C02029: Doctor of Philosophy
Subject Code: 32903
February 2020

CRICOS Code: 009469A

Game theoretical adversarial deep learning algorithms for robust neural network models

Aneesh Sreevallabh Chivukula

Advanced Analytics Institute, School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

Game theoretical adversarial deep learning algorithms for robust neural network models

A thesis submitted in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy
in
Analytics

by

Aneesh Sreevallabh Chivukula

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

February 2020

Abstract

Despite recent advances in deep learning, there is still a significant gap between the robustness of human perception and machine intelligence. Deep learning is not provably secure. In fact, deep neural networks are vulnerable to security attacks from malicious adversaries, which is an ongoing and critical challenge for deep learning researchers. Even innocuous perturbations in training data can change the way a deep network behaves in unintended ways. This means that imperceptibly and immeasurably small departures from the training data can result in a completely different label classification when using the model for supervised deep learning.

In this thesis, we explore adversarial deep learning algorithms. We examine how they exploit vulnerabilities in deep networks and how to make deep networks robust to their attacks. To explore the vulnerabilities, we simulate various model training processes under a range of various attack scenarios. Each attack strategy is assumed to be formulated by an intelligent adversary that is capable of either feature manipulation, label manipulation, or both. The optimal attack policy of our adversaries is determined by the solution for optimization problems that output the adversarial data. We then apply the knowledge that we learned to improve and reinforce the learning procedure so as to better defend against attacks.

As part of this research process, we developed new adversarial learning algorithms to solve for adversarial manipulations in supervised classification networks such as Convolutional Neural Networks (CNNs). The adversarial learning objective for our adversaries is to inject small changes into the data distributions, defined over positive and negative class labels, to the extent that the CNN subsequently misclassifies the data distribution. Thus, the theoretical goal of our deep learning process becomes one of determining whether a manipulation of the input data has reached a learner decision boundary, i.e., where too many positive labels have become negative labels. We began this research undertaking by first studying the performance vulnerabilities in CNNs. With these vulnerabilities identified, we were able to propose CNNs that are secure to those types of adversarial attacks.

We generate adversarial data by solving for optimal attack policies in Stackelberg games where adversaries target the misclassification performance of CNNs. In a sequential game-theoretic formulation, we model the interaction between an intelligent adversary and a deep learning model (a CNN) to generate adversarial manipulations by solving a two-player sequential noncooperative Stackelberg game where each player's payoff function increases with interactions to a local optimum. With a stochastic game-

theoretic formulation, we then extend the two-player Stackelberg game into a multiplayer Stackelberg game with stochastic payoff functions for the adversaries. Both versions of the game are resolved through the Nash equilibrium, which refers to a pair of strategies in which there is no incentive for either the learner or the adversary to deviate from their optimal strategy. In this case, the strategy pair is a learner weight and an evolutionary operation. In addition, we devised each attack scenario as a blackbox attack where the adversaries have no prior knowledge of the CNN's learning processes and its best response strategies. In each case, we show that the Nash equilibrium leads to a supervised classification network that is robust to subsequent data manipulation by a game-theoretic adversary.

We then explore adversaries who optimise variational payoff functions via data randomisation strategies on CNNs designed for multilabel classification tasks. Similarly, the outcome of these investigations is an algorithm design that solves a variable-sum two-player sequential Stackelberg game with new Nash equilibria. In designing the attack scenarios, the adversarial objective was to make small, undetectable changes to the test data. The adversary manipulates variational parameters in the input data to mislead the learning process of the CNN, so it misclassifies the original class labels as the targeted class labels. Our ideal variational adversarial manipulation is the minimum change needed to the adversarial cost function of encoded data that will result in the CNN incorrectly labelling the decoded data.

The resulting attack algorithms were a Simulated Annealing (SA) algorithm and an Alternating Least Squares (ALS) algorithm, which optimise the data manipulations by stochastically searching the strategy spaces of the variational adversaries. The optimal manipulations are found by solving the Nash equilibria optimization problem of the proposed Stackelberg game. Specifically, the optimal attack parameters in ALS and SA solve for the (variational nonlinear non-convex) adversarial cost functions that generate adversarial data. The payoff function for the variational adversary depends on the manipulations determined by a Variational Autoencoder (VAE), while the payoff function for the CNN classifier is evaluated in the input data space.

The adversarial data generated by this variant of the Stackelberg games simulates continuous interactions with the classifier's learning processes as opposed to one-time interactions. To evaluate CNN performance, we assessed the adversarial algorithms over different strategy spaces proposed for the MNIST handwritten digits database and the VGGFace2 database of human faces.

The original CNN model was then retrained using all the adversarial manipulations generated by the game-theoretic adversaries to create a secure CNN model. In an empirical demonstration, we show that the new secure CNN model is robust to subsequent game-theoretic data manipulation by adversaries. This promising result suggests that evolutionary algorithms based on game-theoretic modelling and mathematical optimization are significantly better approach to building more secure deep learning models.

We also empirically demonstrate that variational adversaries are also able to mislead CNNs. In these cases, the learning process of the CNNs was manipulated by an adversary at the input data level as well as the generated data. The optimal manipulations were

to stochastic optima in non-convex best responses strategies. We were able to encode the resulting adversarial data in terms of the multivariate statistical parameters of a Gaussian mixture model. We then retrained the original CNN on the manipulated data to give rise to a secure CNN that is robust to subsequent performance vulnerabilities from variational adversaries.

In applying this research, we developed a deep network model that discovers Granger causes in multivariate temporal financial market data. The model comprises a deep neural network (DNN) and a recurrent neural network (RNN) and discovers Granger-causal features with bivariate regression from bivariate time series data distributions. These features are subsequently used to discover Granger-causal graphs for multivariate regression on multivariate time series data distributions. Our supervised feature learning process with these proposed deep regression networks returned favourable F-tests for feature selection and t-tests against comparisons with other models. Moreover, a regression analysis on a set of experiments with real stock market data from Yahoo Finance demonstrates that our causal features are a significant improvement over the existing deep learning regression models in terms of minimising root mean squared error.

CERTIFICATE OF AUTHORSHIP

I, *Aneesh Chivukula* declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science*, *Faculty of Engineering & Information Technology* at the University of Technology Sydney, Australia, is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note: Signature removed prior to publication.

[Aneesh Sreevallabh Chivukula]

DATE: 10th February, 2020

PLACE: Sydney, Australia

Dedication

To my parents . . .

कारणाभावात् कार्याभावः ।

The absence of the effect is due to the absence of the cause.

Vaisesika Sutra 1.2.1 enunciating the Causality Principle in Indian Philosophy.

ACKNOWLEDGMENTS

I want to thank my principal supervisor, Dr. Wei Liu for his strong support and guidance throughout my degree. His scientific expertise and continuous encouragement of my research have been essential for my achievements. I also want to thank my co-supervisor, Dr. Jun Li who organized a friendly research environment and supported my casual academics at UTS.

I would like to thank Capital Markets CRC Limited (now RoZetta Institute) for providing me with a PhD scholarship. I would like to thank Australian Mathematical Sciences Institute for providing me with a postgraduate research internship.

Finally, I want to thank my wife for her support throughout my PhD.

I acknowledge Chandranath Adak (UTS) for providing this thesis template.

List of Publications

Related to the Thesis :

1. A. S. Chivukula and W. Liu, "Adversarial learning games with deep learning models," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 2758-2767. doi: 10.1109/IJCNN.2017.7966196

2. A. S. Chivukula and W. Liu, "Adversarial Deep Learning Models with Multiple Adversaries," in IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 6, pp. 1066-1079, 1 June 2019. doi: 10.1109/TKDE.2018.2851247

3. Chivukula A.S., Li J., Liu W. (2018) Discovering Granger-Causal Features from Deep Learning Networks. In: Mitrovic T., Xue B., Li X. (eds) AI 2018: Advances in Artificial Intelligence. AI 2018. Lecture Notes in Computer Science, vol 11320. Springer, Cham doi: 10.1007/978-3-030-03991-2_62

4. A. S. Chivukula, X. Yang and W. Liu, "Adversarial Deep Learning with Stackelberg Games," accepted for presentation at the 26th International Conference on Neural Information Processing (ICONIP 2019) of the Asia-Pacific Neural Network Society.

5. A. S. Chivukula, X. Yang and W. Liu, "Game Theoretical Adversarial Deep Learning with Variational Adversaries," in IEEE Transactions on Knowledge and Data Engineering (TKDE), doi: 10.1109/TKDE.2020.2972320.

Others :

6. Prabhu, C.S.R., Sreevallabh Chivukula, A., Mogadala, A., Ghosh, R., Livingston, L.M.J. 2019. Big Data Analytics: Systems, Algorithms, Applications. Springer Nature Singapore Pte Ltd.

Table of Contents

LIST OF FIGURES

FIGURE **Page**

List of Tables

CHAPTER 1

INTRODUCTION

Data mining is the study of automatically learning mathematical patterns from the information in a database. It is a process of knowledge discovery that requires developing computational algorithms [74] for preprocessing, modelling, and postprocessing data given a database system. The design of those algorithms, however, must be based on a machine learning paradigm. Machine learning paradigms are modes of computational learning based on some underlying statistical assumptions, such as the level of human oversight in the training data or the data's underlying distribution. Example paradigms include supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, meta-learning, and deep learning.

A standard statistical assumption, called the stationarity assumption, is that the training data used by a model to learn a mathematical pattern and the testing data used to evaluate how well it recognises those patterns are sampled from the same underlying probability distribution of independent and identically distributed (i.i.d) random variables. Yet the stationarity assumption does not hold in most real-world applications; training and testing data seldom share exactly the same distributions and are not often i.i.d. Therefore, a robust learning paradigm for non-stationary data analytics has become something of a goal in adversarial learning. Adversarial learning has applications in areas like spam filtering, virus detection, intrusion detection, fraud detection, biometric authentication, network protocol verification, computational advertising, recommender systems, social media web mining, complex system performance modelling, and so on [21, 30].

Adversarial learning algorithms are specifically designed to exploit vulnerabilities in a given machine learning algorithm. These vulnerabilities are simulated by training the learning algorithm under various attack scenarios and policies. The attack scenarios are assumed to be formulated by an intelligent adversary [110], and the optimal attack policy is one that can solve one or many optimization problems over one or many attack scenarios, noting that various adversarial learning algorithms may differ in their statistical assumptions over the adversary's knowledge, security violations, attack strategies, and attack influences [34].

As such, a learning algorithm that has been designed to offset an attack becomes robust to that attack; its vulnerabilities are no longer vulnerable. Thus, the goal of adversarial learning can be thought of as one of finding solutions for the objective functions in search and optimization algorithms that defend against attack scenarios. Once found, these solutions can be incorporated into the design of many machine learning algorithms as defence mechanisms to guard against attack.

1.1. Thesis and Contributions

In this thesis, we formulate and customise adversarial loss functions and stochastic optimization algorithms for adversarial learning objective functions. The learner is assumed to be a Deep Neural Network (DNN), such as a Convolutional Neural Network (CNN), in a discriminative learning setting. We then devise adversarial learning algorithms that manipulate DNNs as an adversary would. After demonstrating the vulnerability of DNNs in this way, we propose defence mechanisms to make the DNNs more robust to these kinds of attacks.

Deep learning refers to a particular class of neural network algorithms. These algorithms consist of many stages of nonlinear information processing in hierarchical architectures exploited for pattern classification and feature learning [68]. Deep learning research aims to discover machine learning algorithms at multiple levels of data abstraction.

Deep learning with high-dimensional data has been found to be susceptible to adversarial attacks. Such attacks are crafted by prior knowledge, observation, and experimentation on the loss functions in the deep learning models [89]. A systematic investigation into the design of deep learning loss functions to defend against adversaries is a novel and practical area of research. Furthermore, statistical error analyses of the data-driven loss functions must consider conflicting optimization goals in the

models under attack, such as accuracy, scalability, runtime, and diversity, defined over underlying data distributions.

Loss functions have been defined in the context of multiple machine learning paradigms applicable to deep learning. In supervised learning, loss functions are defined as fitting criteria for class probability estimation [46]. In statistical learning, loss functions are defined as minimizers of Empirical Risk in training data [198]. In computational learning, loss functions are said to minimize Bayes decision rule for predictors by computing the expected probability of classification error [165]. Energy-Based Learning Models [140, 142] are a theoretical framework for statistical inference and computational learning characterized by loss functions.

Specific to supervised learning applications, loss functions evaluate the statistical error of predictive analytics. Typically, the loss function reduces bias in a predictive classification model and variance in a predictive regression model [40]. Here, adversarial loss functions reduce the predictive model's sensitivity to model noise.

Our proposal is to analyse this type of noise in game-theoretic adversarial deep learning with adversarial payoff functions and to do the same in time series predictions with regression loss functions and Granger causality. Through stochastic optimizations of adversarial learning, we empirically analyse a classification loss function to generate misclassified data points. Our adversarial payoff functions are non-differentiable and discontinuous over the search space of the adversarial manipulations, and the adversarial manipulations are generated by optimising the adversarial payoff functions for both evolutionary and variational adversaries.

The intuition of our loss functions is derived from the concept of actions and moves in game theory. During learning, the attack scenarios are modelled as moves made by a learning algorithm and countermoves made by an intelligent adversary. Our game theory studies interactions between independent self-interested agents or players working toward a goal. Each player has a set of associated strategies/moves/actions that optimise a payoff function or utility function for achieving the goal. The game eventually converges to an equilibrium state from which none of the players have any incentive to deviate.

We design Stackelberg games between two players where one of two players, a follower data miner (learner), acts in response to the moves of the other player, an intelligent adversary (adversary) leader, with the goal of converging on an equilibrium state called Nash equilibrium. We have formulated the objective functions in (two-player sequential) Stackelberg (zero-sum) games as bi-level optimization problems, where the game-theoretic goal is defined to simultaneously optimise two payoff functions that

influence one another according to a leader-follower interaction that allows the learner to retrain after each attack. Stochastic search algorithms are then designed to to solve the optimization problems.

The optimal attack policy is thus formulated in terms of stochastic optimization operators and evolutionary computing algorithms. The adversary's payoff function simulates their attack processes, and the learner's simulates its learning processes. The solution to attack processes specifies the adversary's optimal attack policy under constraints. The solution to learning processes specifies the learner's gain given the adversary's gain under the optimal attack policy.

The proposed adversarial loss functions are particularly suitable for deep learning with image classification networks and time series regression problems with Granger causality features. Therefore, in this thesis, we study classification loss functions, regression loss functions, and adversarial loss functions as special cases of an empirical risk minimisation framework for supervised learning and game theory.

Following are the major contributions of this research:

- *Loss functions in Adversarial Machine Learning:* We formulate the problem of finding attack scenarios in adversarial learning and defence mechanisms in deep learning as optimization problems rooted in game-theoretic learning models.
- *Attack scenarios in Adversarial Learning Frameworks:* We formulate two-player games and multiplayer games that simulate adversarial manipulations as adversarial costs and learning errors associated with training and testing data distributions.
- *Game theoretical adversarial deep learning:*
  - We evaluate adversarial manipulations with a tensor-based fitness function and corresponding payoff functions in a game setting.
  - We have developed new search and optimization algorithms that solve for adversarial manipulations in attack scenarios of adversarial learning. Moreover, our game formulation is not sensitive to the choice of search algorithm.
- *Adversarial Examples for Misleading Deep Classifiers:*
  - We demonstrate the solutions to adversarial manipulations through evolutionary algorithms that execute game strategies through stochastic operators.
  - Given a classification model under attack, our algorithm produces a secure learner that is immune to data manipulation by an adversary. Plus, upon

game convergence, the learner can retrain to find weights that are robust to subsequent adversarial attacks.

  - Unlike most existing adversarial learning algorithms, our algorithms can adapt to constant adversarial data manipulations, nor do we assume that the adversary knows anything about the deep network structure, which is more realistic.

- *Generative Adversarial Networks for Adversarial Learning:*

  - We propose to formulate the game strategy space for the adversary as a stochastic optimization problem using autoencoder networks and propose a new SA algorithm to optimise the adversary's payoff function in the game's randomized strategy space.
  - We formulate new attack strategies and adversarial algorithms in game theoretical adversarial learning with variational adversaries. The purpose of our attack scenarios is to mislead CNNs in both two-label and multi-label image recognition settings.
  - We derive adversarial manipulations in the adversary's strategy space based on the raw input data. These manipulations are able to create a generative model for adversarial data found in the CNN's learning space. The effectiveness of the adversary's manipulations is verified through cross-validation experiments on both the MNIST handwritten digits database and the VGGFace2 human faces database.

- *Stochastic Games in Predictive modelling:*

  - We demonstrate the effectiveness of evolutionary adversarial manipulations in both two-player and multiplayer Stackelberg games with stochastic adversaries operating on the MNIST dataset. We also show that both CNNs and generative adversarial networks (GANs) are vulnerable to manipulation by adversaries.
  - We build a secure CNN classifier that is immune to multi-label data manipulation by adversaries based on the Nash equilibrium from our (variable-sum two-player sequential) Stackelberg game model with variational adversaries. Upon convergence, the CNN classifier can find weights that are robust to targeted attacks.

- *Computational optimization algorithms to find Nash equilibria and Stackelberg Strategies:*
  - We use the SA and ALS algorithms to search for adversarial manipulation in the form of randomization parameters of a Variational Autoencoder (VAE). We then optimise these manipulations within a Stackelberg game with the goal of misleading two-label and multi-label classifiers. The stochastic optimization problem is expressed by our proposed adversarial payoff functions.
  - From the Nash equilibrium in our Stackelberg game, we find that a manipulated CNN classifier would be vulnerable to attack. Therefore, we propose a secure CNN classifier that is robust to adversaries. Our attack data is found on class labels of the MNIST database and the VGGFace2 database. Our proposed CNN is also able to adapt to continuous, as opposed to one-time, attacks simulated by a variational adversary.
  - We demonstrate the effectiveness of variational adversarial manipulations using CNNs and, additionally, benchmark the proposed adversarial manipulations against other deep learning baseline methods, including GANs and other game-theoretic adversarial learning models. Our adversarial manipulations also successfully mislead the baseline models.
- *Causal Inference in Deep Learning:* We identify Granger-causal features using deep networks that improve bivariate regression predictions among temporal dependencies in time series distributions.
- *Causal Inference in Time Series Analysis:* We discover Granger-causal graphs in time series distributions to improve multivariate regression in deep networks.
- *Causal Inference in Financial Markets:* We present a theoretical model for solving causal inference problems by defining the stochastic processes found in financial markets and evaluate its efficacy on Yahoo Finance data.

1.2. Research Methods, Objectives and Significance

The primary outcome of this thesis is a set of adversarial learning algorithms that are useful for knowledge discovery in DNNs and big datasets. The significance of this thesis lies in the future computational applications and intelligent systems that could be driven by these algorithms. Specific application scenarios would involve both systemic

and algorithmic aspects of exploratory data analysis, and descriptive, predictive, and prescriptive analytics [123, 161, 214] in data and network science. Common application domains for adversarial learning with intelligent systems include cloud technologies, the Internet of Things, fog computing, mobile internet, social web mining, eCommerce, recommender systems, medical and biological data analysis, next-generation genomics, and renewable energy [6, 195]. Application designs in these areas commonly have adhoc architectures and patterns that are based on criteria for success, which are specific to that particular domain.

In general, the results of this research will be of interest to researchers working on tensor analysis, causal inference, predictive analytics, stochastic optimization, evolutionary learning, game theory, data engineering and big data. The algorithms presented should lead to further research and applications in supervised learning, adversarial learning, deep learning, data mining, and knowledge discovery.

We had the following broad objectives when designing and analysing the adversarial learning algorithms introduced in this thesis [51]:

- To propose and validate new computational algorithms for adversarial learning in deep learning models.
- To propose new adversarial security mechanisms within game-theoretic learning theory.
- To develop optimization methods for use within adversarial learning that are driven by constrained objective functions.
- To adapt the proposed models for adversarial learning with big data in transactional and sequential databases.
- To develop new payoff and loss functions for adversarial learning algorithms.
- To develop new data mining algorithms for feature and representation learning.
- To benchmark the statistical significance tests and tuning parameters needed for identifying the best fitting model for a given dataset with the proposed algorithms.
- To conduct experiments with classification and feature extraction in multi-label datasets.
- To test and evaluate the outcomes of this research with the state-of-the-art deep learning models.

This thesis is partially funded by a CMCRC PhD Scholarship in the Education Program of the Capital Markets Cooperative Research Centre. It is also partially funded by the Australian Postgraduate Research (APR) STEM Internships program for short-term industry and university research collaborations. As part of these scholarship agreements, the research in this thesis will be used to produce data mining models and recommender systems for capital market applications.

