.
- [38] Li Deng. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]”. In: IEEE Signal Process. Mag. 29.6 (2012), pp. 141–142. 10.1109/MSP.2012.2211477. <https://doi.org/10.1109/MSP.2012.2211477>.

- [39] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: arXiv preprint arXiv:1810.04805 (2018).
- [40] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. “AdverTorch v0.1: An Adversarial Robustness Toolbox based on PyTorch”. In: arXiv preprint arXiv:1902.07623 (2019).
- [41] Melanie Ducoffe and Frederic Precioso. “Adversarial active learning for deep networks: a margin based approach”. In: arXiv preprint arXiv:1802.09841 (2018).
- [42] Logan Engstrom et al. “A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations”. In: CoRR abs/1712.02779 (2017). <http://arxiv.org/abs/1712.02779>.
- [43] Logan Engstrom et al. “Exploring the Landscape of Spatial Robustness”. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1802–1811. <http://proceedings.mlr.press/v97/engstrom19a.html>.
- [44] Farzaneh S Fard et al. “Impact of biased mislabeling on learning with deep networks”. In: 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017, pp. 2652–2657.
- [45] Reuben Feinman et al. “Detecting adversarial samples from artifacts”. In: arXiv preprint arXiv:1703.00410 (2017).
- [46] Liam Fowl et al. “Adversarial Examples Make Strong Poisons”. In: CoRR abs/2106.10807 (2021).
- [47] Paul L Garvin. On machine translation: selected papers. Vol. 128. Netherlands: Walter de Gruyter GmbH & Co KG, 2019.
- [48] Amin Ghiasi, Ali Shafahi, and Tom Goldstein. “Breaking Certified Defenses: Semantic Adversarial Examples with Spoofed robustness Certificates”. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020. <https://openreview.net/forum?id=HJxdTxHYvB>.
- [49] Tom Goldstein, Christoph Studer, and Richard Baraniuk. “A field guide to forward-backward splitting with a FASTA implementation”. In: arXiv preprint arXiv:1411.3406 (2014).
- [50] Ian Goodfellow. “NIPS 2016 tutorial: Generative adversarial networks”. In: arXiv preprint arXiv:1701.00160 (2016).

- [51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Cambridge, MA: MIT press, 2016.
- [52] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2015. <http://arxiv.org/abs/1412.6572>.
- [53] Divya Gopinath et al. “Deepsafe: A data-driven approach for checking adversarial robustness in neural networks”. In: arXiv preprint arXiv:1710.00486 (2017).
- [54] Sven Gowal et al. “On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models”. In: CoRR abs/1810.12715 (2018). <http://arxiv.org/abs/1810.12715>.
- [55] Kathrin Grosse et al. “On the (statistical) detection of adversarial examples”. In: arXiv preprint arXiv:1702.06280 (2017).
- [56] Shixiang Gu and Luca Rigazio. “Towards deep neural network architectures robust to adversarial examples”. In: arXiv preprint arXiv:1412.5068 (2014).
- [57] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. “Badnets: Identifying vulnerabilities in the machine learning model supply chain”. In: arXiv preprint arXiv:1708.06733 (2017).
- [58] Tianyu Gu et al. “BadNets: Evaluating Backdooring Attacks on Deep Neural Networks”. In: IEEE Access 7 (2019), pp. 47230–47244. 10.1109/ACCESS.2019.2909068.
- [59] Charles R. Harris et al. “Array programming with NumPy”. In: Nature 585.7825 (Sept. 2020), pp. 357–362. 10.1038/s41586-020-2649-2.
- [60] Kaiming He et al. “Deep residual learning for image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 770–778.
- [61] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: Proceedings of the IEEE international conference on computer vision. 2015, pp. 1026–1034.
- [62] Dan Hendrycks and Kevin Gimpel. “Early methods for detecting adversarial images”. In: arXiv preprint arXiv:1608.00530 (2016).

