Adversarial Learning and Secure AI

Providing a logical framework for student learning, this is the first textbook on adversarial learning. It introduces students to attacks and vulnerabilities of deep learning, and to methods for defending against attacks and making AI generally more robust.

It is the ideal resource for upper undergraduate and first-year graduate courses on AI security and adversarial learning. Students and instructors will benefit from these features

- application examples, case studies, and real-world student projects in each chapter, connecting theory with practice
- a project-driven approach that strengthens critical thinking when evaluating attacks and defenses
- a variety of application areas covered by the examples and projects, for example, image classification, text classification, point cloud classification, and a regression example from finance.

**David J. Miller** is professor of Electrical Engineering at the Pennsylvania State University.

**Zhen Xiang** is a post-doctoral research associate of Computer Science at the University of Illinois, Urbana-Champaign.

**George Kesidis** is a professor of Computer Science and Engineering, and of Electrical Engineering, at the Pennsylvania State University.
“This textbook is one of the first major efforts to systematically examine adversarial machine learning. It clearly outlines the most common types of attacks on machine learning/AI, and defenses, with rigorous yet practical discussions. I would highly recommend it to any instructor or machine learning student who seeks to understand how to make machine learning more robust and secure.”

Carlee Joe-Wong, Carnegie Mellon University

“This is a clear and timely introduction to the vital topic of adversarial learning. As leading international experts, the authors provide an accessible explanation of the foundational principles and then deliver a nuanced and extensive survey of recent attack and defense strategies. Multiple suggested projects allow the book to serve as the core of a graduate course.”

Mark Coates, McGill University

“Remarkably comprehensive, this book explores the realm of adversarial learning, revealing the vulnerabilities and defenses associated with deep learning. With a mix of theoretical insights and practical projects, the book challenges the misconceptions about the robustness of Deep Neural Networks, offering strategies to fortify them. It is well suited for students and professionals with basic calculus, linear algebra, and probability knowledge, and provides foundational background on deep learning and statistical modeling. A must-read for practitioners in the machine learning field, this book is a good guide to understanding adversarial learning, the evolving landscape of defenses, and attacks.”

Ferdinando Fioretto, Syracuse University

“In a field that is moving at break-neck speed, this book provides a strong foundation for anyone interested in joining the fray.”

Amir Rahmati, Stony Brook
Adversarial Learning and Secure AI

DAVID J. MILLER
Pennsylvania State University

ZHEN XIANG
University of Illinois, Urbana-Champaign

GEORGE KESIDIS
Pennsylvania State University
DJM dedicates this book to his children Joshua and Madeline

ZX dedicates this book to his son Ian

GK dedicates this book to Fozzie, Gonzo and Therese

The authors also share a dedication of this book to their collaborators Xi Li and Hang Wang
## Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>page xiii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notation</td>
<td>xvii</td>
</tr>
</tbody>
</table>

### 1 Overview of Adversarial Learning

- 1.1 Machine Learning and Its Attack Vectors 2
- 1.2 Attacker/Defender Goals and Assumptions 2
- 1.3 Test-Time Evasion Attacks (TTEs) or Adversarial Inputs 4
- 1.4 Data Poisoning (DP) Attacks 8
- 1.5 Reverse-Engineering Attacks (REAs) Targeting the Deep Neural Network (DNN) 13
- 1.6 Attacks on Privacy of Training Data 14
- 1.7 Chapter Summary 17
- 1.8 References for Further Reading 18