1.3. Report Organisation

This report begins with a review of related work in Chapter 2 along with a comparison of new versus existing approaches to game theoretical adversarial deep learning. Chapter 2 positions our contributions in contrast to related literature on (i) Adversarial Security Mechanisms and Generative Adversarial Networks, (ii) Adversarial Examples for Misleading Deep Classifiers and Game Theoretical Adversarial Deep Learning Models, and (iii) Computational optimization algorithms for Derivative-free Stochastic Optimization and Spam Filtering application.

The problems formulation and games illustration is presented in Chapter 3, Chapter 4 and Chapter 5. The adversarial algorithms and experimental evaluations for proposed games are presented in Chapter 3, Chapter 4 and Chapter 5. Chapter 6 presents an example application of the theories and algorithms developed in the finance arena. The dissertation ends in Chapter 7 with a summary of current and future work.

Part I

CHAPTER 2

RELATED WORK

Adversarial learning problems in discriminative learning models are typically formulated with statistical divergence metrics between training data features and adversarial data features. The latent space on high dimensional training data can also be searched by deep networks to construct adversarial examples. Depending on the goal, knowledge and capability of an adversary, adversarial examples can also be crafted by prior knowledge, observation, and experimentation on the loss functions in deep learning.

Thus the existing adversarial learning algorithms differ in design assumptions regarding adversary's knowledge, attack strategies, attack influence, and security violation. Furthermore, adversarial examples are known to transfer between data-specific manifolds of machine learning models. Therefore the predictive performance of deep learning models under attack is an interesting area for research.

In this chapter, we shall conduct a literature review to provide new insights on the relation between adversarial learning and cybersecurity. We seek to survey and summarize non-stationary data representations learnt by deep learning networks. The robustness of deep learning networks shall be surveyed to produce a taxonomy of adversarial examples and adversarial algorithms. We shall also survey the use of convex optimization, stochastic optimization and evolutionary computing in adversarial deep learning formulations. Another interesting study shall be that of defence mechanisms available for deep learning models deployed in real world environments.

In general we are interested in adversarial deep learning applications in cyber-physical systems and transfer learning, big data and dictionary learning, kernel machines

and robust learning, and reinforcement learning and game theory.

2.1. Adversarial Machine Learning

Traditional machine learning models assume training data samples, testing data samples and validation data samples follow the same, independent and identically distributed data distribution. This assumption creates security vulnerabilities in machine learning models subject to attack from intelligent adversaries with a malicious intent. Given training data samples, such adversaries design adversarial examples to increase model error. Securing intelligent systems from such adversarial examples is an active area of research in artificial intelligence, security diagnostics, generative learning, deep learning, information security, autonomous systems, intelligent systems and data analytics.

Adversarial examples can mislead learning models as long as adversary's attack is planned after learning model has completed training and therefore cannot react to new samples. From this observation, adversarial algorithms incorporate adversary into training process of learning models. Thus, adversarial algorithms model adversarial machine learning as an interaction between two agents – the learning model and one or more intelligent adversaries.

Game theory provides a framework to study interactions between learning model (or learner for short) and intelligent adversary (or adversary for short) in terms of interaction between evolving strategies of the learner and the adversary. Game theory interactions were first formulated in life sciences as nonlinear differential equations that study interactions between populations of biological systems.

In machine learning, loss functions quantify the impact of information uncertainty over a distribution of analytics predictions. Adversarial algorithms formulate machine learning loss functions for a training process that prevents model overfitting to training data in presence of rational, adaptive adversaries that simulate evolving changes to learning environment as adversarial examples.

In game theoretical adversarial learning, adversarial examples are generated by designing machine learning algorithms under various attack scenarios in adversary's strategy space. Optimal attack strategy for adversarial manipulation is formulated as solution to (often non-linear and non-convex) optimization problems.

Adversarial examples are hard to detect because machine learning models trained on limited data are required to produce expected output for every possible input. Reinforcement learning agents can also be manipulated by adversarial examples to result in

degraded agent performance in the presence of perturbations too subtle to be perceived by a human.

In the following literature review, we provide an overview of the existing adversarial machine learning algorithms, each differing in attack scenarios and defense mechanisms for deploying reliable data analysis systems and robust pattern recognition systems. We also summarize the state-of-the-art techniques in game theoretical adversarial learning and adversarial reinforcement learning for software based inference and decision making.

Adversarial algorithm	Cost function	Search algorithm	Convergence conditions	Attack strategy	Attack influence	Security violation	Adversary's knowledge	Algorithm moves	Learning games
Classifier ensembles [32]	(2.1) $E = 2 - \frac{2}{n}\Sigma_{k=1}^{n} F(k), F(k) = \frac{\Sigma_{i=1}^{k}\lvert w_i\rvert}{\Sigma_{j=1}^{n}\lvert w_j\rvert}.$	randomized sampling	ensemble size, feature subset size	reorder features by importance for discriminant function	causative	targeted, availability	training features	average discrimination functions	none
Feature weighting [130]	(2.2) $min_w \frac{1}{2}w^T w + \frac{1}{m}\Sigma_{i=1}^{m} l(w^T(S^{-1}x), y), l(f, y) = max(0, 1 - yf).$	feature bagging	number of base models	addition/deletion of binary features	causative	indiscriminate, availability	training features	average estimated weights	none
SVM : inputs [36]	(2.3) $max_{x_c} L(x_c) = \Sigma_{k=1}^{m}(1 - y_k f x_c(x_k))\frac{\partial L}{\partial u} = \Sigma_{k=1}^{m} M_k \frac{\partial Q_{kk}}{\partial u} + \frac{\partial Q_{kc}}{\partial u}\alpha_c.$	gradient ascent	change in test error	train noise injection	causative	targeted, integrity	gradient of loss	incremental learning svms	none
SVM : labels [241]	(2.4) $L(D_{tr}) = argmin_{f \in F}[\Omega(f) + C.\hat{R}(f, D_{tr})],$ $\hat{R}(f, D_{tr}) = \frac{1}{n}\Sigma_{i=1}^{n} l(f(x_i), y_i), C > 0,$ $V_L(D_{tr}, D_{vd}) = f_{D_{Tr}}^2 + C.\hat{R}(f_{D_{tr}}, D_{vd}),$ $f_{D_{Tr}} = L(D_{tr}), V_L(z, y) = V_L((x_i, z_i), (x_i, y_i)).$	gradient ascent	svm margin support vectors by lp and qp	label noise injection	causative	targeted, integrity	training labels	update svm weights and hyperplanes	none
Deep learning [91]	(2.5) $\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha)J((\theta, x + \epsilon sign(\nabla_x J(\theta, x, y))).$	backpropagation with L-BFGS	early stopping on adversarial validation set error	linear perturbation on x	causative	targeted, integrity	training and testing data	update decision function parameters	none
Adversarial networks : DNN [183]	(2.6) $S_{p+1} = x + \lambda_{p+1} sgn(J_F[\hat{O}(x)])US_p,$ $\hat{O}(x) = argmax_{j=0\ldots N-1} O_j(x),$ $\delta_x = \epsilon sgn(\Delta_x c(F, x, y)),$ $c(F, x, y) = p(y = 1\lvert x) = exp((x - \mu)^T \beta(x - \mu)),$ $F(x) = f_n(\theta_n, f_{n-1}(\theta_{n-1}, \ldots f_2(\theta_2, f_1(\theta_1, x)))).$	Jacobian-based dataset augmentation	early stopping on adversarial validation set error	observe DNN outputs given inputs chosen by the adversary	exploratory	targeted, integrity	testing data	Jacobian-based regularization and distillation	none

Table 2.1: Adversarial Algorithms Comparison 1

Adversarial algorithm	Cost function	Search algorithm	Convergence conditions	Attack strategy	Attack influence	Security violation	Adversary's knowledge	Algorithm moves	Learning games
Adversarial networks : DAE [96]	(2.7) $J_{DCN}(\theta) = \Sigma_{i=1}^{m}(L(i^i, y^i) + \Sigma_{j=1}^{H+1}\lambda_j \|\frac{\partial h_j^i}{\partial h_{j-1}^i}\|_2)$ $min_r c\|r\|_2 + L(x+r, l).$	stacking DAEs into a feed forward neural network	training error	Gaussian additive noise	exploratory	indiscriminate, availability	testing data	penalty function smoothing the adversarial data	none
Game theory : regularized loss [150]	(2.8) $max_a min_w -\lambda a^T a + \lambda w^T w + C\Sigma_{i=1}^{n} Loss(y_i, w^T(x_i + a)).$	trust region	adversary's payoff does not increase or the maximum number of iterations is reached	move positive data towards negative samples	causative	targeted, availability	training and testing data	rebuild classifiers though his regularized loss function	Zero sum game
Game theory : sparse attacks [231]	(2.9) $min_{w \in R^d} \|y - Xw\|_2 + \Sigma_{i=1}^{d}\lambda_i \|w_i\|_1.$	minimum budget	accumulated cost more then minimum budget	select features affect estimated weights	causative	indiscriminate, privacy	estimated weights	robust regularization	Non-zero sum game
Game theory : support vector machines [86]	(2.10) $min \frac{1}{2}\|w\|^2 + C\Sigma_i[1 - y_i w^T x_i + t_i]$ $t_i \geq Kz_i + \Sigma_j v_{ij}$ $t_i \geq Kz_i + \Sigma_j v_{ij}$ $v_i \geq 0$ $z_i + v_i \geq (y_i x_i w).$	quadratic programming	training error subject to regularization terms	delete different features from different data points	causative	targeted, integrity	training features	parameters regularization	Non-zero sum game
Game theory : deep learning (Our method)	(2.11) $Maxmin : (a^*, w^*) = argmax_{a \in A} J_L(a, argmin_{w \in W} J_L(a, w)).$	Evolutionary algorithm	Nash equilibrium	Move positive samples towards negative samples	causative	targeted, integrity	Training data	Update estimated weights for adversarial manipulation	Constant sum game

Table 2.2: Adversarial Algorithms Comparison 2

2.1.1 Adversarial Security Mechanisms

This section presents a literature review and attack taxonomy of adversarial learning algorithms. The adversarial algorithms are summarized in Table 2.1, Table 2.2 in terms of algorithm design and algorithm application. The algorithms are primarily compared on the adversarial cost function (or cost function for short). It is a measure of the expected performance of the learning algorithm in the presence of an adversary. It is formulated differently for different adversarial learning algorithms. The tables' columns list the various features for comparing adversarial learning algorithms. Our algorithm is termed 'Game theory : deep learning'. The tables' rows list the various algorithms under comparison. Across the rows, we list computational models vulnerable to adversarial data for feature extraction, deep learning, support vector machines, classifier ensembles where the input data for simulating adversarial attacks is taken to be text spam, image spam and biometric spam. The algorithms are compared on cost function, search algorithm, convergence conditions, attack strategy, attack influence, security violation, adversary's knowledge, algorithm moves, learning games. The 'cost function' is the objective function to solve for adversarial data. The 'search algorithm' is the algorithm used to find an optimal solution. The 'convergence conditions' is the search criteria for creating adversarial data. The 'attack strategy' is the attack scenario under which the adversary operates. The 'attack influence' of a strategy determines the access that the adversary has to train data and test data input to the learning algorithm. The 'security violation' is the purpose of the adversary's attack. The 'adversary's knowledge' is the semantic information of the adversary. The 'algorithm moves' are the actions taken by learning algorithm to adapt to adversarial data manipulation. From the tables, we see that most of the existing research work do not add game theory formulations to the cost function. Thus most of the existing learning algorithms cannot adapt to continuous adversarial data manipulations. As shown in the column 'Learning games', it is the only adversarial learning algorithm that has a game theory based formulation of training and testing data distributions input to deep learning models.

In addition to Table 2.1 and Table 2.2, the existing adversarial learning algorithms and their application domains can also be classified by the learner's defence mechanisms and corresponding adversarys' attack scenarios [22, 32, 34, 203]. Learner's defence mechanisms have been proposed by designing secure learning algorithms [34], multiple classifier systems [32], privacy-preserving machine learning [203] and use of randomization or disinformation to mislead the adversary [22].

Biggio *et al.* [34] discuss learner's defence mechanism in terms of an empirical

framework extending the model selection and performance evaluation steps of pattern classification by Duda *et al.* [70]. The framework recommends training the learner for 'security by design' rather than 'security by obscurity'. The framework recommends following additional steps to validate the defence mechanisms proposed in case of both generative learning models and discriminative learning models under attack.

- Pro-actively anticipate the most relevant adversarial attacks through a what-if analysis simulating potential attack scenarios.
- Define attack scenarios in terms of goal, knowledge, and capability of adversary
- Propose a generative data distribution model on conditional probabilities that can formally account for a large number of potential attacks and cross-validation samples on training data and testing data

Following assumptions are made regarding the learning algorithm's security. The model performance is then evaluated under an optimal attack strategy simulated according to the framework proposed by Biggio *et al.* [34].

- An adversary's goal is formulated as the optimization of an objective function. The objective function is designed on the desired security violation (that is integrity, availability, or privacy) and attack specificity (from targeted to indiscriminate).
- An adversary's knowledge is defined as knowledge of the components of the classifier viz training data, feature set, learning algorithm, decision function and its parameters, available feedback.
- An adversary's capability is defined as the control adversary has on training data and testing data taking into account application-specific constraints such as attack influence (either causative or exploratory), affect on class priors, fraction of samples and features manipulated by adversary.

Depending on the goal, knowledge and capability of the adversary, these assumptions are also classified in terms of attack influence, security violation and attack specificity.

The attack influence can be causative or exploratory. Causative attack affects both training and testing data. Exploratory attack affects only testing data.

The security violation can target either integrity or availability or privacy of the learner. A machine learning algorithm whose integrity is compromised cannot detect malicious behaviour of the adversary. The integrity of an algorithm with many false

negatives gets compromised. A machine learning algorithm whose availability is compromised exhibits severely degraded performance for legitimate users. The availability of an algorithm with many false positives gets compromised. The privacy of an algorithm whose detailed feedback is made public also gets compromised.

The attack specificity can be either targeted or indiscriminate for attacks that influence prediction or action of the algorithm. In targeted attacks the attack is directed at only a few instances of the training or testing data. In indiscriminate attacks the attack is directed at an entire class of instances or objects.

Our adversarial algorithms have causative attack influence, integrity security violation, targeted attack specificity.

The typical adversary's attack scenarios range across (i) adding noise to features/labels, (ii) adding/deleting features/labels, (iii) slight change or manipulation or perturbation to data distributions and (iv) slight change to decision boundaries. The corresponding optimization problems are solved using search algorithms with sampling and gradients methods. The sampling methods range across incremental sampling, bagging sampling, stacking sampling and randomized sampling. The gradient methods range across linear methods, quadratic methods, convex methods and stochastic methods. These optimization problems are solved on finding a local optimum solution determined by convergence conditions ranging across (i) number of features, (ii) number of regularization terms and (iii) changes to estimated errors over training/testing data.

Our adversarial algorithms cause slight change to data distributions simulated by stochastic optimization and randomized sampling methods. Our optimization problems converge onto solutions computed at Nash equilibria in Stackelberg games. From the adversary's standpoint, the equilibrium solution is a local optimum in case of worst-case attack scenarios and a global optimum in case of best-case attack scenarios. The strength and relevancy of our attack scenarios is determined by the performance of the deep learning models under attack.

2.1.2 Adversarial Learning Frameworks

Papernot *et al.* [185] provide a threat model summarizing various attack scenarios in adversarial learning algorithms. The adversarial classifier's defense mechanisms are then supposed to improve model robustness to its validation data samples. Here, validation data samples are deployed into the trained model's runtime data distribution to be non-iid with respect to testing data samples in trained model's training data distribution.

Papernot *et al.* [185] express their machine learning threat model in steps of adversarial manipulations found during machine learning training process and machine learning inference process. During machine learning training process, adversary is supposed to manipulate either online data collection processes or offline data collection processes. Such an adversarial manipulation either injects adversarial examples or modifies training data with intent of modifying learning model's decision boundaries. During machine learning inference process, adversary is supposed to plan either blackbox attacks or whitebox attacks on learning model's parameters. Such attack settings cause distribution drifts between training and runtime data distributions.

Papernot *et al.* [185] also view machine learning security through the prism of confidentiality, integrity, and availability models where adversary targets classifier's parameters, labels and features respectively. In contrast to machine learning security, machine learning privacy is explored in terms of model performance when (i) training and runtime data distributions differ, (ii) amount of data exposed by learning model is bound by a differential privacy budget, and (iii) learning model's defenses provide fairness, interpretability and transparency to learning outputs. Adversarial environments affecting model complexity, model accuracy and model resilience are formulated in terms of no free lunch theorem for adversarial learning. Papernot *et al.* [185] also motivate game theoretical adversarial learning during machine learning inference within a Probably Approximately Correct (PAC) learning framework.

Biggio *et al.* [39] survey adversarial machine learning for pattern classifiers. The adversarial examples for pattern classifiers are supposed to be created at either training time or testing time. Recent research in adversarial examples for deep networks applications in computer vision and cybersecurity is also discussed. Attack scenarios at training time are called poisoning attacks while attack scenarios at testing time are called evasion attacks. To integrate with deep learning terminology, poisoning attacks are also called adversarial training attacks while evasion attacks are also called adversarial testing attacks. Then security evaluation and defense mechanisms of pattern classifiers under attack are discussed. Here, a proactive security-by-design learning model incorporating adversary designs in learning process is also presented.

Biggio *et al.* [39] categorize adversary designs as optimization problems solving for best attack strategy defined by adversary's goal in attack scenario, adversary's knowledge of targetted learning system and adversary's capability of manipulating input data. Under various assumptions on such adversary designs, optimal attack strategies are then shown to be possible for not only supervised learning algorithms

but also unsupervised learning algorithms such as clustering algorithms and feature selection algorithms. Adversary's goal is further categorized into (i) security violation that compromises one of integrity, availability, privacy of learning system (ii) attack specificity and error specificity that cause misclassification of specific set of samples and specific set of classes respectively. Here, adversary's knowledge of targeted learning system is further categorized into following

- **Perfect-Knowledge whitebox attacks** with complete knowledge of learning parameters. In this case, security evaluation provides upper bound on performance degradation in attack scenario.

- **Limited-Knowledge Gray-Box attacks** with prior knowledge about feature representation and learning algorithm but not training data and learning parameters. Here security evaluation is conducted on a surrogate classifier learning a surrogate dataset available from similar data source as training data. Adversarial examples for surrogate classifier are then tested against targeted classifier to evaluate transferability of attack scenarios between learning algorithms.

- **Zero-Knowledge blackbox attacks** without any knowledge of learning algorithm but partial knowledge of feature representation and training data distribution. Here security evaluation checks whether optimal attack strategy transfers between a optimally trained surrogate model and targeted classifier model. Reinforced feedback on classifier decisions can be used to refine surrogate model.

Biggio *et al.* [39] also categorize adversary's knowledge by application-specific data manipulation constraints on input data distributions, features and classes. A high-level formulation of adversary's optimal attack strategy and classifier's security evaluation curves is also provided. Such a security evaluation considers both differentiable and non-differentiable learning algorithms like neural networks and decision trees respectively. Here sensitivity analysis of deep networks is defined as a study of the phenomenon of minimally perturbing training samples, whereas a more general security evaluation of pattern classifiers is defined to be a study of adversary's attack strength and attack confidence in manipulating classifier's decision boundaries for the targetted classes. Proactive defense techniques summarized across such attack settings include (i) randomizing training data and classifier output, (ii) domain experts correcting classifier decisions, (iii) data sanitization with robust statistics, (iv) automatic drift detection, (v) properly combining classifier ensembles, (vi) iterative adversarial training heuristics, (vii) game

theoretical adversarial learning, and (viii) robust optimization in regularized learning that effectively tackle the curse of dimensionality on large datasets and nonlinear classifiers. Future research of data-driven adversarial machine learning security evaluation is proposed to lie at intersection of software testing, formal verification, robust artificial intelligence, and interpretable machine learning.

2.2. Adversarial Deep Learning

Deng [68] surveys existing literature on deep learning for representation learning and feature learning where a hierarchy of higher-level features or concepts are defined from lower-level ones. Deep learning models, architectures and algorithms are categorised into three classes : generative, discriminative, and hybrid models:

- Generative models characterize the joint probability distributions of the observed data and their associated classes with high-order correlation properties between the observed variables and the hidden variables.

- Discriminative models distinguish between patterns by characterizing the posterior distributions of classes conditioned on the observed data.

- Hybrid models are discriminative models assisted by generative models in a significant way via better optimization or/and regularization of discriminative criteria used to learn parameters from data.

Deep learning is also understood by Deng [68] as an extension of previous research work on shallow architectures solving well constrained problems such as Generalized linear models, Multi-layer perceptrons, Support vector machines, Maximum entropy models, Conditional random fields, Gaussian mixture models and Hidden markov models. For the typical problems addressed by deep architectures, such shallow architectures and their statistical methods tend to produce intractable computational algorithms for class inference.