- [63] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: arXiv preprint arXiv:1606.08415 (2016).
- [64] Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition”. In: IEEE Signal processing magazine 29 (2012).
- [65] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: Neural computation 9.8 (1997), pp. 1735–1780.
- [66] Hossein Hosseini et al. “Blocking Transferability of Adversarial Examples in Black-Box Learning Systems”. In: CoRR abs/1703.04318 (2017). <http://arxiv.org/abs/1703.04318>.
- [67] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 7132–7141.
- [68] Ruitong Huang et al. “Learning with a Strong Adversary”. In: CoRR abs/1511.03034 (2015). <http://arxiv.org/abs/1511.03034>.
- [69] W. Ronny Huang et al. “MetaPoison: Practical General-purpose Clean-label Data Poisoning”. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual. Ed. by Hugo Larochelle et al. 2020.
- [70] Zonghao Huang, Miao Pan, and Yanmin Gong. “Robust Truth Discovery against Data Poisoning in Mobile Crowdsensing”. In: 2019 IEEE Global Communications Conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9-13, 2019. IEEE, 2019, pp. 1–6.
- [71] Miriam Huijser and Jan C van Gemert. “Active decision boundary annotation with deep generative models”. In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 5286–5295.
- [72] Jinyuan Jia et al. “Certified Robustness for Top-k Predictions against Adversarial Perturbations via Randomized Smoothing”. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.
- [73] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. “Multi-class active learning for image classification”. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 2009, pp. 2372–2379.

- [74] Guy Katz et al. “Reluplex: An efficient SMT solver for verifying deep neural networks”. In: International Conference on Computer Aided Verification. Springer. 2017, pp. 97–117.
- [75] Byeong Cheon Kim, Youngjoon Yu, and Yong Man Ro. “Robust Decision-Based Black-Box Adversarial Attack via Coarse-To-Fine Random Search”. In: 2021 IEEE International Conference on Image Processing (ICIP). IEEE. 2021, pp. 3048–3052.
- [76] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: arXiv preprint arXiv:1412.6980 (2014).
- [77] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: arXiv preprint arXiv:1312.6114 (2013).
- [78] Günter Klambauer et al. “Self-normalizing neural networks”. In: Advances in neural information processing systems 30 (2017), pp. 971–980.
- [79] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Tech. rep. Citeseer, 2009.
- [80] Ram Shankar Siva Kumar et al. “Adversarial Machine Learning-Industry Perspectives”. In: 2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, May 21, 2020. IEEE, 2020, pp. 69–75.
- [81] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: arXiv preprint arXiv:1607.02533 (2016).
- [82] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: Proceedings of the IEEE 86.11 (1998), pp. 2278–2324.
- [83] Mohan Li et al. “Deep Reinforcement Learning for Partially Observable Data Poisoning Attack in Crowdsensing Systems”. In: IEEE Internet Things J. 7.7 (2020), pp. 6266–6278.
- [84] Xin Li and Fuxin Li. “Adversarial examples detection in deep networks with convolutional filter statistics”. In: Proceedings of the IEEE International Conference on Computer Vision. 2017, pp. 5764–5772.
- [85] Jing Lin, Ryan Luley, and Kaiqi Xiong. “Active Learning Under Malicious Mislabeled and Poisoning Attacks”. In: 2021 IEEE Global Communications Conference (GLOBECOM). 2021, pp. 1–6. 10.1109/GLOBECOM46510.2021.9685101.