### 2 Deep Learning Background

- 2.1 Deep Learning for Classification, Regression, or Prediction 20
- 2.2 Motivating Deep Neural Network (DNN) Classifiers 23
- 2.3 Linearly Separable Data 23
- 2.4 From Binary to $K$-ary Classification 24
- 2.5 Deep Neural Network (DNN) Architectures 24
- 2.6 Background on Gradient-Based Optimization 31
- 2.7 Heuristic Optimization Methods for Deep Learning 36
- 2.8 Overfitting and DNN Regularization 41
- 2.9 “Certified” Training and Classification Confidence 44
- 2.10 Neural Network Inversion 46
- 2.11 Identification and Visualization of Salient Features 47
- 2.12 Handling Label-Deficient Data: Transfer and Contrastive Learning 47
- 2.13 Other Methods of Extracting Salient Features 50
- 2.14 Other Types of Classifiers: Naive Bayes (NB) and Logistic Regression (LR) 50
- 2.15 Discussion: Statistical Confidence 52
- 2.16 Chapter Summary 53
- 2.17 References for Further Reading 54
- 2.18 Project: Classification of UC Irvine Datasets 54
2.19 Project: Membership-Inference Attack 55
2.20 Project: Classification for the CIFAR-10 Image Domain 55

3 Basics of Detection and Mixture Models 56
3.1 Mixture Densities 58
3.2 Estimating the Parameters: Maximum Likelihood Estimation (MLE) and Expectation-Maximization (EM) 59
3.3 K-Means Clustering as a Special Case 63
3.4 Model Order Selection 63
3.5 Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) 66
3.6 Some Detection Basics 70
3.7 Performance Measures for Detection 72
3.8 Chapter Summary 72
3.9 References for Further Reading 73
3.10 Projects: Receiver Operating Characteristic (ROC), Principal Component Analysis (PCA), and Gaussian Mixture Model (GMM) 73

4 Test-Time Evasion Attacks (Adversarial Inputs) 76
4.1 Previously Proposed Test-Time Evasion (TTE) Attacks 76
4.2 “Robust” and “Certified” Defenses for Test-Time Evasion (TTE) Attacks 80
4.3 Anomaly Detection (AD) of Test-Time Evasion (TTE) Attacks 85
4.4 Background on Generative Modeling and Generative Adversarial Networks (GANs) 90
4.5 Generative Adversarial Network (GAN) Based Test-Time Evasion (TTE) Attack Detection Methodology 93
4.6 Experiments 96
4.7 Deeper Consideration of Test-Time Evasion (TTE) Attack Scenarios 105
4.8 Discussion: Out-of-Distribution Detection (OODD) 110
4.9 Chapter Summary 111
4.10 Project: White Region Counting Defense 112
4.11 Project: Nearest Neighbor (NN) Classification Defense 114
4.12 Project: Test-Time Evasion (TTE) Attacks and Dropout 115

5 Backdoors and Before/During Training Defenses 116
5.1 Backdoor Attacks 117
5.2 Before/During Training Defender’s Goals 120
5.3 Before/During Training Defenses 121
5.4 Training Set Cleansing Reverse-Engineering Defense (TSC-RED) 122
5.5 Experiments 128
5.6 Defense Variations and Additional Experiments 138
5.7 Chapter Summary 139
5.8 Project: Principal Component Analysis (PCA) Based Cluster Impurity (CI) Defense 140
## Contents

6 Post-Training Reverse-Engineering Defense (PT-RED) Against Imperceptible Backdoors 141
   6.1 The Post-Training (PT) Scenario 141
   6.2 Some Post-Training (PT) Defenses 143
   6.3 Imperceptible-Backdoor Post-Training Reverse-Engineering Defense (I-PT-RED) 146
   6.4 Experiments 154
   6.5 Lagrangian Post-Training Reverse-Engineering Defense (L-PT-RED) and Experiments 173
   6.6 Discussion: Robust and Explainable AI 178
   6.7 Chapter Summary 179
   6.8 Project: Imperceptible Backdoor Post-Training Reverse-Engineering Defense (I-PT-RED) on Images 180
   6.9 Project: Consensus Post-Training Reverse-Engineering Defense (C-PT-RED) 182
   6.10 Project: Noisy Backdoor Incorporation 184

7 Post-Training Reverse-Engineering Defense (PT-RED) Against Patch-Incorporated Backdoors 185
   7.1 Perceptible Backdoor Patterns 186
   7.2 Choice of Source Class(es) and Target Class 188
   7.3 Perceptible Backdoor Post-Training Reverse-Engineering Defense (P-PT-RED) 188
   7.4 Experiments 195
   7.5 Chapter Summary 207
   7.6 Project: Exploring What the Deep Neural Network (DNN) is Learning 209
   7.7 Project: Perceptible Backdoor Post-Training Reverse-Engineering Defense (P-PT-RED) and Variations on Images 209