The commonly used deep learning models such as Deep Belief Networks, Variational Autoencoders and Convolutional Neural Network extract structures and regularities in the input features by avoiding difficulties with global optimization. Parameters optimization is done by designing a greedy layer-by-layer training algorithm that helps alleviate the overfitting problem observed in many shallow architectures training millions of

parameters. Thus deep learning models are useful for end-to-end learning of intelligent systems embedding domain knowledge and interpreting uncertainty.

2.2.1 Adversarial Examples in Deep Networks

Papernot *et al.* [183] present a practical demonstration of adversarial samples known to transfer between deep learning models. Such adversarial examples are constructed to control the integrity of a target Deep Neural Network (DNN) without access to the target DNN's architecture, parameters and training data. A substitute DNN is then trained to approximate the target DNN's learned model.The substitute DNN also has no knowledge of the probability vectors encoding the target DNN's belief of the relation between training input and classes. The attack is defined by assuming that the adversary can observe target DNN's outputs given the inputs chosen by the adversary. The adversary model has access to the same training data distribution as the target model.

The substitute DNN is trained through a Jacobian-based dataset augmentation technique. This step in the algorithm is called Substitute Model Training. This dataset augmentation technique allows adversary to select data points that are representative of the target DNN's behaviour in the input domain. The adversarial attack is made tractable by limiting the number of queries put to the target DNN. The queries are formulated by an efficient search heuristic over the input data domain.

After finding adversarial examples, the algorithm fine tunes the perturbations in the adversarial examples to maximize the transferability of adversarial samples. The fine tuning is based on the observation that the substitute DNN's and target DNN's cost gradient sign matrices are correlated. This step in the algorithm is called Adversarial Sample Crafting. In all, the adversarial algorithm goes through phases of Initial Collection, Architecture Selection, Labelling, Training, Augmentation. The adversarial algorithm produces a substitute training set that represents the target model's decision boundaries.

Two algorithms are implemented to search for the adversarial samples. Both search algorithms evaluate the target model's sensitivity to substitute model's input so that a small perturbation in the target model's input achieves the adversarial misclassification goal. Both search algorithms differ in computational efficiency of producing adversarial examples. Defenses proposed against such attacks include the use of adversarial sample training, Jacobian-based regularization and distillation, and a careful analysis of the distribution of queries.

Gu *et al.* [96] study robustness of DNNs by studying pre-processing and training strategies accounting for the structure of adversarial examples as well as the targeted model's network topology. The pre-processing is done with Denoising Autoencoders (DAEs). Autoencoder is chosen as the deep learning model because it preserves the original non-adversarial data distribution by mapping original training data back to itself. The experiments in Gu *et al.* [96] demonstrate that DAEs are able to remove adversarial noise in DNN training strategies. Furthermore, an end-to-end training procedure with a penalty function smoothing the adversarial data is proposed by stacking DAEs into a feed forward neural network called a Contractive Autoencoder (CAE). The additional penalty in CAE minimizes the squared norm of the Jacobian of the hidden representation of input data.

DNNs achieve high performance because deep cascades of nonlinear units allow to generalize non-locally in data-specific manifolds. The ability of DNNs to automatically learn non-local generalization priors from data is both a strength and weakness for adversarial learning in real world environments. Adversarial examples in DNNs are attributed to following reasons by Gu *et al.* [96].

- In high dimensional data, the smoothness assumption that underlies kernel methods does not hold for deterministic feedforward neural network architectures
- Applying kernel methods in manifold space does not guarantee local generalization in the input space
- Due to cross-model cross-dataset generalization properties of adversarial examples the attacker can generate adversarial examples from independent models
- Fewer degrees of freedom in data are captured as the layers of deep neural network increase

Therefore, according to Gu *et al.* [96] the challenge in DNN's design is to train a deep network that not only generalizes in abstract manifold space to achieve good recognition accuracy but also retains local generalization in the input space. In both shallow models and deep models, adversarial examples are also universal and unavoidable by this definition. Thus, deep learning architectures that are robust to adversarial data must be trained to incorporate input invariance with respect to the final network output accounting for adversarial data.

2.2.2 Adversarial Examples for Misleading Deep Classifiers

Although neural networks achieve high performance by expressing an arbitrary computation in terms of massively parallel nonlinear steps, Szegedy *et al.* [221] make observation that neural network layers do not disentangle basis distributions from semantic information. Szegedy *et al.* [221] find that deep networks learn discontinuous input-output mappings so that imperceptible perturbations increase deep network's prediction error even when it is trained on different subsets of a dataset. Such imperceptible perturbations are called adversarial examples. Learning adversarial examples is intrinsically connected to deep network structure and input data distributions. In experiments by Szegedy *et al.* [221], a significant amount of adversarial examples were found to be misclassified by deep networks. These adversarial examples were created by changing deep network's hyperparameter settings such as number of layers, weights initialization and weights regularization. Thus Szegedy *et al.* [221] conclude that adversarial examples are not the result of a particular deep learning model's overfitting.

In high-dimensional input signals to a simple linear model, Goodfellow *et al.* [91] observe that many infinitesimal changes to the input from adversarial examples add up to one large change to the output in deep learning. Goodfellow *et al.* [91] hypothesize that deep network classifiers exhibit such linear behaviour in high-dimensional spaces. Adversarial examples are then analyzed as a property of high-dimensional dot products. Stability of underlying model weights is said to result in stability of adversarial examples. To get adversarial perturbation, cost function for training a deep network is linearized around current parameter values. This method for generating adversarial examples is called Fast Gradient Sign Method (FGSM). In FGSM the direction of adversarial perturbation is hypothesized to be more important than its position in the data space. Then, adversarial training of deep learning models is proposed as a nonlinear regularization that secures deep networks by minimizing their worst case error on FGSM's adversarial examples. Adversarial training is also viewed as an active learning where learning model obtains new labels for the adversarial examples from a heuristic labeler copying labels of the nearby points. Papernot *et al.* [184] introduce a blackbox attack strategy to generate adversarial examples without knowledge of the target deep neural networks internals.

Nguyen *et al.* [178] generate adversarial examples with evolutionary algorithms and call them "fooling images". Fooling images are unrecognizable to human eyes but classified as recognizable objects with high confidence by Deep Neural Networks (DNNs). A population of fooling images is evolved by designing an evolutionary algorithm called multi-dimensional archive of phenotypic elites (MAP-Elites). MAP-Elites keeps best

individual found so far for each objective. Then it mutates a randomly chosen organism from the population and replaces the current champion for any objective if new individual has higher fitness on that objective. DNN's prediction score is taken as fitness function in MAP-Elites. For any class that has been seen before, a fooling image generated with higher prediction score becomes champion for that class. Image pixels of MNIST dataset and image pixels generated by compositional pattern-producing network (CPPN) represent the genomes in MAP-Elites. Various activation functions of a CPPN provide different geometric regularities to a fooling image. Evolution operators of MAP-Elites determine topology, weights, and activation units of each CPPN network in population. For various hypotheses on relations between training dataset and DNN architecture, prediction scores and Mann-Whitney U tests validate the fooling images distributions output by MAP-Elites.

Carlini *et al.* [49] devise whitebox as well as blackbox attack scenarios for feed-forward neural networks acting as classifiers. Across multiple detection mechanisms, new adversarial loss functions are proposed for fooling the neural network classifiers. Experiments are then proposed to explore the data space and transferability properties of targeted adversarial examples. To formulate the attacks, three threat models of Zero-Knowledge Adversary, Perfect-Knowledge Adversary and Limited-Knowledge Adversary are defined. A Zero-Knowledge Adversary has no knowledge of detector's presence while targetting class label predictions of the classifier. The Zero-Knowledge Adversary thus acts as a baseline for targeting any proposed detector. In comparison, a Perfect-Knowledge Adversary has full knowledge of both classifier parameters and detector's detection scheme. Perfect-Knowledge Adversary thus performs a whitebox attack. To perform a blackbox attack, Carlini *et al.* [49] assume that a Limited-Knowledge Adversary knows detector's detection scheme but has no access to trained classifier, trained detector or their training data.

The detector schemes studied by Carlini *et al.* [49] include (i) a second neural network to classify images as natural or adversarial, (ii) Principal Component Analysis (PCA) to detect statistical properties of images or network parameters, (iii) statistical hypothesis tests (such as Maximum Mean Discrepancy tests and Gaussian Mixture Models comparing adversarial data distribution with original data distribution) and (iv) input-normalization with randomization and blurring. A ℓ_2 distance between adversarial examples and training examples is assumed to be the adversarial loss function measuring robustness of defense in the detection mechanisms. In an iterative gradient-descent attack, the adversarial loss function has an additional regularization term that compares

the deep network's log-likelihood of predicting target class with the next-most-likely class. A user-defined threshold on ranked log-likelihood's then assigns either a high-confidence or low-confidence to the generated adversarial examples. The evaluation of the properties of the adversarial examples is recommended to be done according to following evaluation criteria:

- Evaluating adversarial examples across multiple datasets (such as MNIST and CIFAR) with defenses that did not operate directly on pixels
- Evaluating new schemes for strength of an attack which demonstrate an adversary who can generate attacks to evade detection when aware of proposed defense
- Reporting false positive rates in addition to true positive rates in the performance evaluation.

Here a neural network is said to be robust if finding adversarial examples that bypass its detector is a difficult proposition.

Baluja *et al.* [17] proposed a targeted attack where feed-forward neural networks called Adversarial Transformation Networks (ATNs) are trained to generate adversarial examples. ATNs generate adversarial examples that minimally modify classifier's outputs given original input. By contrast, Moosav *et al.* [173] constructed an untargeted attack technique, i.e., DeepFool, which is optimized by distance metrics between adversarial examples and normal examples.

In our research, we generate adversarial examples to effect a poisoning attack on the classification training data. The adversarial examples are generated by the adversarial manipulations learnt during our game theoretical attacks on the training process of the learner. In our blackbox attack scenarios generating testing data distributions, no prior knowledge about the learning model is assumed. Our adversary knows neither the learning model's training process nor the learning model's best response strategies across the Stackelberg game's plays. Our adversary does a targeted attack to manipulate multiple positive labels into a single negative label. The attack strength of our adversarial manipulations is defined in terms of search randomization parameters in ALS and SA. The scalar optima in SA are used to generate the vector optima in ALS. The local optima in ALS converge onto the non-convex stochastic optima solving the Stackelberg game to produce output of optimal adversarial manipulations. The optimal adversarial manipulations are able to encode the adversarial data in terms of the multivariate statistical parameters of a Gaussian mixture model produced in multi-label datasets.

2.2.3 Generative Adversarial Networks

Goodfellow *et al.* [89] state that the primary cause of deep learning networks vulnerability to adversarial examples is their linear nature in high dimensional search spaces. Also the deep learning networks perform poorly on testing data examples that do not have high probability in the training data distribution. Thus adversarial examples can be generated by applying a worst-case perturbation to the training data. The perturbed input results in an incorrect output prediction with high confidence. Thus Goodfellow *et al.* [89] argue for the need of having an adversarial training procedure whose objective is to minimize the worst case error when the training data is perturbed by the adversary. Goodfellow *et al.* [89] then formulate the adversarial training as a minmax game between two deep neural networks. The resulting deep generative model is called Generative Adversarial Networks (GANs).

A variety of deep generative methods are available to create the perturbation between training and testing data distributions [186]. Radford *et al.* [194] propose a stable GAN called DCGAN. Gulrajani *et al.* [99] design IWGAN which undertakes a theoretical analysis of the generative learning process. Berthelot *et al.* [26] propose BEGAN with a new loss function in the training algorithm. Chen *et al.* [54] propose InfoGAN which uses generative learning models for unsupervised representation learning.

Insofar as the learner's defence mechanisms are concerned, our game formulation is similar to the GAN game formulation. However, the objective of our research is to simulate a real adversarial attack scenario on two-label and multi-label classification model in terms of the cost to the adversary. We seek to increase the classification performance when the data distribution is changed with a malicious intent. By contrast, the objective of GAN is to generate synthetic data that is indistinguishable from the original data. Our objective function has cost and error terms defining the attack scenarios in adversarial data generation settings. By contrast, the objective function in GAN is defined in terms of the loss functions of the deep neural networks learning the given training and testing data distributions.

In a minmax game formulation, we seek to create the datasets for attack scenarios in a discriminative learning model and supervised learning problem while GAN addresses a generative learning model and unsupervised learning problem. Furthermore, the generator is the leader of the game in minmax formulation for GAN, whereas in our minmax formulation the intelligent adversary leads the game. While searching for the Nash equilibrium in a minmax game, GANs solve a convex optimization problem with gradient-based optimization algorithms whereas we solve a non-convex stochastic opti-

mization problem with evolutionary learning algorithms. Thus, we are able to estimate the best cost for the adversary in effecting the adversarial attack.

Generative Adversarial Networks (GANs) [89] estimate data likelihood with a adversarial framework involving a two-player game between a generator network G and a discriminator network D. IWGAN [99] improves GAN's estimations with regularization that does not introduce correlations between generated examples.

The objective of our game formulation with variational autoencoders is not to improve classification accuracy by augmenting the original data training the autoencoders. Importantly, we note that the fundamental difference between our research with variational autoencoders and generative networks objective is deceiving the classifier rather than mimicking the original data [99, 194]. We solve a supervised learning problem with variational adversaries while deep generative models generally solve either unsupervised learning or semi-supervised learning problems with generative adversaries.

2.2.4 Generative Adversarial Networks for Adversarial Learning

Adversarial examples have been defined for deep generative models [131]. The distribution of adversarial manipulations in whitebox attacks as well as blackbox attacks have been modelled with AdvGAN [238]. A thread of research on adversarial autoencoders [226] imposes a prior distribution on the output of an encoder network learning training data, where autoencoder discriminatively predicts whether a sample comes from its latent space or from prior distribution determined by the user. By contrast, our game theoretical optimization problem is independent from a particular training data distribution and classification model.

Larsen *et al.* [138] propose generative adversarial learning in the reconstruction loss of a variational autoencoder. Tran *et al.* [225] propose constraints on the distance function to train generative adversarial networks in the latent spaces of an autoencoder. Gregor *et al.* [94] propose an attention mechanism based autoencoder to learn the latent spaces in a sequential variational autoencoder framework. Ha *et al.* [103] propose a recurrent neural network for sketch generation in images. Makhzani *et al.* [158] propose an adversarial training mechanism for probabilistic autoencoders.

A taxonomy of adversarial attack scenarios in deep learning is provided by Gilmer *et al.* [85] and Biggio *et al.* [39]. Our attack scenario with Stackelberg games proposes new adversarial payoff functions. We represent feature space for adversarial manipulations

in terms of adversarial cost functions, stochastic operators and game strategies in a simulated annealing algorithm.

Wang *et al.* [232] survey theories and implementations of generative adversarial networks. A taxonomy of the existing generative adversarial networks formulations relevant for the game theoretical adversarial learning algorithms is summarized in Table 2.3, Table 2.4, Table 2.5 and Table 2.6. Across the rows in tables, the algorithms comparisons are made on Generator network's attack scenario, loss function, strategy space and objective function. Most of the deep generative models do not analyze data distributions in terms of game theoretical optimization of the objective functions. In comparison, our methods propose adversarial payoff functions for optimization and adversarial cost functions for regularization in the objective functions.

Adversarial network	Attack scenario	Discriminator loss function	Generator loss function	Information divergence function	Game type	Payoff function	Cost function	Optimization constraints
Defense-GAN [206]	model the distribution of unperturbed images in whitebox attacks as well as blackbox attacks	same as WGAN	same as WGAN	MSE, L_2 norm	minmax game	reformer network, latent codes	adversarial training augments the training data	representative GAN to reconstruct adversarial examples
GANGs [182]	resource-bounded best responses on synthetic data	classifier score that is function of both real data and fake data	generator payoff as a function of the fake data only	deterministic and nondeterministic Resource-Bounded Nash Equilibrium	zerosum strategic-form game	player payoff under a profile of mixed strategies, Definition 2	Measuring function in Definition 6 and Theorem 10	finite GANGs on discrete data
AdvGAN [238]	semi-whitebox adversarial perturbations with target class and ground truth	static and dynamic distillation model training with alternative minimization approaches	targeted attack loss for LSGAN given in Equation 4	ensemble adversarial learning	minmax game	same as Goodfellow GAN	hinge loss	cross entropy loss
DeLiGAN [102]	limited training data to capture the diversity across the image modality	same as DCGAN	same as DCGAN	m-IS a KL divergence measuring intra-class sample diversity along with the sample quality	minmax game	reparameterization of the latent space in prior distribution	L_2 regularizer to prevent local maxima in generator	uniform mixture weights in gradient descent
EBGAN [251]	handcrafted and regularized contrastive samples generation in supervised, weakly supervised and unsupervised settings	reconstruction loss and energy function in auto encoder	same as goodfellow gan	inception score	minmax game	adversarial training	reconstruction error	grid search over architectural choices and hyperparameters
Fisher GAN [174]	standardized discrepancies in two-samples hypothesis testing and semi-supervised learning	Mahalanobis distance between the feature means embeddings of real and fake distributions	mean embedding distance	Integral Probability Metrics (IPM) and inception score parametrized by generative neural networks	minmax game	same as DCGAN	same as DCGAN	second order moments of the critic that discriminates between the two distributions, crossentropy regularization term in semi-supervised learning
Improved f-GAN [192]	generator divergence ordered from the most mode seeking to most mode covering	same as Goodfellow gan	estimate of the model density to data density density ratio from the current discriminator	f-divergence	minmax game	expectation maximization	real-fake data learning intractable likelihoods	image quality/diversity metrics/factors without dropping modes

Table 2.3: Generative Adversarial Networks Comparison 1

Adversarial network	Attack scenario	Discriminator loss function	Generator loss function	Information divergence function	Game type	Payoff function	Cost function	Optimization constraints
CausalGAN [128]	true/feasible causal graph structured on data labels for both the observational and interventional distributions over images and labels jointly	conditional GAN losses mixed with Label losses	same as DCGAN with Label gradients	total variation distance (TVD)	same as WGAN, DCGAN	same as WGAN, DCGAN	two-stage procedure on multiple label and binary label conditioned image distributions	margin-coefficient tuples in stochastic gradient descent
AM-GAN [255]	predefined labeling and dynamic labeling for image generation, image quality and image diversity	cross-entropy for multi-class classification	same as DCGAN	Inception Score, AM Score	minmax game	none	none	class-aware gradient
MBGAN [61]	none	nonlinear metric in Equation 9 and Equation 10	same as DCGAN	inception score	minmax game	none	none	weight clipping, center penalty
Triangle-GAN [84]	semi-supervised image classification, image-to-image translation and attribute-based image generation	conditional GAN and bidirectional GAN losses	two conditional GAN generators corresponding to two discriminators	Jensen-Shannon divergence (JSD) plus a Kullback-Leibler (KL) divergence	minmax game	none	none	none
f-CLSWGAN [237]	no labeled examples of certain classes in multimodal embedding	softmax classifier for zero-shot learning (ZSL) and generalized zero-shot learning (GZSL)	same as WGAN	multimodal embedding model with labeled examples of seen classes and deep CNN features conditioned on class-level semantic information	minmax game	none	generated data is much lower dimensional than high quality images necessary for discrimination	class embeddings that models the semantic relationship between classes
LSGAN [162]	penalize samples based on their distances to the decision boundary	Equation 2	Equation 2 to generate samples toward the decision boundary and manifold of real data	f-divergence, Pearson Chi-square divergence	minmax game	none	none	deterministic equations between labels for fake data, real data and generated data for one-hot encoding and dimensionality reduction
D2GAN [179]	image generation	Kullback-Leibler (KL) and reverse KL divergences	any multi-modes density function	Inception score and MODE score	minmax game	none	none	minimal enclosing ball, surrogate objectives, multiple players etc in generator's density function

Table 2.4: Generative Adversarial Networks Comparison 2

Adversarial network	Attack scenario	Discriminator loss function	Generator loss function	Information divergence function	Game type	Payoff function	Cost function	Optimization constraints
GAN [90]	adversarial framework for likelihood estimation which backpropagates discriminator derivatives through generative processes	Maxout activations networks	rectifier linear activations and sigmoid activations networks	Kullback-Leibler divergence, Jensen-Shannon divergence	minmax game	none	none	G and D are given enough capacity and training time, No overfitting in D, G must not be trained too much without updating D
IWGAN [99]	better gan training algorithm	critic loss in WGAN	adversary loss in WGAN	Earth Mover Distance	minmax game	none	none	penalty on the gradient norm, normalization schemes which don't introduce correlations between examples
InfoGAN [54]	disentangle interpretable representations from unlabelled data	same as DCGAN	information-regularized generator on incompressible noise and structured semantic features	mutual information and differential entropy	minmax game	none	none	sleep wake algorithm with variational regularization
ss-InfoGAN [217]	extracted and controllable data representations where latent variables correspond to label categories, learn both continuous and categorical codes	same as DCGAN	mutual information between a code vector and real labeled samples and synthetic unlabeled samples, cross-entropy for categorical latent codes, mean squared error for continuous latent codes	mutual information and differential entropy	minmax game	none	none	factors not corresponding to labels will not be discovered in supervised and semi-supervised settings, real and synthetic data distributions are independent, labels follow a fixed distribution and hence have a fixed entropy

Table 2.5: Generative Adversarial Networks Comparison 3

Adversarial network	Attack scenario	Discriminator loss function	Generator loss function	Information divergence function	Game type	Payoff function	Cost function	Optimization constraints
McGan [175]	mean and covariance feature statistics	same as WGAN along with a cross-entropy loss on labelled data	mean L_q norm, covariance Ky-Fan norm (nuclear norm of truncated covariance difference), feature matching Integral Probability Metrics (IPM), IPMs are bounded linear functions defined in the non linear feature space induced by the parametric feature map	geodesic distances between the covariances and probability measures in a multimodal setting	minmax game	none	none	bounded modes of the feature embeddings of real and fake distribution, sufficient samples from "real" and "fake" data for training both "generator" and the "critic" feature space
DCGAN [194]	reusable feature representations from large unlabeled datasets, hierarchical clustering of the intermediate representations	leaky rectified activation	deconvolutions and filtering the maximal activations of each convolution filter in the network	percentage accuracy on training, testing, validation data	minmax game	none	none	batch normalization
BEGAN [26]	matching the distribution of the errors instead of the distribution of the samples, dynamically weighing regularization terms or other heterogeneous objectives	discriminator has two competing goals in closed-loop feedback control: auto-encode real images and discriminate real from generated images	negative of discriminator loss	global measure of convergence by using the boundary equilibrium concept from Proportional Control Theory	minmax game	none	none	correct hyper-parameter selection to maintain a balance between the generator and discriminator losses
BGAN [107]	same as DCGAN	same as DCGAN	boundary-seeking REINFORCE objective with policy gradient training where reward is the normalized importance weights	f-divergence and Jensen-Shannon divergence with importance weights for generated samples	minmax game	none	none	approximated the expectations in normalized importance weights by using the Monte-Carlo sampling
AnoGAN [208]	imaging markers relevant for disease progression and treatment monitoring	discrimination loss enforces the generated image to lie on the learned manifold	residual loss that enforces visual similarity between generated image and query image	anomaly/novelty detection score from feature matching classification functions in latent spaces	same as DCGAN	none	none	suitable feature representation in discriminator

Table 2.6: Generative Adversarial Networks Comparison 4

2.2.5 Causal Inference in Deep Learning

Causality methods have been applied to deep learning problems such as semi-supervised learning and transfer learning. In these problems informed priors retrieved from other networks are used to center the weights in hybrid deep learning networks. Such networks are then used to construct statistical hypotheses on patterns, structure, context and content in actual data [171].