- [86] Jing Lin, Ryan Luley, and Kaiqi Xiong. “From Adversarial Examples to Data Poisoning Instances: Utilizing an Adversarial Attack Method to Poison a Transfer Learning Model”. In: 2022 IEEE International Conference on Communications © [2022] IEEE. Reprinted, with permission. IEEE, 2022.
- [87] Jing Lin, Laurent L. Njilla, and Kaiqi Xiong. “Robust Machine Learning against Adversarial Samples at Test Time”. In: IEEE International Conference on Communications. IEEE, 2020, 1–6. © [2020] IEEE. Reprinted, with permission. 10.1109/ICC40277.2020.9149002.
- [88] Jing Lin, Laurent L. Njilla, and Kaiqi Xiong. “Secure machine learning against adversarial samples at test time”. In: EURASIP J. Inf. Secur. 2022.1 (2022), pp. 1–15. 10.1186/s13635-021-00125-2. <https://doi.org/10.1186/s13635-021-00125-2>.
- [89] Jing Lin et al. “ML Attack Models: Adversarial Attacks and Data Poisoning Attacks”. In: arXiv (2021). 2112.02797.
- [90] Dong C. Liu and Jorge Nocedal. “On the limited memory BFGS method for large scale optimization”. In: Math. Program. 45.1-3 (1989), pp. 503–528. 10.1007/BF01589116. <https://doi.org/10.1007/BF01589116>.
- [91] Yuntao Liu, Yang Xie, and Ankur Srivastava. “Neural trojans”. In: 2017 IEEE International Conference on Computer Design (ICCD). IEEE. 2017, pp. 45–48.
- [92] Zhou Lu et al. “The expressive power of neural networks: A view from the width”. In: Advances in Neural Information Processing Systems. 2017, pp. 6231–6239.
- [93] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: Proc. icml. Vol. 30. 2013, p. 3.
- [94] Aleksander Madry et al. “Towards deep learning models resistant to adversarial attacks”. In: arXiv preprint arXiv:1706.06083 (2017).
- [95] Christoph Mayer and Radu Timofte. “Adversarial sampling for active learning”. In: The IEEE Winter Conference on Applications of Computer Vision. 2020, pp. 3071–3079.
- [96] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: The bulletin of mathematical biophysics 5.4 (1943), pp. 115–133.

- [97] J. H. Metzen et al. “Universal Adversarial Perturbations Against Semantic Image Segmentation”. In: 2017 IEEE International Conference on Computer Vision (ICCV). Oct. 2017, pp. 2774–2783. 10.1109/ICCV.2017.300.
- [98] Chenglin Miao et al. “Attack under Disguise: An Intelligent Data Poisoning Attack Mechanism in Crowdsourcing”. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018. Ed. by Pierre-Antoine Champin et al. ACM, 2018, pp. 13–22. 10.1145/3178876.3186032. <https://doi.org/10.1145/3178876.3186032>.
- [99] Chenglin Miao et al. “Towards Data Poisoning Attacks in Crowd Sensing Systems”. In: Proceedings of the Nineteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2018, Los Angeles, CA, USA, June 26-29, 2018. ACM, 2018, pp. 111–120. 10.1145/3209582.3209594. <https://doi.org/10.1145/3209582.3209594>.
- [100] Seungyong Moon, Gaon An, and Hyun Oh Song. “Parsimonious Black-Box Adversarial Attacks via Efficient Combinatorial Optimization”. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 4636–4645. <http://proceedings.mlr.press/v97/moon19a.html>.
- [101] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 2574–2582.
- [102] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal adversarial perturbations”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, pp. 1765–1773.
- [103] Nir Morgulis et al. “Fooling a real car with adversarial traffic signs”. In: arXiv preprint arXiv:1907.00374 (2019).
- [104] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. “When Does Label Smoothing Help?” In: CoRR abs/1906.02629 (2019).
- [105] Blaine Nelson et al. “Exploiting machine learning to subvert your spam filter”. In: Large-Scale Exploits and Emergent Threats 8 (2008), pp. 1–9.
- [106] Andrew Ng. “Nuts and bolts of building AI applications using Deep Learning”. In: Neural Information Processing Systems Keynote Talk (2016).