8 Transfer Post-Training Reverse-Engineering Defense (T-PT-RED) Against Backdoors 211
   8.1 Transferability of Sample-wise Minimal Perturbations 211
   8.2 Transfer Post-Training Reverse-Engineering Defense (T-PT-RED) Detection Procedure 213
   8.3 Experiments 216
   8.4 Chapter Summary 222
   8.5 Project: Targeted Transfer Post-Training Reverse-Engineering Defense (T-PT-RED) for Multiple Classes 223
   8.6 Project: Transfer Post-Training Reverse-Engineering Defense (T-PT-RED) for Backdoor Patches 224

9 Universal Post-Training (PT) Backdoor Defenses 226
   9.1 Universal Backdoor Detection (UnivBD) Without Clean Labeled Data 227
   9.2 Universal Mitigation of Backdoor Attack (UnivBM) 234
   9.3 Some Additional Experiments 238
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>9.5</td>
<td>Project: Universal Backdoor Detection (UnivBD) Versus Lagrangian Post-Training Reverse-Engineering Defense (L-PT-RED)</td>
</tr>
<tr>
<td>9.6</td>
<td>Project: Reverse-Engineering Backdoor Patterns (BPs)</td>
</tr>
<tr>
<td>9.7</td>
<td>Project: Universal Backdoor Detection (UnivBD) Leveraging Clean Labeled Data</td>
</tr>
<tr>
<td>9.8</td>
<td>Project: Universal Detector Based on Deep Neural Network (DNN) Weight Outliers</td>
</tr>
<tr>
<td>9.9</td>
<td>Project: Mitigation Using Surrogates for Correct Decision Rate</td>
</tr>
<tr>
<td>9.10</td>
<td>Project: Testing the Hypothesis that a Backdoor Preserves Clean Logits</td>
</tr>
<tr>
<td>9.11</td>
<td>Project: Modified UnivBM Objective to Reduce Margin</td>
</tr>
<tr>
<td>9.12</td>
<td>Project: Defense Against Error-Generic Data Poisoning</td>
</tr>
<tr>
<td>10</td>
<td>Test-Time Detection of Backdoor Triggers</td>
</tr>
<tr>
<td>10.1</td>
<td>Some Test-Time Backdoor Detection Methods</td>
</tr>
<tr>
<td>10.2</td>
<td>In-Flight Reverse-Engineering Defense (IF-RED)</td>
</tr>
<tr>
<td>10.3</td>
<td>Experiments</td>
</tr>
<tr>
<td>10.4</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>10.5</td>
<td>Project: Trigger Detection via Test-Time Evasion (TTE) Attack Detection Strategy</td>
</tr>
<tr>
<td>10.7</td>
<td>Project: Margin as an In-Flight Detection Statistic</td>
</tr>
<tr>
<td>11</td>
<td>Backdoors for 3D Point Cloud (PC) Classifiers</td>
</tr>
<tr>
<td>11.1</td>
<td>3D Point Cloud (PC) Classification</td>
</tr>
<tr>
<td>11.2</td>
<td>Backdoor Attacks against 3D Point Cloud (PC) Classifiers</td>
</tr>
<tr>
<td>11.3</td>
<td>A Small Cluster of Backdoor Points</td>
</tr>
<tr>
<td>11.4</td>
<td>Attack Experiments</td>
</tr>
<tr>
<td>11.5</td>
<td>Point Cloud (PC) Anomaly Detectors (ADs) against Backdoor Attacks (BAs)</td>
</tr>
<tr>
<td>11.6</td>
<td>Point Cloud Post-Training Reverse-Engineering Defense (PC-PT-RED)</td>
</tr>
<tr>
<td>11.7</td>
<td>Attack/Defense Experiments</td>
</tr>
<tr>
<td>11.8</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>11.9</td>
<td>Project: During-Training Defense or Robustification of Point Cloud (PC) Classifiers</td>
</tr>
<tr>
<td>12</td>
<td>Robust Deep Regression and Active Learning</td>
</tr>
<tr>
<td>12.1</td>
<td>Background on Active Learning</td>
</tr>
<tr>
<td>12.2</td>
<td>Robust Deep Regression by Active Learning (RDR-AL)</td>
</tr>
<tr>
<td>12.3</td>
<td>A Localized Region of Regression Error</td>
</tr>
<tr>
<td>12.4</td>
<td>Experimental Results for Valuation of a Financial Option</td>
</tr>
</tbody>
</table>
12.5 Discussion Topics: Query by Committee, Reinforcement Learning (RL), Test-Time Evasion (TTE), Classification 290
12.6 Chapter Summary 292
12.7 Project: Clean Label Backdoor Attack 292