Backpropagation learning algorithms for deep networks have been improved by training probabilistic graphical models. Such training is inherently Bayesian where prior distributions inform and constrain analytics models predicting posterior distributions [216]. The improved deep learning algorithms result in a predicted output informed by causal inference. Within a Bayesian framework, causality methods also enhance the interpretability of deep networks operating in an uncertain environment [122].

To compare predicted output with actual data in structured prediction, deep learning models act as computational methods reasoning under uncertain environments. Here, causality methods define various probability, causality and utility functions reasoning about the uncertain environment [125]. Furthermore, the environment may be partially stochastic and partially observable. Then, a deep network's learning objective is to not only minimize the validation error but also to best handle the model uncertainty in the given learning environment. This is our general regime for training, testing and validating deep learning against causal inference.

Granger causality has been used to study causal structures and non-linear regression models in structural time series analysis [15], dependency modelling [56] and structured prediction [139]. The instance space for time varying data generated in uncertain environments consists of concept adapting data structures like strings, trees, networks and tensors. Causal features derived on such instances are to be evaluated on both prediction performance and model selection in deep learning. Formal causal analysis with features interpretation or human intervention or both can also help design the learner's environment.

2.2.6 Causal Inference in Time Series Analysis

Historically, causal reasoning in time series builds on statistical analysis of covariance or correlation between two or more events in time series. The calculated correlation strength is then used to predict causal relation between two events [82]. The disadvantage of this approach is that it cannot determine the direction and significance of causation. It also

cannot discover hidden causes and patterns for which observed events are effects.

In time series analysis, causal inference identifies and classifies events in time series with either deterministic or probabilistic causal relations. Events are identified by mapping logic and structure of natural language to concept lattices and causal graphs [121].

Causal features between two events in a time series can be derived from one or more of statistical relations, human intervention, prior knowledge and temporal ordering in the time series. Statistical relations correspond to correlation information and conditional dependencies on data events used to arrive at associative Causal Markov models on hidden or latent variables [8]. When the causal features are implicit rather than explicit, human intervention or human choice allows the separation of the cause of prediction from the mechanism of prediction. Human intervention also affects the events participating in the causal inference. Typically, an experiment is conducted to find the effect of human intervention on the prediction output. The concepts of temporal ordering and prior knowledge allow human intervention mechanisms that are better than trial-and-error learning. In temporal ordering, events occurring prior to an event are said to be the cause of that event, but not vice versa. Causal features then identify significant causes for events that are desirable as distinguished from spurious causes for events that are undesirable. Such causal features construct causal chains in time series. Prior knowledge on the construction of causal chains in an application domain is also useful as a human intervention mechanism for hypothesis learning.

Granger causality is a simple learning mechanism that allows us to explore all preceding ideas about causality methods in deep learning for time series analysis [16]. Here, Granger causality does not empirically prove actual causation between events but acts as a stepping stone to explore the phenomenon relating two events participating in a cause-effect relationship. Granger-causal features have been discovered with rule-based analytics models [126] and feature-based analytics models [146]. Our approach to causal inference also builds a feature-based analytics model.

2.2.7 Causal Inference in Financial Markets

Due to computational simplicity, Granger causality is a popular method for causal inference in econometrics [98]. Granger causality tests are performed by fitting a vector autoregressive model assuming linearity and stationarity in the causal structure of a multivariate time series [72]. Furthermore, Granger causality has been enhanced with a Granger-causal Markov property for non-linear regression to derive causal structure over

temporal causal graphs [11]. In time series analysis, information theory based models in causal networks have also been used to assess Granger causality graphs of stochastic processes [243].

In cross-sectional econometrics, factor analysis and causal inference is used for estimating the impact of counterfactual policies. The modelling focus is on estimating what would happen in the event of a change that may or may not actually happen. Systematic model selection in machine learning thus adds empirical value to econometrics studies [58]. Such machine learning models may sacrifice predictive performance in the current environment to discover new causal features in a changing environment. The counterfactual policies of econometrics can be used to define new sensitivity analysis, anomaly detection and concept drift algorithms in temporal data mining.

2.2.8 Causal Feature Learning and Adversarial Machine Learning

We are interested in the attack scenarios with latent variable models in game theoretical adversarial learning. Kumari *et al.* [134] study whitebox attacks at the level of the latent layers of the adversarially trained image classification models. Higher robustness at the feature layers is achieved by the adversarial training of latent layers with an iterative variant of FGSM. By contrast, our research creates deep generative models for the adversarial manipulations that provide game theoretical regularizers on the targeted classifier's loss function.

Chattopadhyay *et al.* [52] propose a structural causal model for causal influence of an input feature on a neural network's output. Such causal influences on the prediction function's output are called neural network attributions. They are said to be more interpretable artifacts of the deep network causations rather than regression features that primarily map correlations between the input and the output of the neural network. In sequence prediction tasks with such a structural causal model, the causal dependencies between different input neurons are assumed to be jointly caused by a latent confounder such as a data-generating mechanism applied to time-series models.

Yang *et al.* [244] study the pixel-level features for causal reasoning in pixel-wise masking and adversarial perturbation. Ancona *et al.* [9], Lundberg *et al.* [156] discuss attribution methods in Shapley values from cooperative game theory.

Our research investigation is in creating such interpretable artifacts of the game theoretical adversarial manipulations. Towards this end we have created Granger-causal fea-

tures of the regression predictions. In future work, we shall create predictive baselines in latent variable models of the data-generating mechanisms in neural network attributions. We expect such baselines shall discover counterfactual features in application-specific rule based classifiers.

2.2.9 Explainable Artificial Intelligence and Adversarial Machine Learning

We are interested in Explainable Artificial Intelligence (XAI) of the deep generative models applicable to game theoretical adversarial learning in blackbox attacks. Lou *et al.* [154] introduced Generalized Additive Models (GAMs) as an interpretable extension of Generalized Linear Models (GLMs). Guidotti *et al.* [97] survey explainability of black box models. Rudin [205] compares XAI models with inherently interpretable models. Wang *et al.* [233] propose Hybrid Rule Sets that integrate interpretable models with black-box models. Frosst *et al.* [81] create a decision tree that generalizes a neural networks learning. Ribeiro *et al.* [197] provide textual Anchor Explanations for Image Classification and Visual Question Answering. Ignatiev *et al.* [112] propose a constraint reasoning system to explain predictions.

Strumbelj *et al.* [218] explain predictions with coalitional game theory. Bulo *et al.* [202] define a randomized prediction game that is a non-cooperative game-theoretic formulation in which the classifier and the attacker make randomized strategy selections according to some probability distribution defined over the respective strategy set in handwritten digit recognition, spam detection and malware detection. Peake *et al.* [187] create interpretable structure of association rules from latent factor recommendation system training a matrix factorisation black-box model. Lakkaraju *et al.* [137] create rule-based models with decision set learning designed for interpretability of submodular function optimization. Baehrens *et al.* [14] propose explanation methods for the decisions of any classification method. Ribeiro *et al.* [196] explain predictions of any classifier as a submodular optimization problem. Shrikumar *et al.* [211] compute importance scores for neurons activation in a neural network that show significant advantages over gradient-based methods. Koh *et al.* [129] use influence functions from robust statistics to explain predictions.

Bastani *et al.* [24] propose metrics to evaluate the robustness of deep neural nets. Narodytska *et al.* [176] create Boolean representation of a deep neural network to verify its properties. Tomsett *et al.* [223] survey connections between interpretability and

adversarial attacks. Liu *et al.* [149] develop adversary-resistant detection framework by utilizing the interpretation of machine learning models. Tao *et al.* [222] propose an adversarial sample detection technique for face recognition models, based on interpretability. Fidel *et al.* [77] propose a method for detecting adversarial examples with Shapley Additive Explanations (SHAP) values computed for the internal layers of a DNN. Ilyas *et al.* [115] attribute adversarial examples to the presence of non-robust features. Ignatiev *et al.* [113] demonstrate that the explanations (XP's) of machine learning (ML) model predictions and of adversarial examples (AE's) are related by a first order logic (FOL) framework called hitting set duality.

2.3. Spam Filtering

2.3.1 Biometric Spam

Biometrics is an area of research where security is a major issue. Security in biometrics is determined by the vulnerability of pattern classification methods. Biggio *et al.* [35] investigate attacks and defences for adversarial learning in adaptive biometrics systems. The attacks in an adaptive biometrics system pertain to either the recognition of biometrics or the changes of biometric traits over time. To effectively deal with these attacks the stored biometric templates ought to match the claimed identities submitted during verification.

The attack points found during the process of matching biometrics identity are categorized by Biggio *et al.* [35] as sensor input, feature extraction, template database, matching algorithm, template update, scoring rule and scoring thresholds. In adaptive biometrics additional attack points include template theft and malware infection disregarding intrinsic failures. These attacks are further classified into attacks on sensors, interfaces and channels connecting modules, processing modules and algorithms, template databases.

Following attacks are seen in adaptive biometric systems

- **Spoofing** attacks fabricate a fake biometric trait to impersonate an enrolled client.
- **Replay** attacks stage stolen biometrics as features in the matching algorithm.
- **Hill-climbing** attacks affect the communication channels by iteratively sending perturbed data to the matching algorithm and retaining data that gives maxi-

mum matching score. The iterations in attack continue until convergence of the optimization method used by the adversary.

- **Malware infection** attacks exploit well known software and hardware vulnerabilities through hacking techniques and programming practices.
- **Template Theft** attacks target improperly protected template databases that are not encrypted.

Biggio *et al.* [35] then go onto characterize attacks in biometric systems according to the security framework discussed by Vidyadhari *et al.* [230]. A spoofing attack scenario, a poisoning attack scenario and an evasion attack scenario are discussed as a motivation for adversarial learning algorithms in secure-by-design biometric systems. Pattern matching algorithms in secure-by-design biometric systems are recommended to be be designed ground up according to the considerations in statistical databases. Such considerations include learning with invariances, error tolerance in PAC learning and online learning with game theory.

2.3.2 Image Spam

The image spam detection problem is a part of content-based filtering of multimedia data in adversarial environments. Such multimedia data is often produced on the Internet Communities and Mobile networks. According to the survey by Attar *et al.* [12], image spam is created by embedding spam text message into images. The adversarial objective is to prevent text recognition by optical character recognition software. Keyword detection, text categorisation, image classification, near duplicate detection are the existing techniques for image spam detection. In applying these techniques to adversarial learning, the underlying assumption is that the text in legitimate images (and corresponding features discriminating between spam images and legitimate images) is unlikely to be obfuscated with adversarial features.

The adversarial features can also be built on the rationale of content-based image retrieval where search for either spam images or legitimate images is driven by a set of low-level features found in query images. In such methods, the distance between a query image and templates in database is computed for each feature space and compared to a threshold to decide whether an image is a spam image or a legitimate image. Thus, the generalisation capability of the adversarial learning algorithms over image spam strongly depends on choice of proper features for adversarial data manipulation. In the

existing literature the choice of features depends on assumptions about properties which best discriminate between spam images and legitimate images. The most commonly used features include text obfuscation, text area, low-level image properties (like colour, texture etc), image similarity, image regions similarity, image metadata. The relevant features are then selected based on results concerning classification accuracy, true positive rate, false positive rate, precision, recall. The most common classifiers include support vector machines, decision trees, maximum entropy models, bayesian networks.

2.3.3 Text Spam

Email spam filtering has been analyzed as a lazy learning problem in concept drifts [65]. Kazemian *et al.* [120] compare machine learning techniques to detect malicious web pages. Adversarial examples in reading comprehension systems are analyzed by Jia *et al.* [116]. Chen *et al.* [53] discuss adversarial examples in machine-learning classifiers for malware detection. Miyato *et al.* [172] discuss adversarial training with adversarial examples on word embeddings in a recurrent neural network. Dasgupta *et al.* [64] adversarial attack scenarios in text classification for sentiment analysis on social media sites. Cheng *et al.* [55] craft adversarial examples for sequence-to-sequence (seq2seq) models.

Our method is not specific to any particular datasource. We experiment with image databases, text databases and time series databases.

2.4. Security and Privacy in Adversarial Learning

Evasion attacks Biggio *et al.* [29] discuss adversarial security at test time of a deployed classifier system. Security evaluation is then performed at different risk levels of non-linear classifier performance in malware detection. A secure classifier is proposed by using a gradient-descent approach on a differentiable discriminant function. Adversary's goal is defined in terms of minimizing classifier's loss function with positive adversarial samples that cross decision boundary. This model can also incorporate application-specific adversarial knowledge in the definition of adversarial attack scenarios. Such adversarial knowledge includes prior knowledge about training data, feature representation, type of learning algorithm and its decision function, classification weights and feedback from classifier.

Poisoning attacks In security-sensitive settings, machine learning algorithms cannot assume training data is from a natural and well-behaved distribution. By injecting adversarial examples into training data such that testing error increases, Biggio *et al.* [37] investigate poisoning attacks against Support Vector Machines with linear kernel, polynomial kernel and RBF kernel. A gradient ascent procedure is used to compute adversarial examples as local maxima of SVM's non-convex error surface. In gradient ascent iteration, after each update to attack example optimal decision boundary is computed from solution to an incremental SVM. Search procedure takes many tiny gradient steps. It is stopped when attack example deviates too much from training data. The changes to SVM's decision boundaries due to malicious input are shown to be important in application domains such as spam, worm, intrusion and fraud detection.

Xiao *et al.* [241] propose adversarial label noise to maximize SVM's worst-case classification error by flipping labels in the training data. Attack strategies for creating the adversarial label noise are motivated by a structural risk minimization framework. In this framework, SVM learning minimizes a sum of a regularizer risk and empirical risk in data. Here a regularizer penalizes excessive hypothesis complexity to avoid overfitting in a convex optimization quadratic programming problem. The adversary then optimizes empirical risk on malicious data so that the SVM is misled into shifting decision boundary away from the original data distribution. Empirical risk optimization is further decomposed into two iterative sub-problems solved by quadratic programming and linear programming. Further labels of samples in different classes are flipped in a correlated way to force the hyperplane forming decision boundary to rotate as much as possible. Adversary is assumed to have full knowledge of the feature set in the training data with equal cost assigned to each label flip.

Inference Attacks Shokri *et al.* [210] investigate privacy breach problem of commercial classification models leaking information about their training data on internet applications. Adversary queries target model as a blackbox to retrieve model's output on a given input. Such inputs are generated by training shadow models that imitate behaviour of the target model. In contrast to the blackbox target model, shadow models know ground truth label for the inferred record. The blackbox models are neural network models in Amazon ML and Google Prediction API trained on datasets of images. The details of blackbox models are hidden from their data owners. The datasets are obtained from retail purchases, location traces and hospital inpatient stays. Here, a privacy breach is said to occur if the adversary can use model's output to infer the values of sensitive attributes in the model input. The attribute inference is defined by Shokri *et al.* [210] in

terms of class membership inference of given data record's presence in model's training dataset. The success of the proposed class membership inference is measured in terms of attack precision and attack recall of target model. The shadow model is trained on a synthetic dataset with a hill-climbing algorithm generating candidate records which are classified with high confidence by the target model. In each iteration of hill-climbing a candidate record is proposed by changing randomly selected features of latest accepted record. A candidate record is accepted in hill-climbing algorithm only if it increases probability of being correctly classified by target model. Several defense strategies are proposed against class membership queries. These strategies include restricting prediction vector to top classes, rounding classification probabilities, increasing entropy of prediction vector such that output becomes almost uniform and independent of input, and regularizing classification loss function to penalize large parameters during training.

2.4.1 Adversarial Attack Scenarios in Linear Classifiers

Dalvi *et al.* [62] analyze classifier performance by viewing classification as a game between the classifier adapting to an adversary seeking to make the classifier produce false negatives. Here the cost-sensitive adversary is contrasted with the cost-sensitive classifier where data-generating process in adversarial classification is not only allowed to change over time but also allow this change to be a function of classifier parameters. Adversarial classification is thus defined in terms of a game between two players – the adversary and the classifier – where classifier maximizes its payoff function characterized by classifier's expected payoff over adversary's cost parameters.

Dalvi *et al.* [62] propose that the adversary's goal is to find a classification feature change strategy that maximizes adversary's expected payoff. Adversarial examples are generated by standard feature selection algorithms with naive Bayes classifier's payoff function as evaluation function. The theory of computationally tractable Nash equilibria strategies in adversarial classification is left as an open question analyzing the two-player nonzero-sum games.

Lowd *et al.* [155] introduced adversarial learning algorithms for linear classifiers under attack. The goal of adversarial learning is to learn and attack part of classifier's decision boundary by learning feature weights without (i) constructing domain specific feature representations, and (ii) assuming a stochastic process for training data distribution.

Lowd *et al.* [155] assume that adversary can send membership queries to classifier to distinguish between malicious examples and non-malicious examples. The computational

complexity of possible membership queries is bound by a polynomial number of line searches along each feature dimension. Adversarial learning algorithm then minimizes a linear adversarial cost function over non-malicious instances space to be learnt by the adversary. Optimal adversarial cost function produces non-malicious examples that are most similar to a base adversarial example accessible to the adversary.

Lowd *et al.* [155] also demonstrate adversarial training experiments on linear classifiers like naive Bayes models, support vector machines with linear kernels and maximum entropy models learning boolean features for spam filtering. The proposed learning framework (called ACRE) is useful to study both the attacker or adversary and the defender or classifier. It can be used to determine whether an adversary can efficiently learn enough about defeating a classifier by minimizing an adversarial cost function.

2.4.2 Adversarial Attack Scenarios in Feature Weighting

Traditionally, machine learning algorithms assume that algorithm training can be performed on controlled and high quality data. In the real world, machine learning is performed on noisy and uncertain data. Here, a robust classifier can anticipate noisy features during testing only when it is trained assuming noisy features are present during training. Moreover, robust classifiers must be dense classifiers that train on as many informative or important features as possible. Such considerations are the focus of feature weighting techniques in adversarial environments. Here adversarial data created by an intelligent adversary is different from the random noise found in the natural world.