- [107] Hieu T Nguyen and Arnold Smeulders. “Active learning using pre-clustering”. In: Proceedings of the twenty-first international conference on Machine learning. 2004, p. 79.
- [108] Maria-Irina Nicolae et al. “Adversarial Robustness Toolbox v0.10.0”. In: CoRR 1807.01069 (2018).
- [109] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: IEEE Transactions on knowledge and data engineering 22.10 (2009), pp. 1345–1359.
- [110] Ren Pang et al. “A Tale of Evil Twins: Adversarial Inputs versus Poisoned Models”. In: Proceedings of the ACM Conference on Computer and Communications Security. 2020.
- [111] Nicolas Papernot et al. “Distillation as a defense to adversarial perturbations against deep neural networks”. In: 2016 IEEE Symposium on Security and Privacy (SP). IEEE. 2016, pp. 582–597.
- [112] Nicolas Papernot et al. “The Limitations of Deep Learning in Adversarial Settings”. In: IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016. IEEE, 2016, pp. 372–387. 10.1109/EuroSP.2016.36. <https://doi.org/10.1109/EuroSP.2016.36>.
- [113] Nicolas Papernot et al. “The limitations of deep learning in adversarial settings”. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE. 2016, pp. 372–387.
- [114] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu. “Label sanitization against label flipping poisoning attacks”. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer. 2018, pp. 5–15.
- [115] Gabriel Peyré, Marco Cuturi, et al. “Computational optimal transport: With applications to data science”. In: Foundations and Trends® in Machine Learning 11.5-6 (2019), pp. 355–607.
- [116] Tien Ho-Phuoc. “CIFAR10 to compare visual recognition performance between deep neural networks and humans”. In: arXiv preprint arXiv:1811.07270 (2018).
- [117] Junfei Qiu et al. “A survey of machine learning for big data processing”. In: EURASIP J. Adv. Signal Process. 2016 (2016), p. 67. 10.1186/s13634-016-0355-x. <https://doi.org/10.1186/s13634-016-0355-x>.
- [118] Moheeb Abu Rajab et al. “CAMP: Content-Agnostic Malware Protection”. In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013. The Internet Society, 2013.

- [119] Jonas Rauber, Wieland Brendel, and Matthias Bethge. “Foolbox: A python toolbox to benchmark the robustness of machine learning models”. In: arXiv preprint arXiv:1707.04131 (2017).
- [120] Esteban Real et al. “Regularized evolution for image classifier architecture search”. In: arXiv preprint arXiv:1802.01548 (2018).
- [121] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: CoRR abs/1609.04747 (2016).
- [122] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *Int. J. Comput. Vis.* 115.3 (2015), pp. 211–252. 10.1007/s11263-015-0816-y. <https://doi.org/10.1007/s11263-015-0816-y>.
- [123] Sara Sabour et al. “Adversarial Manipulation of Deep Representations”. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2016. <http://arxiv.org/abs/1511.05122>.
- [124] Motoki Sato et al. “Interpretable Adversarial Perturbation in Input Embedding Space for Text”. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. Ed. by Jérôme Lang. ijcai.org, 2018, pp. 4323–4330. 10.24963/ijcai.2018/601. <https://doi.org/10.24963/ijcai.2018/601>.
- [125] Ozan Sener and Silvio Savarese. “Active learning for convolutional neural networks: A core-set approach”. In: arXiv preprint arXiv:1708.00489 (2017).
- [126] Burr Settles. Active learning literature survey. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.
- [127] Ali Shafahi et al. “Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks”. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. Ed. by Samy Bengio et al. 2018, pp. 6106–6116.
- [128] Mahmood Sharif et al. “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition”. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM. 2016, pp. 1528–1540.

- [129] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: arXiv preprint arXiv:1409.1556 (2014).
- [130] Yang Song et al. “Pixeldefend: Leveraging generative models to understand and defend against adversarial examples”. In: arXiv preprint arXiv:1710.10766 (2017).
- [131] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. “One pixel attack for fooling deep neural networks”. In: IEEE Transactions on Evolutionary Computation (2019).
- [132] Vivienne Sze et al. “Efficient Processing of Deep Neural Networks: A Tutorial and Survey”. In: Proceedings of the IEEE 105 (Mar. 2017). 10.1109/JPROC.2017.2761740.
- [133] Christian Szegedy et al. “Intriguing properties of neural networks”. In: arXiv preprint arXiv:1312.6199 (2013).
- [134] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 2818–2826.
- [135] Rahim Taheri et al. “On defending against label flipping attacks on malware detection systems”. In: Neural Computing and Applications (2020), pp. 1–20.
- [136] Chuanqi Tan et al. “A survey on deep transfer learning”. In: International conference on artificial neural networks. Springer. 2018, pp. 270–279.
- [137] Te Juin Lester Tan and Reza Shokri. “Bypassing Backdoor Detection Algorithms in Deep Learning”. In: IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020. IEEE, 2020, pp. 175–183. 10.1109/EuroSP48549.2020.00019. <https://doi.org/10.1109/EuroSP48549.2020.00019>.
- [138] Sebastian Thrun and Lorien Pratt. Learning to learn. Springer Science & Business Media, 2012.
- [139] Florian Tramèr et al. “Ensemble adversarial training: Attacks and defenses”. In: arXiv preprint arXiv:1705.07204 (2017).
- [140] Florian Tramèr et al. “The Space of Transferable Adversarial Examples”. In: CoRR abs/1704.03453 (2017).