13 Error Generic Data Poisoning Defense 294
13.1 Threat Model 295
13.2 Data Poisoning Defenses 296
13.3 Bayesian Information Criterion Based Mixture Model Training Set Cleansing (BIC-MM-TSC) 298
13.4 Experiments on Binary, Discrete Feature Classification Tasks 304
13.5 Discussion: Experiments with $K > 2$ Classes 312
13.6 Chapter Summary 312
13.7 Project: $K$-Nearest Neighbor ($K$NN) Defense 313
13.8 Project: White Box Data Poisoning Attack 314

14 Reverse-Engineering Attacks (REAs) on Classifiers 315
14.1 Reverse-Engineering Attacks (REAs) Given Domain Samples 316
14.2 Overview of Defense Against Reverse-Engineering Attacks (REAs) 316
14.3 Anomaly Detection of Attacks (ADA) Based Defense Against Reverse-Engineering Attacks (REA-ADA) 317
14.4 Experiments 317
14.5 Chapter Summary 319
14.6 Project: Defense Against Random Querying 321

Appendix Support Vector Machines (SVMs) 322

References 333
Index 351
Preface

Why We Wrote This Book

In the past ten years, deep learning has been applied to many market and government sectors (e.g., health, finance, military, intelligence, manufacturing, sales), including in their critical infrastructure and supply chains (MLOps/AIOps). Application domains include those where operational conditions may change over time (model drift), where safety and security are of great concern, and where significant financial stakes are involved. As such, the deep learning process and the trained Deep Neural Networks (DNNs or “AIs”) themselves have become targets of attack. More generally, basic questions about the robustness and explainability of DNN solutions have also been raised even in the absence of attacks, for example, [132]. A research sub-field assessing and addressing the risks associated with using AIs (and other machine learning models) is known as adversarial learning. This area essentially represents a merger between the fields of computer security and machine learning.

An important aspect of software security is to consider how the software will behave for all possible valid inputs. The reason for this is that an adversary may exploit a vulnerability that pertains to a range of inputs for which the software’s behavior was not carefully considered by its developers. This is a daunting security task for a DNN, whose behavior depends on an enormous set of parameters (even billions) which are heuristically learned, and whose input space may be very high-dimensional. What this means is that DNNs have a substantial attack “surface,” which makes them vulnerable to a variety of attacks/exploits. While some basic adversarial learning research dates back more than 20 years, this field really took off with the observation in 2014 that adversarial inputs may be easily constructed for DNNs – these are small changes to an input pattern, imperceptible to a human being, and yet which greatly alter the DNN’s output (e.g., changing its class decision). Aside from being a security threat, adversarial inputs demonstrate that it is a fallacy – held by many researchers, educators, industrialists, and journalists – that DNNs are generally robust, reliable decision-makers, and are close to fulfilling the promise of artificial intelligence. In the year 2020 alone, more than 1100 papers on adversarial learning were submitted to arXiv.org. While there are a number of review papers, to date there are no books on this subject which are suitable for a course offering.
The Emphasis of the Book is Unsupervised Defenses