Since traditional classifiers cannot continuously adjust to changes in adversarial environments, Kolcz *et al.* [130] attempt to design classifiers that degrade gracefully as the distribution of testing data diverges from the distribution of the original training data. This is done by a feature selection process reweighting less important features for classification. The feature weighting improves model performance by making it robust to concept drift in data at the expense of extra computational cost in the model. The intuition behind this approach is that weights distribution over features for the learning algorithm reflects the importance of the features for unsupervised and supervised learning.

Kolcz *et al.* [130] envision a two stage approach to robust classifier training where classifier is used to assign weights to features in the first stage which are then transformed through feature weighting to induce the final model in the second stage. The final model satisfies an objective function to be optimized. Since a single best reweighting scheme for both supervised and unsupervised learning is not available in the literature, Kolcz *et al.* [130] experiment with several choices for feature weighting. The objective

function for feature weighting is formally analyzed by Kolcz *et al.* [130] as a particular case of regularized risk minimization with a quadratic form regularizer and convex loss function.

The feature weighting methods in experiments by Kolcz *et al.* [130] include feature bagging, partitioned logistic regression, confidence-weighted learning, feature noise injection and sample selection bias correction. These details are described below:

- Feature bagging trains a probabilistic model as arithmetic or geometric mean of several base models. Each base model is trained on possibly overlapping subset of the original features. The performance of the bagged model is supposed to more robust than the performance of any base model because weights of less important features will be overwhelmed by the weights of more important features during the training process.
- Partitioned logistic regression is a special case of feature bagging where feature subsets and class labels are non-overlapping.
- To prevent undertraining, Confidence-weighted learning aggressively updates weights of rare features in the data by maintaining a normal distribution over the weight vector of a linear classifiers. The feature weights are updated such that the Kullback-Leibler divergence between the training data distribution and testing data distribution is minimized without reducing model performance.
- Feature noise injection alleviates the problem of model overfitting to training data by introducing artificial feature noise during model training.
- Sample selection bias correction assigns feature weights such that reweighted training data resembles the available testing data. The correct weights are inferred without explicit density estimation. However sample selection bias correction assumes the testing data is also available during training in the input domain.

Liu *et al.* [148] design supervised learning algorithms secure to adversarial poisoning attacks that do not make independence assumptions on feature distributions. Poisoning attacks are assumed on both dimensionality reduction and predictive regression steps. High-dimensional features are projected into a low-dimensional subspace with high data density. Then linear regression models best characteristics of data. A matrix factorization algorithm is proposed to recover low-dimensional subspace in presence of training data corrupted by both noise and adversarial examples. A principle component regression uses trimmed optimization to estimate regression parameters in low-dimensional subspace.

In the adversarial attack scenario proposed by Liu *et al.* [148], the regression model can choose its training process and defense strategy without access to training data before adversarial manipulation. The adversary has full knowledge of training algorithm and parameters. The adversarial attack scenario is simulated as a zero-sum Stackelberg game where adversary's payoff function minimizes a certain budget of poisoning training data while regressor's payoff function is regression accuracy. The learning process is formally characterized in terms of a model function relating the adversarial input and the predicted output. A quadratic loss function and a threshold function bounding loss function are also analyzed in the regression. An alternative maximization solves proposed optimization problem on HTTP logs. The adversarial data is generated by moving training data samples along a direction to manipulate regressor until it cannot predict correctly. Results are benchmarked against robust regression models like OLS linear regression and ridge regression predictions in presence of noise.

2.4.3 Poisoning Support Vector Machines

According to the previously discussed adversarial security mechanisms, poisoning attacks are causative attacks where specially crafted attack points are injected into the training data. In a poisoning attack, an adversary cannot access the training database but can provide new training data. Poisoning attacks compromise the security of a large-scale learning system that infers hidden patterns in large complicated datasets to support decision making with behavioural statistics. Previous poisoning attacks have been studied with anomaly detection methods.

In the Probably Approximately Correct (PAC) model, the structural risk minimization of SVM learning is studied in the context of a convex quadratic programming problem. The impact of stochastic and adversarial label noise on Support Vector Machines (SVMs) classification errors have been theoretically analyzed under the PAC learning model. Poisoning attacks in SVMs have been addressed by considering data sanitization as a form of outlier detection, multiple classifier systems, incremental learning and robust statistics. Evasion attacks in SVMs have been addressed by explicitly embedding knowledge of adversarial data manipulation into the learning algorithm using (i) game theoretical models for classification, (ii) probabilistic models of data distribution drift under attack and (iii) multiple classifier systems.

Biggio *et al.* [36] demonstrate that an intelligent adversary can predict change in SVM's decision function due to adversarial input. Poisoning attacks against an SVM inject adversarial examples into training data to increase the SVMs testing error. The

attack proposed by Biggio *et al.* [36] has an incremental learning technique with a gradient ascent strategy. The gradient is computed based on properties of the SVM's optimal solution. Since the attack depends on gradient (of dot products between points in input space), the attack is also kernelized by using both linear and non-linear kernels in the input space. For increasing the testing error, the gradient ascent procedure converges to local maxima of the non-convex validation error surface. The proposed gradient ascent strategy assumes that the adversary knows the training data used by the learning algorithm. In real world attack scenario, a substitute training dataset could be used instead of the original training dataset. The convergence of proposed gradient ascent strategy depends on the smoothness of SVM parameters and the manifold geometry of data points found in the solution of a quadratic programming problem. The proposed attack strategy can also be extended to a coalition of attacks where choosing the best subset of data points for attack is a subset selection problem.

Huang *et al.* [241] propose an adversarial learning algorithm for attacks on SVMs that maximize classification error by flipping labels in the training data. The proposed contamination attack is a poisoning attack because it targets SVM's testing error (also called empirical risk in PAC model) by contaminating the training data labels. The proposed adversarial data manipulation is called label noise injection. Two attack algorithms are proposed to account for adversarial data manipulations. Both algorithms assume that the adversary has access to the feature set of training data. Each label flip by adversary is assumed to have equal cost that is independent of the feature values in the sample. The first algorithm greedily maximizes SVM's test error through continuous relaxation of the label values in a gradient ascent procedure. The second algorithm does a breadth first search to greedily construct sets of candidate label flips that are correlated to the SVM's testing error. Both algorithms can be understood as a search for labels that achieve maximum difference between empirical risk for classifiers trained on original data and contaminated data. The algorithms can also be used to simulate a constant sum game between the attacker and the classifier whose aim is to respectively maximize and minimize testing error on the untainted test dataset. Different game formulations can be simulated if the players use non-antagonistic objective functions. Improvements to the algorithms are possible by the study of an incremental SVM under label perturbations. The problem of label noise injection creating the attacker manipulation in an SVM is also related to the classification problems for SVMs in semi-supervised learning, active learning and structured prediction.

2.4.4 Robust Classifier Ensembles

Biggio *et al.* [32] propose that an ensemble of linear classifiers can improve not only accuracy but also robustness of supervised learning. That is because more than one classifier has to be evaded or poisoned to compromise the whole ensemble of classifiers. The training strategy evenly distributes the feature weights between discriminative and non-discriminative features in data. Undermining the discriminative weights in the classifier can then undermine accuracy of the classifier. The objective of robust classifier ensembles is then to find such a correct tradeoff between robustness and accuracy. Here an adversary is forced to modify a large nature of feature values to manipulate the classifier.

Biggio *et al.* [32] design Boosting and Random Subspace Method (RSM) to distribute weights in the adversarial algorithm. The adversarial behaviour is modelled in terms of two scenarios - a worst -case scenario where the adversary has complete knowledge of the classifier and an average-case scenario where the adversary has only an approximate knowledge of the classifier. The ensemble discrimination function is then obtained by averaging different linear classifiers trained on different randomly selected subsets of the original feature set.

The averaging method by Biggio *et al.* [32] to find ensemble performance is an extension of the idea to use average performance of linear classifier to prevent overfitting or underfitting in imbalanced data. By reducing the variance component of classification or estimation error, the randomized sampling used in the algorithm reduces instability in decision or estimation function. Such a stable decision function is not supposed to undergo large changes in output for small perturbations in input data due to adversarial data or stochastic noise.

In Biggio *et al.* [32] the experimental evaluation has two objectives. The first objective is to understand the conditions under which randomized sampling produces an evenly distributed weight distribution in an ensemble of classifiers. The second objective is to evaluate whether the evenly distributed weights improve robustness of classifiers ensemble as compared to a single base classifier. Thus randomization based sampling techniques are shown to be useful in design of pattern recognition systems in adversarial environments.

Biggio *et al.* [33] extend adversarial environments in linear classifiers to randomisation-based multiple classifier systems (MCS). The MCS combines linear base classifiers via bagging and random subspace sampling. For improvements to classification accuracy and robustness, MCS's weight distributions are investigated for more even distribution

of weight values than single classifier weights. In worst-case attacks, adversary is assumed to have complete knowledge of the feature set, classifier parameters and the decision function. In non-worst-case attacks, adversary is assumed to have an incomplete knowledge of classifier's decision function. Classifier robustness is then evaluated as a function of attack strength, representing maximum number of features which can be modified by the adversary. In non-worst-case attacks, adversary approximates feature weights by overestimating or underestimating the importance of most discriminant features in classification performance. In worst-case attacks, adversary is supposed to modify the features to minimize decision function and maximize decrease in the classifer performance.

Biggio *et al.* [31] formally measure hardness of evasion for an adversary targetting pattern classification systems in general and a ensemble classifier architecture in particular. Hardness of evasion is taken into account in the choice of features and choice of classifier architecture. It is defined as expected value of minimum number of features to be modified by an adversary who seeks to evade classifier. It is calculated on disjoint subsets of discriminant (class-conditioned iid) features' weights assumed to be equally distributed amongst multiple classifiers with same decision functions. Classifier parameters are chosen to minimize a classification cost given in terms of the false positive and the false negative errors.

2.4.5 Robust Clustering Models

Adversarial clustering problems cannot be solved by clustering stability criteria that address stochastic noise in dataset, rather than targeted adversarial manipulations. Biggio *et al.* [38] devise poisoning and obfuscation attacks for single-linkage hierarchical clustering. In such poisoning attacks, the adversary's goal is injecting adversarial examples in the clustering quality measure. In obfuscation attacks, the adversary's goal is to hide data samples in existing cluster by manipulating the feature values. In these attacks, a cluster is defined as not only the hard partition (and the soft partition) of partitional clustering algorithms but also as the dominant sets (and parameterized hierarchy of subsets) in linkage-type clustering algorithms.

Biggio *et al.* [38] put additional constraints on attack scenarios with a distance metric between non-manipulated training data and manipulated adversarial data. Degree of knowledge of the adversary is encoded by entropy of a probability distribution in an attack sample. The probability distribution is defined over the knowledge space of the adversary giving information about the dataset and its parameterization in the

clustering algorithm. Supposing such an adversarial knowledge, adversary's goal is an objective function expressed in terms of a (i) real-valued distance measure between clusterings evaluating attack samples for poisoning attacks and (ii) nonnegative real scalar divergence measure between attack samples and target samples. Here, a greedy heuristic dendrogram cut criteria represents the single-linkage hierarchical clustering output as a binary matrix of probabilities assigning samples to clusters.

2.4.6 Robust Feature Selection Models

Xiao *et al.* [240] provide an adversarial framework to investigate poisoning attacks on feature selection methods such as LASSO, ridge regression and elastic net. Such feature selection is used to derive security-sensitive actionable information in large-scale high-dimensional data-driven technologies like spam detection, malware detection, web-page ranking and network protocol verification. Xiao *et al.* [240] assume feature selection to be a problem of filtering a relevant feature subset inferring a iid random process for the training data. Feature selection criteria is then represented as an optimization of an objective function such as classification error and prediction information gain.

Xiao *et al.* [240] define the adversary's goal in terms of a security violation which can be categorized as any one of integrity violation, availability violation and privacy violation in feature selection. Integrity violation slightly modifies selected feature subset to facilitate subsequent evasion attack. Availability violation compromises feature selection algorithm to produce an output feature subset with largest generalization error. Privacy violation reverse-engineers feature selection process to infer information about feature subset, training data and system users. A targeted attack affects specific feature subset while indiscriminate attack affects selection of any feature.

Xiao *et al.* [240] suppose that adversay's knowledge can be about assumptions on training data, feature representation, feature selection algorithm, and feature selection criteria. Then adversary's influence can be either causative or exploratory to affect either training data or testing data respectively. Here, poisoning attacks for feature selection manipulate feature values and labels in training data to construct poisoning samples that will be misclassified subsequently. Evasion attacks for feature selection manipulate testing data to evade detection by proposing distance measures and adversarial strategies to compare original data, non-manipulated attack sample and attack sample. Such adversarial strategies are expressed in terms of the adversary's knowledge, the adversary's capability, the adversary's goal, and the adversary's influence in affecting

the computation of an adversarial loss function empirically simulating feature selection algorithms on poisoned data.

For each feature selection algorithm in the experiments, Xiao *et al.* [240] optimize an adversarial loss function with a (sub) gradient-ascent algorithm solving a convex optimization problem. The feature spaces are assumed to define continuous and discrete features, differentiable and non-differentiable features. To evaluate the feature selection under attack settings, a stability index is proposed to indicate anti-correlation rankings between feature subsets of the feature selection algorithm. Experiments show that an adversary can easily compromise feature selection algorithms that promote sparsity in the feature representation. The poisoning attacks and evasion attacks are said to mislead model's decision making by introducing model bias and model variance respectively into the feature selection algorithm's mean squared error decomposition.

Mei *et al.* [167] poison Latent Dirichlet allocation (LDA) corpus so that LDA produces adversarially manipulated topics in LDA user decisions. Adversarial attack is formulated as a bilevel optimization problem for variational inference under budget constraints. It is solved by a computationally efficient gradient descent method based on implicit functions. The optimization employs a KL-divergence between LDA learner's word-topic distribution and fully factorized variational distribution constrained by Karush-Kuhn-Tucker (KKT) conditions. The adversary poisons training corpus such that topics learnt by LDA are guided towards target multinomial distributions defined by the adversary. Adversary's goal is to minimize an attacker risk function which defines distance between adversarial multinomial distribution and training multinomial distribution. Adversary's risk combined with learner's KL-divergence gives a bilevel optimization framework for constructing the adversarial examples. Adversarial examples on words and sentences misleading LDA topics are created on a corpus sourced from the United States House of Representatives floor-debate transcripts, online new year's wishes, and TREC AP newswire articles.

2.4.7 Robust Anomaly Detection Models

Kloft *et al.* [127] explore adversarial examples for an (online centroid) anomaly detection algorithm. The adversarial attack scenario is expressed in terms of the efficiency and the constraints of formulating an optimal attack on outlier detection. The outlier detection finds unusual events across finite sliding windows in computer security applications such as automatic signature generation and intrusion detection systems. A poisoning attack is assumed to create adversarial examples on training data where a certain percentage of

training data is controlled by the adversary. An anomalous data point is then measured according to the Eucledian distance from empirical mean of the training data. The empirical mean is calculated on training data by a finite sliding window online algorithm for non-stationarity data. By pushing the empirical mean point towards adversarial examples, the adversary forces the anomaly detection algorithm to accept anomalous data point as normal training data.

Kloft *et al.* [127] express the relative displacement of original empirical mean in terms of the attack direction vector between the attack point and the mean point. A greedy optimal attack is then proposed to locate attack points in a Voronoi cell on data points that maximize relative displacement of the empirical mean. For a Eucledian norm, the greedy attack is optimized with either a linear program or a quadratic program. The mixing of normal points and attack points is modelled by Bernoulli random variables which are iid in a kernel Hilbert space. The attack progress is measured by projecting the current empirical mean onto an attack direction vector. Theoretical analysis is provided for bounding the expectation and the variance of relative displacement by number of training points and attack points in the current sliding window. The adversary is assumed to have full knowledge of the training data and the anomaly detection algorithm. The anomaly detector's defence to adversarial attack is proposed in terms of controlling the false positive rate.

Rubinstein *et al.* [204] evaluate poisoning attacks and training defenses for (Principal Component Analysis) PCA-subspace anomaly detection where principal components maximize robust measures of training data dispersion. Adversary's goal is expressed as increasing false positives and false negatives of model under attack. A time series of traffic volumes between pairs of points is the dataset represented a routing matrix. Robust PCA of the routing matrix then identifies volume anomalies in an abnormal subspace. Adversary's poisoning strategies consider attacks with increasing amounts of variance information in the attack scenarios. The weakest attack strategy knows nothing about the traffic flows and adds random noise as adversarial examples. In locally-informed attack strategy, the adversary intercepts information about current traffic volume on network links under attack. In globally-informed attack strategy, the adversary has knowledge of traffic volumes on all network links and network levels. In short-term attack, the anomaly detector is retrained for each week of training data during which adversary attacks network. In long-term attack, anomaly detector's principal components are slowly poisoned by the adversary over several weeks. In each attack scenario, the adversary decides quantity of data to add to target traffic flow according

to a Bernoulli random variable. A robust PCA analysis on adversarially altered routing matrix then produces adversarial examples which are classified as innocuous by the anomaly detector. A tractable analytic solution to robust PCA is derived by Rubinstein *et al.* [204] from an objective function with relaxation approximations maximizing the attack vectors projected onto normal subspace covariance matrices. A Projection Pursuit method then produces feasible solutions for the objective function in direction of its gradient.

Feng *et al.* [75] present adversarial outliers for logistic regression. Linear programming procedure estimates logistic parameters in presence of adversarial outliers in the covariance matrix in binary classification problems. Non-robustness of logistic regression to adversarial outliers is calculated from the maximum likelihood estimate of log likelihood's influence function as well as the loss function in high-dimensional training data that has been corrupted by the adversarial outliers. In the attack scenario, adversarial outliers seek to dominate correlations in the objective function of the logistic regression model. Robustness bounds are then derived on the population risk and the empirical risk with Lipschitz continuous loss functions.

2.4.8 Robust Task Relationships Models

Zhao *et al.* [252] propose data poisoning attacks on relatedness of tasks in multi-task relationship learning (MTRL). Optimal attacks in MTRL solve a bilevel optimization problem adaptive to arbitrary target tasks and attacking tasks. Such attacks are found by a stochastic gradient ascent procedure. The vulnerability of MTRL to adversarial examples is categorized into feature learning approaches, low-rank approaches, task clustering approaches and task relationship approaches where the learning goal is to jointly learn a prediction function. Then MTRL of linear prediction functions with arbitrary convex loss functions and positive semi-definite covariance matrices is studied. Adversary's goal is defined as degradation of the performance of a set of target tasks by injecting poisoned data to a set of attacking tasks. Adversary's payoff function is defined as empirical loss of training data on target tasks where adversary has complete knowledge of the target MTRL model. In the gradient ascent procedure, poisoning data is iteratively updated in the direction maximizing the adversarial payoff function. The prediction function is a least-squares loss function for regression tasks and squared hinge loss function for classification tasks. The prediction performance is evaluated by maximizing area under curve for classification tasks and minimizing normalized mean squared error for regression tasks.

2.4.9 Adversarial Machine Learning in Cybersecurity

Adversarial attacks have several applications in computer vision, natural language processing, cyberspace security and the physical world [193]. In computer vision, adversarial attacks are created for image classification [173] and object detection [242]. In natural language processing, adversarial attacks are created for text classification [207] and machine translation [71]. In cyberspace security adversarial attacks are created for cloud services [153], malware detection [200], [95], intrusion detection [111].

In physical world, adversarial attacks are created to scale adversarial training to large models and datasets [136]. Kurakin *et al.* [135] discuss physical world scenarios with cameras and other sensors as input. Eykholt *et al.* [73] generate robust visual adversarial perturbations under different physical conditions for the real-world case of road sign classification. Here computer vision tasks act as control pipelines in physical systems where the main challenge with generating robust physical perturbations is environmental variability. Melis *et al.* [168] create physical world attacks on robot-vision systems. Sharif *et al.* [209] create physical world attacks on facial biometric systems using Face Recognition models for surveillance and access control. Xiao *et al.* [239] spatial transformation of adversarial perturbations with L_p distance as a metric of perceptual quality of the penalizing adversarial perturbations. Akhtar *et al.* [3] survey adversarial attacks on deep learning in computer vision. In comparison to game-theoretic attacks, the physical world attacks physically change the appearance of an object to deceive trained detection. They are restricted to once-only attack plays applied to targeted threats [219] in the game theoretical settings. They seem applicable to generating the stochastic search policy in our game with search heuristics such as monte carlo tree search [43] in combinatorial game theory.