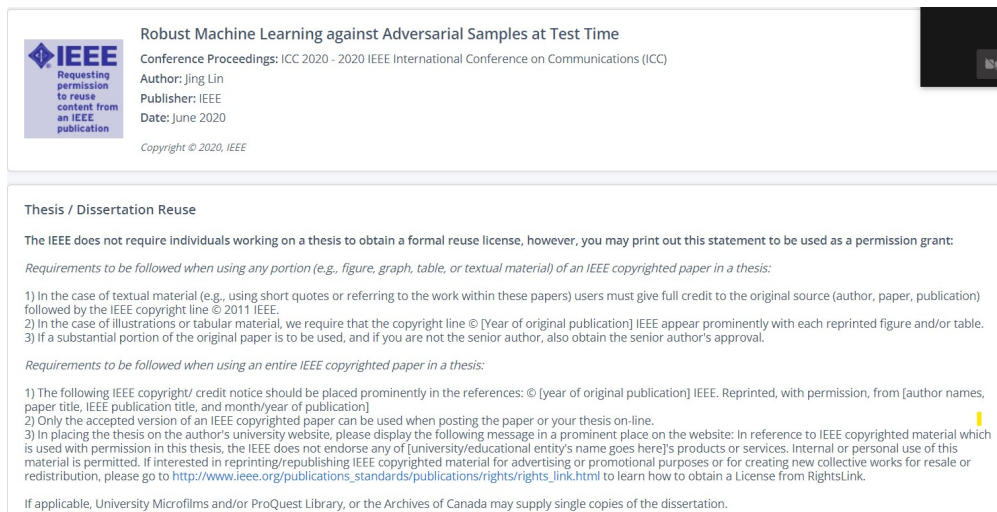
- [141] Brandon Tran, Jerry Li, and Aleksander Madry. “Spectral Signatures in Backdoor Attacks”. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 8011–8021.
- [142] Loc Truong et al. “Systematic Evaluation of Backdoor Data Poisoning Attacks on Image Classifiers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 788–789.
- [143] Chun-Chen Tu et al. “AutoZOOM: Autoencoder-Based Zeroth Order Optimization Method for Attacking Black-Box Neural Networks”. In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 742–749. 10.1609/aaai.v33i01.3301742. <https://doi.org/10.1609/aaai.v33i01.3301742>.
- [144] Bolun Wang et al. “Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 707–723. 10.1109/SP.2019.00031. <https://doi.org/10.1109/SP.2019.00031>.
- [145] Dan Wang and Yi Shang. “A new active labeling method for deep learning”. In: *2014 International joint conference on neural networks (IJCNN)*. IEEE. 2014, pp. 112–119.
- [146] Sandamal Weerasinghe et al. “Defending Support Vector Machines Against Data Poisoning Attacks”. In: *IEEE Trans. Inf. Forensics Secur.* 16 (2021), pp. 2566–2578.
- [147] Jacob O. Wobbrock et al. “The aligned rank transform for nonparametric factorial analyses using only anova procedures”. In: *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*. Ed. by Desney S. Tan et al. ACM, 2011, pp. 143–146.
- [148] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. “Wasserstein Adversarial Examples via Projected Sinkhorn Iterations”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. *Proceedings of Machine Learning Research*. PMLR, 2019, pp. 6808–6817.