Generally, defenses and attacks continuously evolve. New vulnerabilities are discovered by attackers (either in the system being protected or in its defenses) and exploited. Defenses may evolve to address newly identified vulnerabilities (including those recently revealed by new exploits). It may be unrealistic to suppose that the defender has detailed knowledge of an attack that may be mounted. This is why we focus on unsupervised defenses that aim to protect against a whole family of attacks (rather than relying on somehow obtained knowledge of a specific known attack [269, 270])\(^1\). On the other hand, in the quest for the glory (and concomitant research funding) associated with finding a new vulnerability and devising an exploit for it, some researchers ignore existing or obvious defenses which would be effective against their attacks, or get carried away and unrealistically assume an omniscient or omnipotent attacker (e.g., one who completely controls the training dataset and training process, or controls how new samples are labeled in an active learning context). Given an omnipotent adversary, a defense may be able to do little more than increase the attacker’s work factor. That said, though “security through obscurity” is commonly practiced and may be effective in some cases, assuming some attacker knowledge of a defense is not unreasonable. This is especially true considering spectacular leaks by insiders and breaches in privacy protections in the recent past.\(^2\)

Purpose, Target Audience and Prerequisites

The targeted audience for this book is senior undergraduates and graduate students in all branches of science and engineering. The purpose of this book is to introduce students to existing attacks and vulnerabilities of deep learning (and machine learning in general) and to methods for defending against these attacks, as well as for making AI generally more robust (even in the absence of attack). Along the way, students will also enhance their appreciation for what deep neural networks are in fact learning (and what they are not learning). For example, students will learn that training dataset augmentation (i) may improve generalization performance, (ii) instead, may cause degradation in DNN accuracy (e.g., by overfitting through adversarially robust learning), (iii) or may result in the planting of a backdoor in the DNN. As another example, students will better understand the circumstances in which DNNs learn patterns that are spatially invariant (occurring anywhere in an image), or only patterns that are spatially fixed. The book covers many attack-defense scenarios and involves many case studies and real-world problems addressed by the state-of-the-art in recent research publications.

\(^1\) Note that an antivirus system or firewall typically functions in response to known attacks, that is, they are supervised defenses. Hence periodic updates (with patching of exposed vulnerabilities) are needed, typically after a new exploit has been detected and carefully studied to identify its signature.

\(^2\) Which can go both ways, that is, new attacks can also be leaked before they are launched; but, to reiterate, we focus on unsupervised defenses herein.
Prerequisites for this book include a basic introduction to calculus, linear algebra, and probability. Though the second and third chapters provide the necessary background material on deep learning, detection, and statistical modeling, a student would benefit from a more broadly scoped course on pattern recognition and machine learning based on, for example, [63, 190], and from an introduction to numerical analysis, for example, [8].

Projects

There are course projects at the ends of the chapters that give hands-on experience to students in devising and evaluating both attacks and defenses against machine learning systems. These projects are intended as the primary homework exercises for a course on robust and adversarial learning. They also serve the dual purpose of helping students to obtain familiarity and facility in machine learning design within the Python programming environment (in particular, the use of PyTorch for deep learning). Moreover, these projects provide a window for students into how much research work is being conducted in AI/machine learning – with promising new ideas postulated and then experimentally assessed, both to validate (or reject) them and to obtain greater insight into the problem at hand. Given some Python experience, students can learn PyTorch [209] while studying the first few chapters of this book. Also, a tutorial on the Pillow fork of the Python Image Processing Library (PIL) will be useful, for example, [88]. PyTorch code for projects given at the end of Chapters 4, 5, 6 and 13 is available at: www.cambridge.org/millersecureAI.

Quite a bit of code is provided for the first few preliminary PyTorch projects (the provided code should be carefully studied by the student), while little to no code is provided for subsequent projects. The idea is that the students can “fill in the blanks” for the first PyTorch projects that are assigned but have to produce all of the code for subsequent ones.

Chapter Roadmap

The first three chapters respectively provide background on attack types and attack nomenclature, on deep learning, and on detection and estimation. If students have taken a prior course on pattern recognition or machine learning, they may be able to skip Chapters 2 and 3. Note that subsequent chapters frequently refer back to material in Chapters 2 and 3.

Chapter 4 addresses defenses against adversarial inputs at test-time, also known as test-time evasion (TTE) attacks.

Chapter 13 addresses defense against general data poisoning attacks against classifiers.