Deep learning methods can be used to advance cyber security objectives such as detection, modeling, monitoring, analysis and defense against various threats to sensitive data and security systems [159]. Rossler *et al.* [201] discuss synthetic image generation and manipulation benchmarks based on DeepFakes, Face2Face, FaceSwap and NeuralTextures as prominent representatives for facial manipulations in data-driven forgery detectors. Matern *et al.* [166] exhibit artifacts from face tracking and editing to expose manipulations in face editing algorithms like Deepfakes and Face2Face.

Further applications of adversarial machine learning in cybersecurity include malware detection, malware classification, spam detection, phishing detection, botnet detection, intrusion detection and intrusion prevention and anomaly detection. Tong *et al.* [224] discuss evasion attacks in PDF malware detection. Melis *et al.* [169] apply

explainable machine-learning models in Android malware detection. Marino *et al.* [163] explain incorrect classifications in data-driven Intrusion Detection Systems. Corona *et al.* [60] provide a taxonomy of adversarial attacks in Intrusion Detection Systems (IDSs) and computing infrastructures. Demetrio *et al.* [66] propose feature attribution to provide meaningful explanations to the classification of malware binaries. Fleshman *et al.* [79] quantify system robustness of machine learning based anti-virus products using malware detection models.

2.5. Game Theoretical Adversarial Deep Learning Models

Stochastic games defining strategy spaces for adversarial manipulations have been used to generate adversarial examples [118]. Such a strategy space is defined in terms of two or more adversaries actions and corresponding payoff functions. Each of such an adversary can engage one or more learners in a game and vice versa. From the learner's standpoint, adjusting parameters of a game theoretical model is computationally less expensive than building a new model that is robust to the adversarial manipulation. From the adversary's standpoint, the attack scenarios can be characterized by the stochastic optimization parameters estimated in the game theoretical interactions with the learner.

2.5.1 Game-Theoretic Learning Models

The ideas of two player Sequential games (or Stackelberg Game) and multiplayer Cooperative games have been employed as game theoretic frameworks training adversarial learning algorithms. To search for equilibrium in such games is equivalent to solving a high dimensional optimization problem. The eventual model performance is then estimated by stochastic optimization methods based on computationally efficient heuristic search algorithms. So long as the objective function is bounded, global optimization methods such as genetic algorithms, simulated annealing, stochastic hill climbing can be applied to search for the convergence criteria that lead to subgame perfect equilibria.

Globerson *et al.* [86] discuss a classification algorithm with a game theoretic formulation. The proposed algorithm is robust to feature deletion according to a minmax objective function optimized by quadratic programming. In Liu *et al.* [150], the interactions between an adversary and data miner are modelled as a two-player sequential

Stackelberg zero-sum game where the payoff for each player is designed as a regularized loss function. The adversary iteratively attacks the data miner using the best possible strategy for transforming the original training data. The data miner independently reacts by rebuilding the classifier based on the data miner's observations of the adversary's modifications to the training data. Such a game is repeated until the adversary's payoff does not increase or the maximum number of iterations is reached. Liu *et al.* [152] propose an extension to Liu *et al.* [150] where a one-step game is used to reduce the computing time of the minmax algorithm. The one-step method converges to Nash equilibrium by utilizing Singular Value Decomposition (SVD). Liu *et al.* [246] formulate a bi-level optimization problem from a non-zero sum game on adversarial data transformations. The game experiments with sparse regularizers for designing robust classification objectives.

A game ends in an equilibrium with payoffs to each player based on their objectives and actions. The learner has no incentive to play a game that leads to too many false positives with too little increase in true positives. The adversary has no incentive to play a game that increases the utility of false negatives not detected by the learning algorithm. At equilibrium, the adversary is able to find testing data that is significantly different from the training data whereas the learner is able to update its model for new threats from adversarial data.

All player's are assumed to act in their rational interest to maximize the payoffs. This assumption, at every stage of game, eliminates Nash equilibria with non-credible threats to the learner and creates an equilibrium called the subgame perfect equilibrium. Here perfect equilibrium assumes that each player knows about the others utility function. The players utility functions vary by application domain.

2.5.2 Game theoretical adversarial learning

Dalvi *et al.* [62] analyzed classifier performance by viewing classification as a game with the classifier adapting to an adversary, aiming to make the classifier produce false negatives. Here a cost-sensitive adversary is combined with a cost-sensitive classifier to define adversarial classification in a game theoretical framework. In adversarial classification, the data generating process is allowed to change over time such that the data change can be expressed as a function of classifier parameters. Classifier is then assumed to maximize an expected payoff over adversary's cost parameters. In turn, the adversary's strategy is to find classification feature changes that maximize

adversary's expected payoff. Nash equilibria are demonstrated for both sequential games and repeated games where parameters of both players are known to each other.

Lowd *et al.* [155] introduced adversarial algorithms to learn a linear classifier's decision boundary. An ACRE learning framework is used to determine whether an adversary can efficiently learn enough about defeating a classifier by minimizing a linear adversarial cost function.

Biggio *et al.* [37] defined poisoning attacks against Support Vector Machines (SVMs) by injecting adversarial examples into training data. A gradient ascent procedure computes adversarial examples as local maxima of SVM's non-convex error surface.

Bruckner *et al.* [45] proposed prediction games to model interaction between a learner building predictive models and a data generator controlling data generation process. Kantarcioglu *et al.* [119] designed a subgame-perfect Nash equilibrium, which optimizes attribute selection with cost functions in an adversarial classification Stackelberg Game. Liu *et al.* [151] modelled competing behaviour between a rational adversary and a black-box data miner as a sequential Stackelberg game. Chivukula *et al.* [57] enhanced [151] proposals for deep learning models while Yin *et al.* [247] extended them for sparse attack scenarios. Zhou *et al.* [253] explored a nested game framework, where adversarial strategy is chosen according to a probability of making prediction about classifier's decision boundary in a single leader multiple followers game.

In this research, we evaluate multi-label adversarial learning algorithms in stochastic optimization settings. We empirically generate adversarial manipulations at Nash equilibrium in a constant-sum and a variable-sum sequential Stackelberg game. Our adversary's strategy space is determined by evolutionary parameters and variational parameters learnt on the input data distribution. Therefore the optimal adversarial manipulations found by our adversaries define a generative model for the adversarial data found in classifier's input data space.

Furthermore, we define adversarial cost functions on a strategy space encoding original data distribution. Our adversarial payoff functions are optimized by a simulated annealing algorithm randomizing step changes in adversary's strategy spaces. Randomization in our adversarial attack strategies is also defined by the latent space reconstructing original data distribution with a Variational Autoencoder (VAE). The proposed (variational nonlinear non-convex) adversarial cost function leads to better regularization of the adversarial payoff function converging to a Nash equilibrium in our Stackelberg game.

2.5.3 Game theoretical adversarial deep learning

In Liu *et al.* [150], the interactions between an adversary and data miner are modelled as a two-player sequential Stackelberg zero-sum game where the payoff for each player is designed as a regularized loss function. Each player's move is based on the observation of the opponent's last play. The adversary iteratively attacks the data miner by best strategy for transforming the original training data. The data miner reacts by rebuilding classifier based on data miner's observations of the adversary's modifications to the training data. The adversary's strategy of play is determined independently by the adversary. The game is repeated until adversary's payoff does not increase or the maximum number of iterations is reached.

The maxmin problem for optimization proposed in Liu *et al.* [150] is solved without making assumptions on the distribution underlying training and testing data. The empirical evaluation of the optimization algorithm is conducted on image spam and text spam data. Different settings of loss functions yield different types of classifiers such as logistic regression with log linear loss function and support vector machines with hinge loss function. For the chosen loss functions, the optimization objective is formulated as a unconstrained convex optimization problem. The optimization problem is solved by the trust region method minimizing objective function on a constrained neighbourhood of polar coordinates. At Nash equilibrium the solution of the maxmin problem achieves highest false negative rate and lowest data transformation cost simultaneously. This leads to robust classification boundaries at the test time. The weight vector computed at Nash equilibrium also gives features that are more robust to adversarial data manipulations.

Liu *et al.* [152] propose an extension to Liu *et al.* [150] where one-step game is used to reduce computing time of the minmax algorithm. The one-step method converges to Nash equilibrium by utilizing Singular Value Decomposition (SVD). SVD gives orthogonal basis vectors or singular vectors acting as the 'principal components' of training data. Thus the singular vectors characterize each type of class present in the training data. The label of a test data is then taken by Liu *et al.* [150] to be the training class generating smaller residue vector. This SVD driven classification algorithm is taken to be the initial state of the game theoretical model. The adversary goal is to transform target class (or positive class) instances into the negative class. This goal is achieved in the testing data by shifting positive instances in training data by a small amount such that the class distribution is skewed towards positive class label. Moreover, the payoff function for the adversary is formulated as the difference in singular vectors before and after the adversarial data manipulation. Thus, a rational adversary not only attempts to mini-

mize the distance between distributions of negative instances and transformed positive instances but also minimizes the transformation itself. The payoff for the adversary is then approximately solved using a trust region sub-problem which is solved using a subspace approximation while avoiding the expensive computation of the gradient matrix and hessian matrix of the adversary's loss function and classifier's loss function respectively. In validating the algorithm, the adversarial examples are constructed to produce high false positive rate for the classifier at the initial step of the game. At the end of the game, the false positive rate of the learning algorithm is reduced by taking into account adversarial data manipulations.

Wang *et al.* [231] assumes that adversary changes any feature of the classifier at will and pays a cost proportional to size of the feature subset that has been changed. Such an attack on classifier is called sparse feature attack in the paper. The minmax optimization problem is then formulated as a non-zero sum game. In a non-zero sum game, the gain of classifier is not necessarily the loss of adversary. Regularized loss functions are proposed for both the data miner and adversary to make the game's objective a convex bi-level optimization problem. Both l_1 and l_2 regularizers are examined with respect to the proposed sparse models. The adversary is assumed to apply change in data by minimizing the changes to loss function. The adversary selects an attack strategy with full knowledge of the data miner's feature weighting strategy. In regularizing the adversary's loss function, both the number of positive samples and the number of negative samples are used to account for imbalance in the data. Then data miner chooses the feature weights based on samples in the manipulated data space. The goal of data miner is to determine a decision boundary based on continuously manipulated data in each step. The goal of adversary is to determine a manipulating vector based on a given budget in each step. These steps are repeated sequentially until convergence. Upon convergence, the classifier finds feature weights that are robust against the proposed sparse feature attacks. For solving a l_2 regularized least squares objective, the adversary's data manipulations are assumed to be bounded by a perturbation matrix of l_1 norms that set a reasonable budget (or accumulated cost) as the convergence criteria for the adversary. The elements of perturbation matrix are tuned to the input data by cross validation over training and testing data ordered in time. The various games are then simulated by various l_1 and l_2 regularizers on the perturbation matrix. The choice of regularizers for the data miner leads to trade-offs between sparsity and accuracy, bias and variance of the classifiers. Experimental evaluation validates that game-theoretic classifiers deteriorate at a slower rate than regular classifiers on both near future and far future data.

To derive the payoff functions in the game, we assume that the adversary has no knowledge of either the deep neural network layers or loss functions in the deep learning model. Our proposed game theoretical optimization problems are solved without making assumptions on the learner's training and testing data distributions. The strategy space for algorithmic randomization and data manipulation in our game is determined by the stochastic operators in evolutionary algorithms and variational networks defining the attack scenarios.

2.5.4 Stochastic Games in Predictive modelling

The interaction between an adversary and the classifier has been modelled as a Stackelberg Game. Here adversary's role is not that of a static data generator but an intelligent agent making deliberate data manipulations to evade classifiers. Failure of considering adversarial evasion in classifier design exposes security concerns in fraud detection, computer intrusion detection, web search, spam detection and phishing detection applications. Re-learning classifier weights is a weak solution to robust classification since evasion attacks are generated at cheaper and faster rate than re-learning.

Li *et al.* [144] proposed a feature cross-substitution attack to demonstrate objective-driven adversaries exploiting limitations of feature reduction in adversarial settings. Adversary is able to query the classifier according to a fixed query budget and a fixed cost budget. An adversarial evasion model with a sparse regularizer is then presented. Constructing the classifier on feature equivalence classes rather than feature space is proposed as a solution to improve classifier resilience. Another solution proposes bi-level Stackelberg game of interactions between classifier and a collection of adversaries. Stackelberg game is solved by mixed-integer linear programming with constraint generation.

Bianchi *et al.* [50] presented repeated games for random prediction problems. The problem of sequential prediction is modelled in the framework of Nash equilibrium found in normal form games. Specific minmax theorems are discussed to analyze two-player zero-sum games. A mistake bounds framework is provided to analyze game theoretical learning algorithms. Bruckner [44] proposes prediction games to model the interaction between a learner who builds the predictive model and the adversary who controls the process of data generation. Prediction games framework allows explicit models on game-theoretical players' interests, actions, knowledge and decisions. Then the generalization error of a predictive model is analyzed in terms of the Nash equilibrium by solving non-cooperative two-player static and dynamic Stackelberg games. Stochastic two-player

zero-sum games incorporating multiple adversaries were analyzed by Ummels [228]. Cai *et al.* [254] analyzed polymatrix games that provide multiplayer generalizations to two-player zero-sum games. The corresponding Nash equilibrium strategies may deviate from max-min optimization and are obtained from linear programming solutions.

Zhou *et al.* [254] surveyed two-player and multiple player Stackelberg games in adversarial learning algorithms and cyber security applications. The interaction between adversary and classifier is modelled as one or more of simultaneous games, sequential games where adversary can be either a leader or a follower in the game. Alpcan *et al.* [7] presented large-scale strategic games and reduced games computation consumption and information limitation on Nash equilibrium solutions. Oliehoek *et al.* [182] proposed a deep generative adversary with resource-bounded best responses and Nash equilibrium on synthetic data. The generative adversary has a generator network and a discriminator network in a supervised learning problem and operated on discrete data. Specifically, the generator network's loss function depends only the "fake" data whereas the discriminator network's loss function depends on both "real" data and "fake" data. Both the generator and discriminator participate in a zerosum strategic-form game where each player's payoff function is defined on a mixed strategy space.

Papernot *et al.* [185] provided a threat model summarizing various attack scenarios in adversarial learning algorithms. The threats to machine learning models are adversarial manipulations, which are generated during both training process and inference process. In the training stage, adversaries can manipulate the data collection processes by injecting adversarial examples into the training data with intent of modifying learning model's decision boundaries. In the inference stage, adversaries can plan blackbox or whitebox attacks on learning model's parameters to cause distribution drifts between training data distribution and runtime data distribution. Papernot *et al.* [185] also proposed no free lunch theorem and probably approximately correct model for adversarial learning. Yang *et al.* [245] analyze the Nash equilibrium of a differential dynamical system modelling the Advanced persistent threat (APT) cyberattack scenario.

We propose new nonconvex best responses in every play of the prediction game solving for the adversarial manipulations. Our bilevel stochastic optimization problem in the prediction game is formulated as a repeated sequential variable-sum two-player Stackelberg game. The optimization problem is solved by an Alternating Least Squares (ALS) search procedure that continuously attacks retrained classifier with adversarial manipulations optimized until Nash equilibrium. The ALS procedure evaluates candidate adversarial manipulations generated by a Simulated Annealing (SA) procedure for an

increase in adversarial payoff function over the targeted class labels. Therefore, the adversarial data generated in the Stackelberg game simulates continuous interactions rather than one-time interactions with the learning processes of the classifier.

2.5.5 Nash equilibria and Stackelberg Strategies in Differential Games

In a Stackelberg game, adversarial strategies are modeled and solved for the solution rationale and decision-making problem defining the Nash equilibria. The solution space for Nash equilibria are expressed in terms of the necessary and sufficient conditions for game players' convergence criteria [212]. Typical convergence critera are (i) zero-sum game vs nonzero-sum game; (ii) two-player vs multiplayer game; (iii) static game vs evolutionary game; (iv) sequential game vs continuous game; (v) deterministic game vs stochastic game. Typical players' strategies consider cases where a pair of players (i) do not know each other's performance criteria; (ii) compute each other's strategies at different speeds; (iii) have linear and non-linear payoff functions that may or may not be discontinuous; (iv) participate in a game with distributed control vs decentralized control. In such games, the Stackelberg Strategies and Nash equilibria are analyzed in terms of the structural properties of the coefficient matrices of higher-order matrix-Riccati differential equations.

The optimization of such game theoretical payoff functions presents a complex problem in optimization theory. Such problems are often modelled as decision problems in noncooperative differential games [42]. The solutions to these problems are presented as Pareto optima, Nash and Stackelberg equilibria, co-co (cooperative-competitive) solutions for the payoff function.

The Riccati differential equations are also analyzed as differential games in optimal control theory. If the game theoretical players can observe state of the control system, then the Nash equilibrium is computed according to an open-loop solution for the control system. If the game theoretical players cannot consider feedback strategies then the Nash equilibrium is computed according to a closed-loop solution for the control system. Principles of dynamic programming are used as the computational methods finding the game theoretical optima to the necessary and sufficient conditions for optimal control system.

Furthermore, partial differential state equations of the control system can augment the players payoff functions to result in stochastic control [4] in game theoretical in-

teractions. Here the game theoretical equilibria are determined by the necessary and sufficient conditions on the coefficients solving for the Stackelberg Riccati differential, difference and algebraic equations [83]. The study of such equilibria and their numerical computational methods is the subject of evolutionary and differential game theory [23].

2.6. Adversarial Defense mechanisms

Brendel *et al.* [41] categorize threat models generating adversarial perturbations into (i) gradient-based attacks relying on detailed model information, (ii) score-based attacks relying on confidence scores such as class conditioned probabilities, (iii) transfer-based attacks relying on substitute models for target models, and (iv) proposed decision-based attacks relying on information about final model decision. Proposed decision-based attacks are called Boundary Attack and applied to blackbox models for target model.

2.6.1 Securing classifiers against feature attacks

Li *et al.* [144] demonstrate limitations of feature reduction in adversarial settings with objective-driven adversaries. Each adversary is supposed to be able to substitute across similar features in a feature cross-substitution attack. Adversary is also assumed to be able to query classifier according to a fixed query budget and cost budget. An evasion model with sparse regularizer is presented in adversarial setting. Constructing classifier on feature equivalence classes rather than feature space is proposed as a solution to improve classifier resilience to evasion model. Another solution proposes bi-level Stackelberg game of interactions between classifier and a collection of adversaries. Stackelberg game is solved by mixed-integer linear programming with constraint generation. Adversaries' objectives are inferred from (query budget and cost budget) constraint generation converging to local optima on training data.

Globerson *et al.* [86] analyze classifier robustness with a game theoretical formulation. For a classsifier trained on multiple features with varying importance, any single feature is not given too much weight during testing. Adversary is supposed to be able to delete features in testing data that were present in training data. Then a classifier is constructed which is optimal under a worst case feature deletion scenario. Such a scenario is formulated as a solution to a two player game between classifier and feature deleter with a minmax objective. Classifier chooses actions that give robust classifier parameters. Feature deleter chooses to delete features that are most harmful to classifier

performance. Structure of uncertainty in game convergence is related to existence vs non-existence of a feature. A support vector machine with regularized hinge loss and linear constraints is taken to be the training objective for classifier. Game deletes features that lead to maximum decrease in classifier loss. Cooperative games with Shapley value objectives measuring performance change after deleting a feature are an alternative to the proposed minmax objectives deleting multiple features simultaneously.

Given an evasion attack scenario, Zhang *et al.* [248] investigate impact of feature reduction on classifier security, if adversary-aware feature selection is not present in classifier training. A smaller feature set is shown to significantly worsen classifier performance under attack which may usually not be the case for a classifier not under attack. Classifier security model is expressed as a regularizer to be optimized and estimated along with classifier's generalization capability during feature selection process. It is implemented as a wrapper-based feature selection method using forward-selection and backward elimination of features suitable for both linear and nonlinear classifiers with differentiable discriminant functions. Evasion attack scenario is considered an exploratory integrity attack on the testing data fed to a classifier trained on original training data. Optimal evasion strategy is formulated as an optimization problem minimizing the distance between the adversarial examples and the training data such that classifier's discriminant function misclassifies adversarial examples. An adversary-aware feature selection approach maximizes not only the generalization capability of the classifier but also the classifier security against evasion attacks. Here the classifier security is weighted by application-specific constraints and parameters while the classifier generalization capability is estimated according to the application-dependent discriminant functions and performance measures. Instead of searching for best evasion point by querying classifier with candidate samples of a blackbox search approach, computationally efficient adversarial algorithms are devised to exploit adversary's knowledge of targeted classifier's objective function. They are defined by choices for distance function between adversarial examples and training data, feature representation in classification algorithm. A gradient-descent procedure finds gradient steps that reduce distance between adversarial data and training data while projecting current point onto feasible domain of adversarial examples as soon as discriminant function misclassifies it. Initial attack point in gradient-descent is set to closest sample that is either classified as legitimate or classified as malicious. Performance of true classifier gracefully decreases against attacks of increasing attack strength determined by upper bound on maximum amount of adversarial modifications and lower bound on misled classifier confidence.