- [149] Han Xu et al. “Adversarial Attacks and Defenses in Images, Graphs and Text: A Review”. In: *Int. J. Autom. Comput.* 17.2 (2020), pp. 151–178. 10.1007/s11633-019-1211-x. <https://doi.org/10.1007/s11633-019-1211-x>.
- [150] Weilin Xu, David Evans, and Yanjun Qi. “Feature squeezing: Detecting adversarial examples in deep neural networks”. In: *arXiv preprint arXiv:1704.01155* (2017).
- [151] Weilin Xu, Yanjun Qi, and David Evans. “Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers”. In: *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.
- [152] Yunzhe Xue, Meiyang Xie, and Usman Roshan. “Towards adversarial robustness with 01 loss neural networks”. In: *19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020, Miami, FL, USA, December 14-17, 2020*. Ed. by M. Arif Wani et al. IEEE, 2020, pp. 1304–1309. 10.1109/ICMLA51294.2020.00204. <https://doi.org/10.1109/ICMLA51294.2020.00204>.
- [153] Yao-Yuan Yang et al. “libact: Pool-based active learning in python”. In: *arXiv preprint arXiv:1710.00379* (2017).
- [154] X. Yuan et al. “Adversarial Examples: Attacks and Defenses for Deep Learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 1–20. ISSN: 2162-237X. 10.1109/TNNLS.2018.2886017.
- [155] Xiaoyong Yuan et al. “Adversarial examples: Attacks and defenses for deep learning”. In: *IEEE transactions on neural networks and learning systems* (2019).
- [156] Valentina Zantedeschi, Maria-Irina Nicolae, and Amrith Rawat. “Efficient defenses against adversarial attacks”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 39–49.
- [157] Valentina Zantedeschi, Maria-Irina Nicolae, and Amrith Rawat. “Efficient defenses against adversarial attacks”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 39–49.
- [158] Zhengli Zhao, Dheeru Dua, and Sameer Singh. “Generating natural adversarial examples”. In: *arXiv preprint arXiv:1710.11342* (2017).

- [159] Chen Zhu et al. “Transferable clean-label poisoning attacks on deep neural nets”. In: arXiv preprint arXiv:1905.05897 (2019).
- [160] Jia-Jie Zhu and José Bento. “Generative adversarial active learning”. In: arXiv preprint arXiv:1702.07956 (2017).
- [161] Barret Zoph et al. “Learning transferable architectures for scalable image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 8697–8710.

Appendix A

Copyright Permissions



 **Requesting permission to reuse content from an IEEE publication**

Robust Machine Learning against Adversarial Samples at Test Time
Conference Proceedings: ICC 2020 - 2020 IEEE International Conference on Communications (ICC)
Author: Jing Lin
Publisher: IEEE
Date: June 2020

Copyright © 2020, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

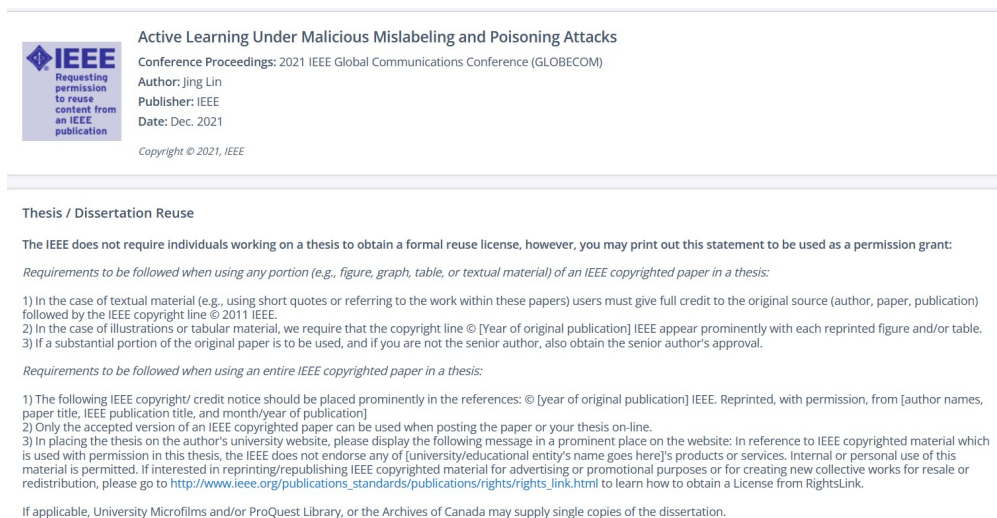
- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.


Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure 25. 2020 IEEE ICC Copyright Permission



 **Requesting permission to reuse content from an IEEE publication**

Active Learning Under Malicious Mislabeling and Poisoning Attacks
Conference Proceedings: 2021 IEEE Global Communications Conference (GLOBECOM)
Author: Jing Lin
Publisher: IEEE
Date: Dec. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Figure 26. 2021 IEEE Global Communications Conference Copyright Permission

Copyright

- Copyright on any open access article in a journal published by SpringerOpen is retained by the author(s).
- Authors grant SpringerOpen a [license](#) to publish the article and identify itself as the original publisher.
- Authors also grant any third party the right to use the article freely as long as its integrity is maintained and its original authors, citation details and publisher are identified.
- The [Creative Commons Attribution License 4.0](#) formalizes these and other terms and conditions of publishing articles.

In addition to SpringerOpen's copyright policy, some journals also follow an Open Data policy and the [Creative Commons CC0 1.0 Public Domain Dedication waiver](#) applies to all published data in these journals. Further information can be found on the individual journals pages and in the [license agreement](#).

Where an author is prevented from being the copyright holder (for instance in the case of US government employees or those of Commonwealth governments), minor variations may be required. In such cases the copyright line and license statement in individual articles will be adjusted, for example to state '© 2016 Crown copyright'. Authors requiring a variation of this type should [inform SpringerOpen](#) during or immediately after submission of their article. Changes to the copyright line cannot be made after publication of an article.

Figure 27. SpringerOpen Copyright Permission from <https://jis-eurasipjournals.springeropen.com/submission-guidelines/copyright>

*Correspondence: xiongk@usf.edu

The paper is an extension of our original work presented in J. Lin, L. L. Njilla and K. Xiong, "Robust Machine Learning against Adversarial Samples at Test Time," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1–6, doi: 10.1109/ICC40277.2020.9149002.

¹ICNS Lab and Cyber Florida, University of South Florida, Tampa, FL, USA


Full list of author information is available at the end of the article



© The Author(s). 2022 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

mislead a DNN model to make a wrong classification. As shown in Fig. 1, Carlini and Wagner (C&W) attack [17] is applied to an image of number "5" from the MNIST dataset [18] to generate an image shown on the right in this figure. This generated image appears the same as the

Figure 28. Open Access Statement



Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)


This is a human-readable summary of (and not a substitute for) the [license](#). [Disclaimer](#).

You are free to:


- Share** — copy and redistribute the material in any medium or format
- Adapt** — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.


Figure 29. Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) from <https://creativecommons.org/licenses/by-nc-sa/4.0/>



CC BY-SA: Creative Commons Attribution-ShareAlike
This license allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, so long as attribution is given to the creator. The license allows for commercial use. If you remix, adapt, or build upon the material, you must license the modified material under identical terms.



CC BY-NC-SA: Creative Commons Attribution-Noncommercial-ShareAlike
This license allows reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator. If you remix, adapt, or build upon the material, you must license the modified material under identical terms.



CC BY-NC-ND: Creative Commons Attribution-Noncommercial-NoDerivatives
This license allows reusers to copy and distribute the material in any medium or format in unadapted form only, for noncommercial purposes only, and only so long as attribution is given to the creator.

Many publishers allow "accepted manuscripts" to be deposited with a CC BY-NC-ND license, but there may be an embargo period. Check the journal's policies to be certain.

Figure 30. ArXiv License Information from <https://arxiv.org/help/license>