Chapter 14 addresses defense against reverse-engineering (probing) attacks.
A road map for the remaining chapters on backdoor defense is as follows.

- Chapter 5 addresses backdoor defense implemented by the training authority, who has access to the (possibly poisoned) training set and who controls the training process.

- The next four chapters address post-training backdoor defense, where the defender does not in fact have access to the training set, but only to the trained classifier and to a (relatively) very small set of clean (unpoisoned) labeled samples.

- Chapter 6 addresses defense against imperceptible backdoor attacks. One approach reverse-engineers putative backdoor patterns that are additively incorporated either to the input (raw features) or to an internal layer of the neural network (embedded features). The reverse-engineered backdoor pattern has utility beyond post-training detection (it can also be used for test-time detection of backdoor triggers and for mitigating the effect of backdoors). Moreover, it is an important element of explainable AI (XAI), indicating patterns in the presence of which a DNN’s decision-making is fragile.

- Chapter 7 addresses post-training defense against backdoors that are embedded by replacing a “patch” of input features by the backdoor pattern. These backdoor attacks can be implemented either digitally or physically (e.g., by placing an object – the backdoor pattern – in a given scene). One reverse-engineering defense exploits the fact that the attack should be “scene-plausible” in order to be evasive.

- The defenses in Chapters 6 and 7 are not very suitable when the number of classes in the problem is small (e.g., the two-class case), since in this case there are insufficient detection statistics available for estimating the parameters that specify a detection rule. The post-training defenses in Chapter 8 address this problem. The main defense developed there was found to be effective with a constant detection threshold (1/2), irrespective of the DNN architecture and classification domain.

- Chapter 9 considers defenses that aim to be universal, that is, without any explicit or implicit assumptions about the backdoor pattern or how it was embedded.

- Chapter 10 considers “in-flight” detection, that is, detection of backdoor triggers in input patterns at test time. Such detection may give the potential to catch culprits in the act of exploiting the backdoor mapping. One such described defense leverages a reverse-engineered backdoor pattern.

- Chapter 11 considers backdoor detection for non-image point cloud data classifiers.

- Chapter 12 considers backdoors for regression rather than classification and also discusses active learning.

The authors acknowledge the support of students and colleagues. In particular, we thank Yujia Wang (Chapters 4 and 14), Xi Li (Chapters 10, 12 and 13), and Hang Wang (Chapters 4 and 9), as well as Zhicong Qiu and Xinyi Hu. We also thank Vladimir Lucic for consultations regarding Chapter 12.

The authors acknowledge the sources of our research support through the Pennsylvania State University: an AFOSR DDDAS grant (2017–2021), an ONR NROTC education grant (2021–2022), an NRC Research Associate Fellowship with AFRL (2021–2022), and two Cisco Systems gifts (2019, 2022). Through Anomalee Inc., we also acknowledge the research support of an NSF SBIR Phase-One grant (2022–2023).
Notation

Typically,

- random objects are denoted by capital (upper-case) letters
- non-vector matrices are denoted by bold capital letters, for example,
  \[ V = [v_{i,j}]_{i=1,...,n, j=1,...,m} \]
  denotes an \( n \times m \) matrix with entry \( v_{i,j} \) in the \( i \)th row and \( j \)th column, and both \( m > 1 \) and \( n > 1 \)
- column vectors are denoted by underlined lower-case letters
- datasets are typically denoted by calligraphic capital letters
- some variables not defined below, such as \( x, y, z, n, m, i, j, k, \alpha, \beta, \kappa, \theta \), are often repurposed in various chapters
- some symbols, such as \( f, g \), are typically used for functions and are also often repurposed