A most gracefully degraded classifier is expected to be most secure after retraining on training data as well as adversarial examples. Experiments validating Student's t-tests and classification accuracies of feature weight distributions are conducted on TREC 2007 email corpus consisting of legitimate and spam emails. Application-specific constraints on data distributions make it harder for adversary to imitate feature values of legitimate class eventually leading to a low probability of evading detection.

2.6.2 Adversarial classification tasks with regularizers

Demontis *et al.* [67] analyze evasion attacks of linear classifiers in a robust optimization framework. Relation between sparsity of feature weights and defence of linear classifiers is investigated to propose a regularizer. Linear classifiers are chosen in adversarial learning algorithm due to their interpretable decisions obtained from the low storage, processing time and power consumption in mobile and embedded systems. Adversary is supposed to have complete knowledge about target classifier's training data, feature set and classification algorithm. Adversary's capability of modifying data is given as an application dependent data constraint. Typically, such data constraints are defined as ℓ_1 and ℓ_1 norms on number of modified features called sparse and dense attacks respectively. Adversary's attack strategy is formulated as an optimization problem minimizing target classifier's discriminant function for data subject to a distance constraint between adversarial examples and the original data. With an idea of finding sparse and uniform weights, a linear convex combination of ℓ_1 and ℓ_∞ norms is proposed as a robustness regularizer for adversary's attack strategy. Behaviour of such regularization against evasion attacks on a support vector machine classifier with hinge loss is then investigated in classification applications for handwritten digit classification, spam filtering and malware detection. Performance measurement in adversarial settings is done with area under the ROC curve combined with a sparsity and security measures proposed on classifier's weight distributions.

2.6.3 Adversarial Reinforcement Learning

Mandlekar *et al.* [160] synthesize whitebox attacks in deep reinforcement learning policies. Behzadan *et al.* [25] demonstrate adversarial examples that are transferable across various Deep Q-Networks. The spatio-temporal features of the training process are conjectured to provide defence mechanisms against such adversarial examples. Kos *et al.* [132] create poisoning attacks over time in deep reinforcement learning. The

adversarial examples in image classification settings are compared with adversarial examples in reinforcement learning settings. Learning agent's policy resilience through re-training is also investigated. Ilyas *et al.* [114] improve blackbox attack scenarios with bandit optimization based gradient estimation. Pinto *et al.* [190] train a reinforcement learning agent in the presence of a destabilizing adversary. The adversary applies differences in training and testing conditions as disturbance forces in the reinforcement learning. The policy learning trajectory is then formulated as solution to a two-player zero-sum Markov game. Li *et al.* [145] discuss operational constraints in the adversarial evasion of security policies. The task of adversarial classification is separated into the task of learning to predict attack preferences and the task of optimizing operational policy that explicitly abides by the operational constraints on the predictor. Then adversary's best response strategies are computed as randomized operational decisions.

Jun *et al.* [117] propose reward-manipulation attack protocols in online learning with limited feedback. The adversarial objective is to promote or obstruct actions chosen by a stochastic contextual bandit algorithm. Ma *et al.* [157] propose data poisoning attacks to hijack the behaviour of a contextual bandit in an online recommender system. The adversarial data is found by solving a quadratic program with linear constraints. Lin *et al.* [147] propose adversaries that lure the agent through a preferred sequence of actions to a designated target state. A generative model is used to plan and predict the future states of the agent. Ho *et al.* [108] propose a generative learning framework to learn imitation learning policies from expert trajectories. The policies are learnt by generative algorithms that bypass learning of the cost functions of the maximum causal entropy in inverse reinforcement learning. Goyal *et al.* [92] train the generator in generative adversarial networks with a Temporal Difference (TD) objective rather than gradients of the discriminator. Pfau *et al.* [189] view generative adversarial networks as actor-critic methods where actor cannot affect the reward. Finn *et al.* [78] reformulate minmax games in deep generative models as bilevel optimization problems.

2.7. Computational optimization algorithms

In generalized least squares models and generalized linear models for predictive analytics, classification loss functions optimize the class-conditioned data likelihood functions [101, 104] of the targeted deep networks. In this thesis, the adversarial cost functions regularize such likelihood functions with norms, gradients and expectations of game theoretical objective functions inferred on the adversarial loss functions. The types

of such objective functions determine the types of adversaries participating in prediction games with the classifier. In this thesis we have proposed adversaries solving for evolutionary objectives and variational objectives in the prediction games. The optimal values for the objectives are searched by evolutionary algorithms such as genetic algorithms, simulated annealing algorithms and alternating least squares algorithms.

In this section we review additional computational algorithms, stochastic operators and convergence criteria for computational optimization in deep learning models. Such a study is expected to lead us to better randomization, convergence and parallelization in computation of the step magnitude and the step direction in our stochastic optimization methods [215]. In designing the iterative update rules of optimization algorithms and fitness functions solving for systems of equations, we are interested in robust optimization, numerical optimization and nonlinear optimization. In addition to game theoretical models, deep learning optimizations of our interest include utility functions found in expectation maximization algorithms, maximum entropy models, learning classifier systems, deep factorization machines and probabilistic graphical models.

Fogel [80] categorizes the simulated evolution techniques in stochastic optimization of neural networks. Depending on the facet of natural evolution (that is viewed as optimizing problem-solving process), the techniques are called genetic algorithms, evolution strategies, and evolutionary programming. These techniques do not use higher-order statistics of the fitness function to converge onto optimal solutions. These techniques are not as sensitive as gradient based methods to adversarial perturbations in the fitness function. Pirlot [191] describes the strengths and weaknesses in Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithms (GAs). Ledesma *et al.* [143] review the procedure to practically implement simulated annealing. Bandyopadhyay *et al.* [20] use simulated annealing to minimize misclassification rate across decision boundaries in pattern classification. A deterministic annealing algorithm is proposed by Rose [199] to optimize the problems related to clustering, compression, classification and regression. A hybridization of GA and SA is given by Adler [1]. SA is combined with local search methods into a Markov chain by Martin *et al.* [164]. Back *et al.* [13] and Beyer *et al.* [28] survey developments in Evolution Strategies (ESs) that allow correlated mutations in GA. Das *et al.* [63] review all major theoretical studies and algorithm variants of Differential Evolution (DE) applied to multiobjective, constrained, large scale, and uncertain optimization problems. Pelikan *et al.* [188] propose linkage learning across candidate solutions in evolutionary computations.

Zhang *et al.* [250] survey machine learning problems in an evolutionary computation

framework. Goldberg [87] provides more detail on applications of genetic algorithms in machine learning. Michalewicz [170] discusses numerical optimization of the genetic operators to lead to evolutionary programs. Bandaru *et al.* [18, 19] describe descriptive models and predictive models for data mining in multi-objective optimization datasets. Bertsekas [27] discusses derivative free stochastic optimization problems. Nemirovski *et al.* [177] discuss convex-concave stochastic optimization of objective functions given in the form of an expectation integral. Sinha *et al.* [213] review evolutionary solutions to bilevel optimization problems. Suryan *et al.* [220] review evolutionary algorithms in inverse optimal control theory that has applications in game theory.

The operation of evolutionary algorithms in constrained environments is analyzed by Eiben [2]. Cantu-Paz [47] provides a survey of parallel constructions in genetic algorithms. Ocenasek *et al.* [181] surveys the designs for parallel estimation of distribution algorithms. Sudholt [69] introduces design and analysis of parallel evolutionary algorithms on multicore CPU architectures. Genetic algorithms have been implemented in embarrassingly parallel programming models such as MapReduce [76, 229]. Whitley *et al.* [235] give guidelines for debugging and testing evolutionary computations. The no free lunch theorems for optimization [236] apply to the comparisons between optimization criteria of evolutionary computations. Whitley *et al.* [181] provide a theoretical analysis of the research problems, objective functions and optimization algorithms in evolutionary computations. Comon *et al.* [59] propose an Enhanced Line Search (ELS) principle to apply the Alternating Least Squares (ALS) algorithm in iterative optimization of nonlinear systems of equations represented by tensor decompositions. A theoretical analysis of Simulated Annealing (SA) solving for Boltzmann machine and Cauchy machine is given by Tsallis *et al.* [227].

2.7.1 Derivative-free Stochastic Optimization

Evolutionary Algorithms (EA) have been used in stochastic optimization to generate rule-based data mining models with attribute interactions [249]. The EA-based stochastic search and optimization algorithms are Evolutionary Programming (EP), Evolutionary Strategies (ES), Genetic Algorithms (GA), Differential Evolution (DE), Estimation of Distribution Algorithm (EDA) and Swarm Intelligence (SI) algorithms [234], [105].

In our adversarial algorithm, the search and optimization algorithm is either a genetic algorithm or a simulated annealing algorithm. The adversarial data samples are generated by the selection, crossover, mutation search operators in the genetic algorithm and the annealing search operator in the simulated annealing algorithm. By using

probabilistic hill climbing algorithms over Markov chains in multivariate models, the current search operators can be extended to define explicit probabilistic distributions performing a complex neighbourhood search for the candidate solutions [5].

CHAPTER 3

GAME THEORETICAL ADVERSARIAL DEEP LEARNING WITH EVOLUTIONARY ADVERSARIES AND STOCHASTIC ADVERSARIES

In this chapter we discuss the problem formulation for the proposed adversarial learning algorithm with evolutionary adversaries. A stochastic multiplayer game is formulated in terms of multiple two player sequential games.

3.1. Sequential game formulation

The training algorithm simulates the adversarial learning as a constant sum Stackelberg game between two players. The two players are called Leader (L) and Follower (F). The leader initiates the game by making the first action/move/play. In our algorithm, the adversary is the leader and the learner is the follower. In a constant sum game, the learner's loss is assumed to be the adversary's gain and vice versa.

Each player is associated with strategy spaces A and W for L and F respectively. The strategy space is a choice of moves available to each player. The outcome of a strategy is determined by the player's payoff function J_L and J_F. For a given observation of $w \in W$, the best strategy $\alpha^* \in A$ for the leader is

$$\alpha^* = argmax_{\alpha \in A} J_L(\alpha, w). \tag{3.1}$$

Similarly for L's move α, F's best strategy is

$$w^* = argmax_{w \in W} J_F(\alpha, w). \tag{3.2}$$

Moreover, the sum of payoff functions J_L for the adversary and J_F for the classifier is assumed to be a constant profit Φ. This allows us to rewrite the expression for w^* in terms of J_L

$$w^* = argmax_{w \in W} \Phi - J_L(\alpha, w) = argmin_{w \in W} J_L(\alpha, w). \tag{3.3}$$

Combining Equation 3.1 with Equation 3.3 we formulate the following maxmin problem in the game.

$$Maxmin : (\alpha^*, w^*) = argmax_{\alpha \in A} J_L(\alpha, argmin_{w \in W} J_L(\alpha, w)). \tag{3.4}$$

We train a Convolutional Neural Network (CNN) as the learner. With knowledge of only the learner's classification error, the adversary is assumed to target the true positives. In each iteration of the game, the learner trains the weights w in the CNN layers for the input α presented by the adversary. The adversary then adapts the data manipulations to the weights trained by the CNN. Thus, each player's move is based on the opponent's last play. The game is initiated by the adversary. Thus the adversary is the leader L and the learner is the follower F. Each game iteration is composed of the moves made by L and F.

Using an evolutionary algorithm, the adversary searches for data manipulations that maximize classification error $error(w)$. The fitness function in evolutionary algorithm $J_L(\alpha, w)$ is defined to be the adversary's payoff function. The fitness function increases with iterations of the game. The game converges when the adversary does not see an increase in the payoff function or the maximum number of iterations is reached. The game convergence criteria depend on the search and optimization criteria of the evolutionary algorithm used in each game iteration. The game converges to an adversarial data manipulation α^* on the learner with weights w.

For labelled input training data X_{train}, X_{test} available during the game, the adversary searches for a move α that maximizes the following payoff function or fitness function $J_L(\alpha)$ where $error$ is the classification error as measured by $recall$ for the current adversarial data. The term $cost$ is the ℓ_2 norm for the current α.

$$J_L(\alpha, w) = 1 + \lambda * error(w) - cost(\alpha), \tag{3.5}$$

$$error(w) = 1 - recall(w). \tag{3.6}$$

$$cost(\alpha) = ||\alpha||_2. \tag{3.7}$$

The negative $cost(\alpha)$ term in Equation 3.5 ensures that the adversary minimizes changes to the current α while maximizing the positive $error(w)$ term. By definition of the fitness function, $error(w)$ is maximized by minimizing the corresponding $recall(w)$. $recall(w)$ is computed for each iteration of the game on the manipulated training data $X_{train} + \alpha$ so that the α which gives the maximum value for $J_L(\alpha, w)$ is selected for subsequent iterations in the game. $cost(\alpha)$ is enhanced by an weighting term λ that is empirically evaluated for each dataset. A constant 1 is then added to $J_L(\alpha)$ to ensure a positive fitness function in the evolutionary algorithm.

Therefore, from a theoretical standpoint, we characterize the statistical difference between the input training data distribution X_{train} and adversarial testing data distribution $X_{test} + \alpha^*$ in terms of the adversary's cost $cost(\alpha)$ and the learner's error $error(w)$. During the game's iterations, the manipulation of the training data distribution X_{train} into $X_{train} + \alpha$ allows us to find the α that maximizes the adversary's payoff $J_L(\alpha, w)$. After game convergence, the manipulation of the testing data distribution X_{test} into $X_{test} + \alpha^*$ allows us to find the α^* that minimizes learner's payoff $J_F(\alpha, w)$. Equation 3.5 is solved for α^* – the additive pixel level manipulations generating the adversarial manipulations. α^* allows us to find an adversarial testing data distribution $X_{test} + \alpha^*$ that is non-stationary with respect to the original training data distribution X_{train}.

Various multiplayer multi-label game formulations would allow us to optimize various adversarial manipulations on the input data distribution. Equation 3.5 can also be solved for more types of adversarial manipulations whose cost of generation $cost(\alpha)$ is characterized by structured adversarial manipulations on complex data structures like tensors and graphs increasingly found in big data models.

From an application standpoint, we observe that the adversary's interest is in converting illegitimate data to legitimate data with a minimum of changes and not vice versa. To account for this objective, we assume the learner's error to be classification recall. By converting true positives to either false positives or false negatives, the adversary tries to reduce the true positives in the learner's performance regardless of the changes to the true negatives. The adversary's attack scenarios are expressed in terms of the parameter settings in the evolutionary algorithms. Each different parameter setting allows us to converge onto a different adversarial data manipulation on the testing data distribution.

From the perspective of improving the robustness of deep learning networks, our formulation generates the manipulated data distributions maximizing the error of the deep network and measuring the cost of the corresponding attack scenarios generating

data manipulations over the training data in adversarial learning. A successful attack scenario allows us to estimate a manipulated data distribution that leads to statistically significant testing error. The estimation of neural networks weights on such data distributions is the area of our research and investigation.

3.2. Stochastic game formulation

In this section, we generalize Equation 3.5 for multiple adversaries participating as players in the game. At the end of the game, we propose a learner that is robust to the adversarial data generated by each adversary.

Suppose a set L of M adversaries $L = \{L_1, L_2, L_3, ..., L_M\}$ associated with the strategy space A are attacking the single learner F associated with the strategy space W. The outcome of all M adversaries strategy set $\mathbb{A}_S = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_M\}$ is determined by the vector $\boldsymbol{J_L}$ of M adversary payoff functions $\boldsymbol{J_L} = \{J_{L_1}, J_{L_2}, J_{L_3}, ..., J_{L_M}\}$ such that J_{L_i} corresponds to α_i where $i = 1, 2, 3, ..., M$. For a given best observation of $w^* \in W$ and best estimate of $\alpha^* \in A$, under the same assumptions, we can generalize the maxmin problem in Equation 3.4 to the following Equation 3.8 for a multiplayer constant sum stochastic game.

$$Maxmin : (\mathbb{A}_S^*, w^*) = argmax_{\mathbb{A}_S \in A} \boldsymbol{J_L}(\mathbb{A}_S, argmin_{w \in W} \boldsymbol{J_L}(\mathbb{A}_S, w)). \tag{3.8}$$

In this research, we assume that the vector $\mathbb{A}_S$ is the set of its component scalar α_i where $i = 1, 2, 3, ..., M$, so that the multiplayer stochastic game can be simplified as many two-player sequential games. Each two-player game is played as a sequential game following Equation 3.4 and Equation 3.5. The output of all the games is a set $\mathbb{A}_S^*$.

Upon convergence, each game i outputs adversarial data manipulation α_i^* where $i = \{1, 2, 3, ..., M\}$. Each α_i^* is added to the original training data distribution X_{train} and original testing data distribution X_{test} to create final adversarial manipulations $X_{train} + \mathbb{A}_S^*$ and $X_{test} + \mathbb{A}_S^*$.

Definition 1. $X_{train} + \mathbb{A}_S^* = \{\cup_{i=1}^{M} \{X_{train} + \alpha_i^*\}\}$

Definition 2. $X_{test} + \mathbb{A}_S^* = \{\cup_{i=1}^{M} \{X_{test} + \alpha_i^*\}\}$

We choose a Convolutional Neural Network (CNN) as the learner. The CNN architecture's input layers and output layer are described in Krizhevsky *et al.* [133] and available in the Tensorflow [1] as the CIFAR10 model. The CNN has input layers consisting of

[1] `https://www.tensorflow.org/tutorials/deep_cnn#cifar-10_model`

convolution layers, maxpooling layers, regularization layers and activation units. The CNN has output layer of the softmax probability distribution function. The overall loss function of the learner is defined by the CNN's input and output layers.

From multiplayer game output $\mathbb{A}_S^*$, we define the following CNN models corresponding to manipulated training data distribution $X_{train} + \mathbb{A}_S^*$ and testing data distribution $X_{test} + \mathbb{A}_S^*$. X_{train} is sampled from a given MNIST database $X_{original}$ as well as a Generative Adversarial Network (GAN) output $X_{generated-gan}$.

Definition 3.
$CNN_{original} = CNN(X_{train}, X_{test})$,
$X_{train} = Sample(X_{original})$,
$X_{test} = Sample(X_{original})$

Definition 4.
$CNN_{manipulated-cnn} = CNN(X_{train}, X_{test} + \mathbb{A}_S^*)$,
$X_{train} = Sample(X_{original})$,
$X_{test} = Sample(X_{original} + \mathbb{A}_S^*)$

Definition 5.
$CNN_{manipulated-gan} = CNN(X_{train}, X_{test} + \mathbb{A}_S^*)$,
$X_{train} = Sample(X_{generated-gan})$,
$X_{test} = Sample(X_{original} + \mathbb{A}_S^*)$

Definition 6.
$CNN_{secure} = CNN(X_{train} + \mathbb{A}_S^*, X_{test} + \mathbb{A}_S^*)$,
$X_{train} = Sample(X_{original} + \mathbb{A}_S^*)$,
$X_{test} = Sample(X_{original} + \mathbb{A}_S^*)$

3.3. Stochastic game illustration

Figure 3.1 illustrates the learning process in the game formulation as a flow chart. The $CNN_{original}$ is trained on training data X_{train} and evaluated on testing data X_{test} given as 'learner performance' in the experiments described in Section 3.6. Figure 3.1 illustrates a two-player game. The game has moves executed by each of the adversaries and the learner during each interaction. In these moves, an adversary targets the learner by the adversarial sample produced from the evolutionary operators. The learner then

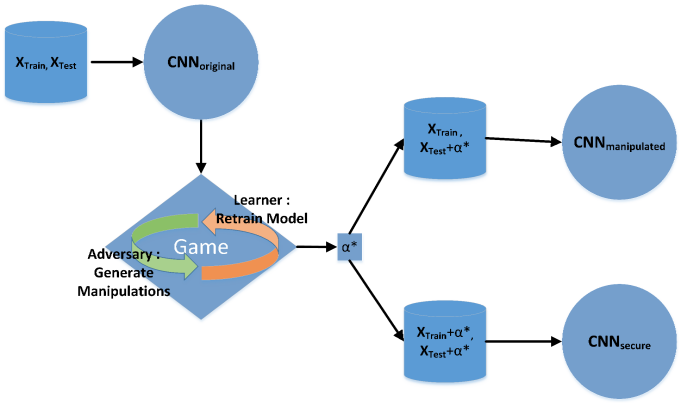


Figure 3.1: A flow chart illustrating the benefits of a game theoretic learner. The two-player game is played by a single adversary and one Learner. The game produces a final deep learning network CNN_{secure} that is better equipped to deal with the adversarial manipulations than the initial deep learning network $CNN_{original}$.

adapts the deep learning operators for the adversarial data by retraining the CNN on the new cross-validation sample.