More specifically, we define the following mathematical symbols and operators

- \( \mathbb{R} \) is the set of real numbers
- \( \mathbb{Z} \) is the set of integers
- \( \mathbb{Z}^+ \) is the set of positive integers (natural numbers)
- \( N \) is the dimension of the input sample space (space of input patterns) of a feed-forward neural network, that is, the space of \( N \)-dimensional, real-valued column vectors, \( \mathbb{R}^N \)
- \( z' \) is the transpose of column vector \( z \), that is, \( z' \) is a row vector
- \( \langle z, y \rangle = z' y = \sum_{j=1}^{N} z_j y_j \) is the inner (dot) product of (column) vectors \( z, y \in \mathbb{R}^N \)
- \( \|x\|_q = \left( \sum_{i=1}^{N} x_i^q \right)^{1/q} \) is the \( l_q \)-norm (or \( q \) norm) of vector \( x \in \mathbb{R}^N \)
- \( \|x - y\|_q \) is the \( l_q \) distance between \( x \) and \( y \) of the same dimension
- \( \|x\| = \|x\|_2 = \sqrt{x' x} = \sqrt{\langle x, x \rangle} \) is the Euclidean (\( l_2 \)) norm of \( x \)
- \( x \odot m \) is element-wise multiplication of the vectors (or matrices) \( x, m \) resulting in another vector (or matrix), that is the \( i \)th element of \( x \odot m \), \( (x \odot m)_i = x_i m_i \)
- \( \mathcal{X} \) is the set of data samples that are used for training a neural network (deep learning), where \( \mathcal{X} \subset \mathbb{R}^N \)
- \( T = |\mathcal{X}| < \infty \) is the number of samples in the dataset \( \mathcal{X} \)
- \( K \) is the (finite) number of classes in \( \mathcal{X} \) for classification problems (but \( K \) has different meaning in the context of \( K \)-means clustering or \( K \)NN classification)
Notation

- $\mathcal{Y}$ is the set of classes in $X$, that is, $K = |\mathcal{Y}|$, for example, $\mathcal{Y} = \{1, 2, \ldots, K\}$
- $c(x) \in \mathcal{Y}$ is the true class label of $x \in \mathbb{R}^N$
- $\hat{c}(x)$ is the inferred class of input sample $x$ by a classifier
- $E(X) = E[X]$ is the expectation of random variable $X$
- $P(A) = P[A]$ is the probability of event $A$
- $\{x_1, x_2, \ldots, x_n\}$ is a set with $n$ elements
- $\{x_a | a \in A\} = \{x_a : a \in A\}$ is the set of elements $x_a$ such that $a$ parameter or index $a$ belongs to the set $A$ (here $x(a)$ or $x^{(a)}$ may be used instead of $x_a$ to indicate the dependence of $x$ on $a$)
- $A \cup B$ and $A \cap B$ respectively are the union and intersection of the sets $A$ and $B$
- $A \setminus B$ is the set of elements in the set $A$ that are not in the set $B$
- $\emptyset$ is the empty set
- $[a, b) = \{r \in \mathbb{R} : a \leq r < b\}$, with $b > a$, is a contiguous interval of real numbers including $a$ but not $b$
- $1\{\xi\} = I(\xi)$ is an indicator function, equal to one if the statement $\xi$ is true and zero if $\xi$ is false
- $I$ is a square identity matrix, with 1s on the diagonal and 0s off diagonal
- $a := b$ or $a \doteq b$ means $a$ equals $b$ by definition
- $\mathbf{0}$ is a vector all of whose entries are zero
- $\mathbf{1}$ is a vector all of whose entries are one
- $X \sim F$ means random vector $X$ has (multivariate) distribution $F$
- $\det(A) = |A|$ is the determinant of square matrix $A$
- $\Delta x$ is a change in the quantity $x$

List of Acronyms

- 3D: three-dimensional
- ACC: Accuracy (on a clean test/evaluation set)
- AD: Anomaly Detection (short name for I-PT-RED in Chapter 6)
- AI: Artificial Intelligence (often synonymous with a DNN)
- AL: Active Learning
- a.s.: almost surely (with probability one)
- ASR: Attack Success Rate
- AUC: Area Under the (ROC) Curve
- BA: Backdoor Attack (Trojan)
- BIC: Bayesian Information Criterion
- BP: Backdoor Pattern
- CDF or cdf: Cumulative Distribution Function
- CNN: Convolutional Neural Network
- CS: Cosine Similarity
- DNN: Deep Neural Network
- DP: Data Poisoning (attack)
- ET: Expected Transferability
• FPR: False Positive Rate (fraction or percentage)
• GAN: Generative Adversarial Network
• GMM: Gaussian Mixture Model
• HC: High Confidence
• i.i.d.: independent and identically distributed
• JSD: Jensen–Shannon Divergence
• KL: Kullback–Leibler divergence
• KNN: K Nearest Neighbors
• LC: Low Confidence
• LEM: Local Error Maximizer
• LeNet-\(n\): Learnable Neural Network architecture with \(n\) layers [140]
• LR: Logistic Regression
• LSTM: Long Short-Term Memory (a recurrent NN)
• MAD: Median Absolute Deviation
• MAE: Mean Absolute Error
• MAP: Maximum a posteriori
• ML: Machine Learning
• MLE: Maximum Likelihood Estimation
• MM: Mixture Model (or Maximum Margin in Chapter 9)
• MSE: Mean-Squared Error
• NB: Naive Bayes
• NN: Neural Network
• OOD: Out-Of-Distribution
• OODD: Out-Of-Distribution Detection
• pAUC: partial (ROC) Area Under the Curve
• PCA: Principal Component Analysis
• pdf: probability density function
• pmf: probability mass function
• PMM: Parsimonious Mixture Modeling [86]
• PT: Post-Training
• RE: Reverse-Engineering
• RE-AP: Reverse-Engineering Additive Perturbation
• RE-PR: Reverse-Engineering Patch Replacement
• REA: Reverse-Engineering Attack
• RED: Reverse-Engineering Defense
• ResNet-\(n\): Residual Neural Network architecture with \(n\) layers [97]
• RL: Reinforcement Learning
• ROC: Receiver Operating Characteristic
• SGD: Stochastic Gradient Descent
• SIA: Source-class Inference Accuracy
• SVD: Singular Value Decomposition
• SVM: Support Vector Machine
• TPR: True Positive Rate (fraction or percentage)
• TSC: Training Set Cleansing
xx Notation

- TTE: Test-Time Evasion (attack), that is, adversarial input
- WB: White Box
- XAI: eXplainable AI

The following list contains the “proper names” of some attacks and defenses used in this book, with bibliographic citations

- AC-GAN: Auxiliary-Classifier GAN based TTE detection [284, 285]
- ADA: Anomaly Detection of TTE Attacks [179]
- B3D: Black Box Backdoor trigger Detection [61]
- BIC-MM-TSC: BIC-MM based TSC against error generic DP [148]
- CI: Cluster Impurity defense [308]
- CIFAR-n: Canadian Institute for Advanced Research color image dataset with \( n \) object classes [129]
- CW: Carlini–Wagner TTE attack [33]
- FGSM: Fast Gradient Sign Method for TTE attacks [83]
- FP: Fine Pruning backdoor defense [156]
- i-FGSM or BIM: iterative-FGSM or Basic Iterative Method for TTE attacks [133]
- IF-RED: In-Flight backdoor trigger RED [149]
- JSMA: Jacobian based Saliency Map Approach for TTE attacks [203]
- KD: Kernel Density based defense [68]
- L-PT-RED: Lagrangian PT-RED [305]
- MD: Mahalanobis Distance based defense [142]
- MNIST: Modified National Institute of Standards and Technology dataset of handwritten digits [141]
- NC: Neural Cleanse backdoor detection [282]
- NC-M: NC based backdoor Mitigation [282]
- P-PT-RED: Perceptible backdoor PT-RED [304]
- PC-PT-RED: Point Cloud PT-RED against backdoors [306, 309]
- PGD: Projected Gradient Descent for TTE attacks [271]
- STRIP: STRong Intentional Perturbation backdoor trigger detection [73]
- T-PT-RED: Transferable PT-RED against backdoor DP [302]
- TSC-RED: Training dataset Cleansing RED against backdoor DP [301]
- UnivBD: “Universal” Backdoor Detection approach [286]
- UnivBM: “Universal” Backdoor Mitigation approach [286]
- ZOO: Zeroth Order Optimization based TTE attacks [41]