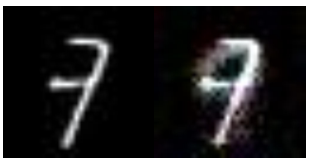
A set L of M adversaries $L = \{L_1, L_2, L_3, ..., L_M\}$ targets this performance by engaging the CNN in multiple two-player sequential games. In each two-player game, the CNNs trained on the original and generated data samples and tested on the adversarial data are $CNN_{manipulated-cnn}$, $CNN_{manipulated-gan}$ respectively. All these CNNs are given under the umbrella term 'manipulated learner performance' in the experiments in Section 3.6. We find that $CNN_{manipulated-cnn}$ as well as $CNN_{manipulated-gan}$ are significantly worse performing than the original CNN $CNN_{original}$ trained on the original training and testing data (X_{train}, X_{test}). Thus we conclude adversarial manipulation succeeds in attacking the learner. A new Convolutional Neural Network CNN_{secure} is then retrained on $(X_{train} + \mathbb{A}^*_S, X_{test} + \mathbb{A}^*_S)$ to adapt to adversarial manipulations. It is given as 'secure learner performance' in the experiments described in Section 3.6. CNN_{secure} is our proposed model. It is found to be better than the manipulated CNN's $CNN_{manipulated-cnn}$ and $CNN_{manipulated-gan}$.

Therefore, we conclude that the new CNN_{secure} has successfully adapted to adversarial data generated by multiple adversaries while the given $CNN_{original}$ is vulnerable to each adversarial manipulation α^*_i generated by each adversary L_i playing a game i on the given training/testing data distributions. Our algorithm is able to find a data sample that affects the performance of a CNN. The CNN that is able to recover from our adversarial attack is better equipped to deal with unforeseen changes in the underlying data distribution. The game between adversary and learner allows us to produce

(a) 2 manipulated to look like 8

(b) 3 manipulated to look like 8

(c) 4 manipulated to look like 9



(d) 7 manipulated to look like 9

Figure 3.2: Examples of transformed images found at Nash equilibrium in a Stackelberg game. To avoid detection, the adversary adds pixels in (a) and (d), and changes shape in (b) and (c).

adversarial data manipulations for a CNN trained on the underlying data distribution.

3.3.1 Illustrative Adversarial Examples

Figure 3.2 shows examples of data transformations on positive labels that appear to be negative labels in the adversarial algorithm solving a stochastic game. Some of the transformations that avoid detection are adding and deleting pixels, and changing the shape and size of the image. In Figure 3.2(a), the handwritten digit 2 has been manipulated to look like a 8 by adding pixels. In Figure 3.2(b), the handwritten digit 3 has been manipulated to look like 8 by changing the thickness and shape. In Figure 3.2(c), the handwritten digit 4 has been manipulated to look like 9 by adding pixels and deleting pixels. In Figure 3.2(d), the handwritten digit 7 has been manipulated to look like 9 by adding pixels but not changing the shape.

Figure 3.3 has examples for data transformations on positive labels that appear to be negative labels in the adversarial algorithm solving a sequential game. Some of the transformations that avoid detection are adding and deleting pixels, changing shape and size of the image. In Figure 3.3, Figure (a) has handwritten digit 2 manipulated to looks like 8 by adding pixels. Figure (b) has handwritten digit 3 manipulated to looks like 8 by changing thickness and shape. Figure (c) has handwritten digit 4 manipulated to looks like 9 by deleting pixels and changing shape. Figure (d) has handwritten digit 7 manipulated to looks like 9 by deleting pixels and changing shape.

Compared to adversarial manipulations in Figure 3.2, the manipulations in Figure 3.3

(a) 2 manipulated to look like 8

(b) 3 manipulated to look like 8

(c) 4 manipulated to look like 9

(d) 7 manipulated to look like 9

Figure 3.3: Examples of transformed images found at Nash equilibrium in a Stackelberg game. To avoid detection, the adversary adds pixels in (a), changes shape in (b), deletes pixels and changes shape in (c) and (d)

visibly change the positive class into data points that look more like negative class and less like positive class. To reduce the chance of creating such manipulations in our experimental validation criteria, we set attack parameter values to give more importance to the adversarial cost rather than the classification cost.

3.4. Stochastic Game Algorithm

In this section we discuss the adversarial learning algorithm with a set L of M adversaries participating in a stochastic game. Each adversary L_i participates in a sequential two-player game i with the CNN to output α_i^* according to either Algorithm 2 or Algorithm 7.

Algorithm 1 solves for $\mathbb{A}_S^*$ in Equation 3.8. As input, the algorithm requires the labelled training data X_{train} and labelled testing data X_{test} Each record in the training and testing data is a tensor with pixel values representing an image. A variable M determines the number of adversaries participating in the multiplayer game. M is set to 1 for the two-player game. The variable $gametype$ allows each adversary in a multiplayer game to choose between GA and SA as the optimization procedure for a two-player game.

The cumulative effect of all the adversaries attack is validated by datasets $X_{train} + \mathbb{A}_S^*$ and $X_{test} + \mathbb{A}_S^*$ following Definition 1 and Definition 2 respectively. In Section 3.6.5 and

Table 3.5, these datasets are used to validate the CNN F1-score performance according to Definition 3, Definition 4, Definition 5 and Definition 6.

Algorithm 1 Multiplayer Game with Multiple Adversaries

Input:
1: Labelled training data X_{train}, Labelled testing data X_{test}, Number of adversaries M, Optimization flag variable $gametype$
Output:
2: Adversarial manipulations $\mathbb{A}_S^*$, Attack performance F1-score$_{manipulated}$, Retraining performance F1-score$_{secure}$
3: **for** $i \in [1..M]$ **do** ▷ Iterate i on M adversaries choosing either GA or SA as flag variable $gametype$
4: **Begin**
5: **If** $gametype$ == GA **then**
6: α_i^* = twoplayergame-ga(X_{train})
7: **If** $gametype$ == SA **then**
8: α_i^* = twoplayergame-sa(X_{train})
9: **End**
10: $\mathbb{A}_S^* = \{\alpha_i^*\}, i = \{1,2,3,...,M\}$
11: Generate $X_{train} + \mathbb{A}_S^*$ from Definition 1
12: Generate $X_{test} + \mathbb{A}_S^*$ from Definition 2
13: Train CNN on X_{train}
14: Calculate F1-score$_{manipulated}$ by testing CNN on $X_{test} + \mathbb{A}_S^*$
15: Train CNN on $X_{train} + \mathbb{A}_S^*$
16: Calculate F1-score$_{secure}$ by testing CNN on $X_{test} + \mathbb{A}_S^*$
17: return $\mathbb{A}_S^*$, F1-score$_{manipulated}$, F1-score$_{secure}$

3.5. Sequential Game Algorithm

The pseudocode for a two-player sequential game i with output α^* is discussed in Section 3.5.1 and Section 3.5.2. By assuming that each sequential game i is independent of the remaining games in the stochastic game with M adversaries, we drop game number i in the sequential game pseudocode. The experiments in Section 3.6.3, Section 3.6.2 and Table 3.4 validate the sequential game. Section 3.6.2. Section 3.6.4 provides a conclusion to these experiments.

3.5.1 Genetic algorithm

Algorithm 2 gives the training algorithm for the two-player sequential game that takes into consideration adversarial attacks. As input, Algorithm 2 requires the training data X_{train}. Each example in the input data is a three dimensional tensor of RGB pixel values. Algorithm 3 calculates the fitness function values for candidate solutions. Algorithm 4, Algorithm 5, Algorithm 6 give the genetic operators used by Algorithm 2 to search for candidate solutions.

Algorithm 2 initializes the game by training the CNN on Line 2 to store the weights to disk. The fitness function values are computed on Line 5 for each randomly initialized α. α belongs to a genetic population $\alpha_{population}$ operating on the input data X_{train}.

$\alpha_{population}$ is randomly initialized around the mean of the positive class. It is assigned to $population$ on Line 4. A variable $maxpayoff$ is used to keep track of the adversary's current payoff in the current iteration of the game from Line 7 to Line 25. The if condition on Line 11 ensures that the algorithm converges when $maxpayoff$ does not exceed the current payoff $currpayoff$ by a small number 0.0001. In each game iteration, the current α_{curr} which gives the best fitness value is selected on Line 8. On Line 9, the CNN is retrained to react to attack $X_{train} + \alpha_{curr}$. The variable $maxpayoff$ is updated on Line 13 if α_{curr} satisfies the game convergence criteria.

From Line 14 to Line 20, the standard genetic operators selection, crossover, mutation, and clone are used to generate $population$ in the genetic algorithm. On Line 14, the selection operator conducts a weighted sampling without replacement where the weights are proportional to the fitness function values for the current $population$. The selection function in Algorithm 4 randomly samples the current population to return the selected candidates and the remaining candidates as the offspring and parents respectively. On Line 17, the crossover operation is applied between the odd children and even children in the offspring. The crossover function in Algorithm 5 randomly slices and swaps the pixels of the current children. The starting and ending indices for slicing are also selected randomly. On Line 19, random mutations are applied to the pixels of each offspring. The mutation function in Algorithm 6 applies a mask of randomintegers that are added to the pixel values in the current child. Since the masking allows for pixel values in the range of -255 and +255, any mutated pixel values crossing 255 and 0 are taken to be 255 and 0 respectively. The pixels for mutation are selected with a uniform probability.

The new population for the next iteration is cloned on Line 20. Line 21 calls Algorithm 3 to recompute the fitness function values for new $\alpha_{population}$. Line 4 of Algorithm 3 in turn calls an evaluation function to compute the performance metrics on the data subject to adversarial manipulation. These metrics are calculated subject to the current softmax probabilities of the learner. Line 6 of the Algorithm 3 calls Equation 3.7 to compute the cost of α^*, $cost(\alpha)$.

In Algorithm 2, Line 26 and Line 27 find α^* that is the final converged attack for the adversary. On Line 28, the training algorithm Algorithm 2 returns the final testing performance F1-score on input testing data X_{test} subject to adversarial data manipulation $X_{test} + \alpha^*$.

Algorithm 2 Two-player Game with Genetic Algorithm

function TWOPLAYERGAME-GA(X_{train})
  Train CNN on X_{train}
  $maxpayoff = 0$, exitloop = False
  $population = \alpha_{population}$ ▷ Initialize population to size ψ
  $F(X_{train}) = fitness(X_{train}, \alpha_{population})$

  while gen<maxiter $\wedge$ $\neg$ exitloop **do**
    $\alpha_{curr}, currpayoff = max(F(X_{train}))$
    Train CNN on $X_{train} + \alpha_{curr}$

    If abs(currpayoff - maxpayoff) < 0.0001 **then**
      Begin
        $maxpayoff = currpayoff$
        parents,offspring = selection(population,0.5)

        for child1 in odd offspring and child2 in even offspring **do**
          child1, child2 = clone(crossover(child1,child2))
        for mutant in offspring **do**
          mutant = mutation(mutant)
        population = clone(parents + offspring)
        $F(X_{train}) = fitness(X_{train}, \alpha_{population})$
      End
    else
      exitloop = True
  end while
  $\alpha_{curr}, maxpayoff = max(F(X_{train}))$
  $\alpha^* = \alpha_{curr}$
  return α^*
end function

Algorithm 3 Genetic Operators: Fitness function

function FITNESS($X, Y, \alpha_{population}$)
  for $\alpha \in \alpha_{population}$ **do**
    Begin
      $metrics = evaluate(X + \alpha, Y)$ ▷ Compute performance measures on manipulated data
      $error = \lambda * metrics['recall']$
      $\alpha_{fitness} = 1 + error - cost(\alpha)$ ▷ Update fitness values for population
    End
  return $\alpha_{population}$
end function

Algorithm 4 Genetic Operators: Selection function

function SELECTION(P, ζ)
  Retrieve fitness values W_P for all P
  Sample P without replacement biased by W_P for ζ percentage of children C
  return $P - C, C$
end function

Algorithm 5 Genetic Operators: Crossover function

function CROSSOVER($c1, c2$)
  Randomly slice $c1$ and $c2$ into $c1_{sliced}$ and $c2_{sliced}$ with minimum width η
  Swap $c1_{sliced}$ with $c2_{sliced}$
  return $c1, c2$
end function

Algorithm 6 Genetic Operators: Mutation function

function MUTATION(m)
  Randomly select a $step$ between δ and $-\delta$
  Randomly generate a boolean tensor $mask$ for m
  $m[mask] = m[mask] + step$ ▷ Slice m by mask and update by step
  return m
end function

3.5.2 Simulated annealing algorithm

Algorithm 7 gives the adversarial learning algorithm for attacks from adversaries using a simulated annealing algorithm. The game is played on labelled training data X_{train}. Algorithm 8 uses the annealing operator used to generate candidate solutions called α. The game converges onto a final solution α^*.

Algorithm 7 starts the game on Line 2 and ends the game on Line 43. The game iteration is defined in terms of the payoff function value $currpayoff$ which in turn is determined by the convergence criteria for the simulated annealing. The game is initialized between Line 2 and Line 7. On Line 2, a target CNN is trained on X_{train} and tested on X_{test}. The purpose of the game is to find a α^* such that $X_{test} + \alpha^*$ decreases the performance of this CNN found on X_{test}. On Line 3, two flagging variables $maxpayoff$ and $exitloop$ are created to control the convergence of the game. Line 4 initializes the variables controlling the convergence criteria for simulated annealing. Line 6 initializes the candidate solutions to be generated by simulated annealing. Line 6 initializes the mask to be used in the anneal operators. The current mask $mask$ is taken to be all the nonzero pixels contained in the mean images of both the positive class μ_+ and negative class μ_-. Line 7 computes the fitness function value $evalc$ for the initial solution α_c. The game's iteration is given between Line 9 and Line 41. The current temperature for simulated annealing is initialized on Line 4 and is updated on Line 34. On Line 11, the adversary's payoff in the game is taken to be the same as fitness function value for candidate solution α_g. For both the genetic algorithm and the simulated annealing algorithm, the fitness function is computed by the function defined in Algorithm 3 discussed earlier. The game is continued only when there is an increase seen in the payoff $currpayoff$ for the adversary on Line 13. The game ends and the program control shifts to Line 41 if there is no increase in $currpayoff$.

Within each iteration of the game, the candidate solutions are generated by the annealing operator between Line 17 and Line 35. For every temperature T_{curr}, ν number of candidate solutions α_n are generated. Initially α_n is generated from a randomly initialized α_c. Further solutions from simulated annealing are generated by the current α_c updated on Line 25 and Line 31 according to the searching criteria of the simulated annealing. α_c is also updated on Line 36 at the end of every iteration of the game. The combination of the search parameters sample size ν, reduction rate ρ, maximum temperature T_{max} and minimum temperature T_{min} determines the number and rate of generation of candidate solutions α_n. On Line 20, each α_n is obtained by applying the annealing operator $anneal$ on the best solution α_c defined in Algorithm 8. The masking

process used in the operator allows for the randomized selection of tensor regions in the game's search procedure. The random masks are generated by the Boolean tensor mask $maskb$ on Line 2 and the sliced index mask $mask_{sliced}$ on Line 6 alongwith the initialization mask $maska$ passed as a function argument on Line 1 of Algorithm 8. A step of tensor values between parameter $-\delta$ and δ is then applied on the region selected for adversarial manipulation.

On Line 21, the fitness function value $evaln$ is computed for every α_n. If a given candidate solution α_n is found to be fitter than the best solution α_c, then the best solution α_c is updated on Line 25. If an increase in adversary's payoff is seen on the given candidate solution α_n, then the adversarial manipulation α_g is updated on Line 27. The corresponding fitness function values $evalc$ and $evalg$ are also simultaneously updated. On Line 30, the search procedure in simulated annealing is randomly restarted with a probability $e^{(evaln-evalc)/T_{curr}}$ depending on the current fitness function and current temperature values. The values of α_c and $evalc$ are updated by the current candidate solutions α_n and $evaln$ at every such restart on Line 31. The current temperature T_{curr} is decreased on Line 34. The best solution α_c found from the simulated annealing iteration within the game iteration is updated on Line 36. At the end of the game, the α_g giving maximum payoff to the adversary is assigned to α^* on Line 41. Line 42 finds the manipulated performance F1-score of the CNN on adversarial data $X_{test} + \alpha^*$ which was trained in Line 2 on the original data X_{train}.

3.6. Experiments

In this section we discuss the experimental validation and stochastic parameters of the adversarial learning algorithm. During the game, an adversary finds adversarial data manipulations using either a genetic algorithm or a simulated annealing algorithm as the search algorithm. For a two-player game, various parameter settings produce adversarial manipulation α^* on the images such that the positive class examples are misclassified as negative class examples by the CNN aka learner. For example, the CNN misclassifies the handwritten digit 7 which had been positively labelled before adversarial manipulation as the negatively labelled handwritten digit 9 after adversarial manipulation. The CNN is then secured against attacks in a stochastic game with multiple adversaries by defending against adversarial manipulations in many sequential games with two adversaries. The performance of the proposed secure CNN model is also compared with the performance of a CNN model augumented by the data produced from

Algorithm 7 Two-player Game with Simulated Annealing Algorithm

function TWOPLAYERGAME-SA(X_{train})
    Train CNN on X_{train}
    $maxpayoff = 0$, exitloop = False
    $T_{max} = 1000, T_{min} = 5, \nu = 50, \rho = 0.6, T_{curr} = T_{max}$
    Randomly initialize $\alpha_c, \alpha_g, \alpha_n$ to same tensor values
    $mask = \mu_+ \wedge \mu_-$ ▷ Initialize the SA parameters and masks
    $evalc = fitness(X_{train}, \alpha_c)$

    while ¬ exitloop **do**
        $evalg = fitness(X_{train}, \alpha_g)$
        $currpayoff = evalg$

        If abs(currpayoff - maxpayoff) < 0.0001 **then**
            Begin
                $maxpayoff = currpayoff$

                while $T_{curr} \geq T_{min}$ **do**
                    $i = 1$
                    while $i \leq \nu$ **do**
                        α_n = anneal($\alpha_c, mask$)
                        $evaln = fitness(X_{train}, \alpha_n)$

                        If evaln > evalc **then**
                            Begin
                                $\alpha_c = \alpha_n, evalc = evaln$
                                If evalg < evaln **then**
                                    $\alpha_g = \alpha_n, evalg = evaln$
                            End
                        else
                            If random(0,1) <= $e^{(evaln-evalc)/T_{curr}}$ **then**
                                $\alpha_c = \alpha_n, evalc = evaln$
                        i+ = 1
                    end while
                    T_{curr} * = ρ
                end while
                $\alpha_c = \alpha_g$
            End
        else
            exitloop = True
    end while
    $\alpha^* = \alpha_g$
    return α^*
end function

Algorithm 8 Simulated Annealing Operators: Anneal function

function ANNEAL($\alpha, maska$)
    Randomly generate a boolean tensor $maskb$ for α
    $mask = maska \wedge maskb$
    Randomly generate a tensor $step$ with values between δ and $-\delta$
    ▷ $step$ has same shape as α
    Randomly slice $mask$ into $mask_{sliced}$
    $\alpha[mask_{sliced}]$+ = $step[mask_{sliced}]$
    ▷ Slice α by $mask_{sliced}$ and update by $step$
    return α
end function

various Generative Adversarial Network (GANs). In both the multiplayer game and the two-player game, we observe that the manipulated learner performance is lower than the original learner performance. Also, the secure learner performance is higher than the manipulated learner performance.

3.6.1 Dataset description

Class Labels	Positive Class	Positive Class Cardinality	Negative Class Cardinality
(2,8)	2	6990	6825
(4,9)	4	6824	6958
(1,4)	1	7877	6824
(5,8)	5	6313	6825
(3,8)	3	7141	6825
(7,9)	7	7293	6958
(6,8)	6	6876	6825
(2,6)	2	6990	6876

Table 3.1: Datasets of colour images used in the experiments

We use a cross-validation dataset of colour images split between two class labels. The dataset is taken from the MNIST handwritten images database [141]. The two class labels and their cardinality are described in Table 3.1. The lower digit is taken to be the positive class. For example, if the class labels are 7 and 9, the class label 7 is taken to be the positive class. The learner is Tensorflow's CIFAR10 CNN model [2] [133]. The learner's testing performance is the baseline performance targeted by the adversary in the game. We assume the adversary has a mean image of positive label data to initialize the population in genetic algorithm Algorithm 2. We also assume that the adversary has a mean image of positive label and negative label data to initialize the masking in simulated annealing algorithm Algorithm 7.

3.6.2 Genetic algorithm validation in Sequential Game

In this section we validate the adversarial data in a sequential game that is constructed by the mutation, crossover, and selection genetic operators defined on the images. The testing performance for mutation, crossover, selection operators and population size on the data manipulated by final α^* is reported in Figure 3.4. The x-axis has variation in the

[2] `https://www.tensorflow.org/tutorials/deep_cnn#cifar-10_model`

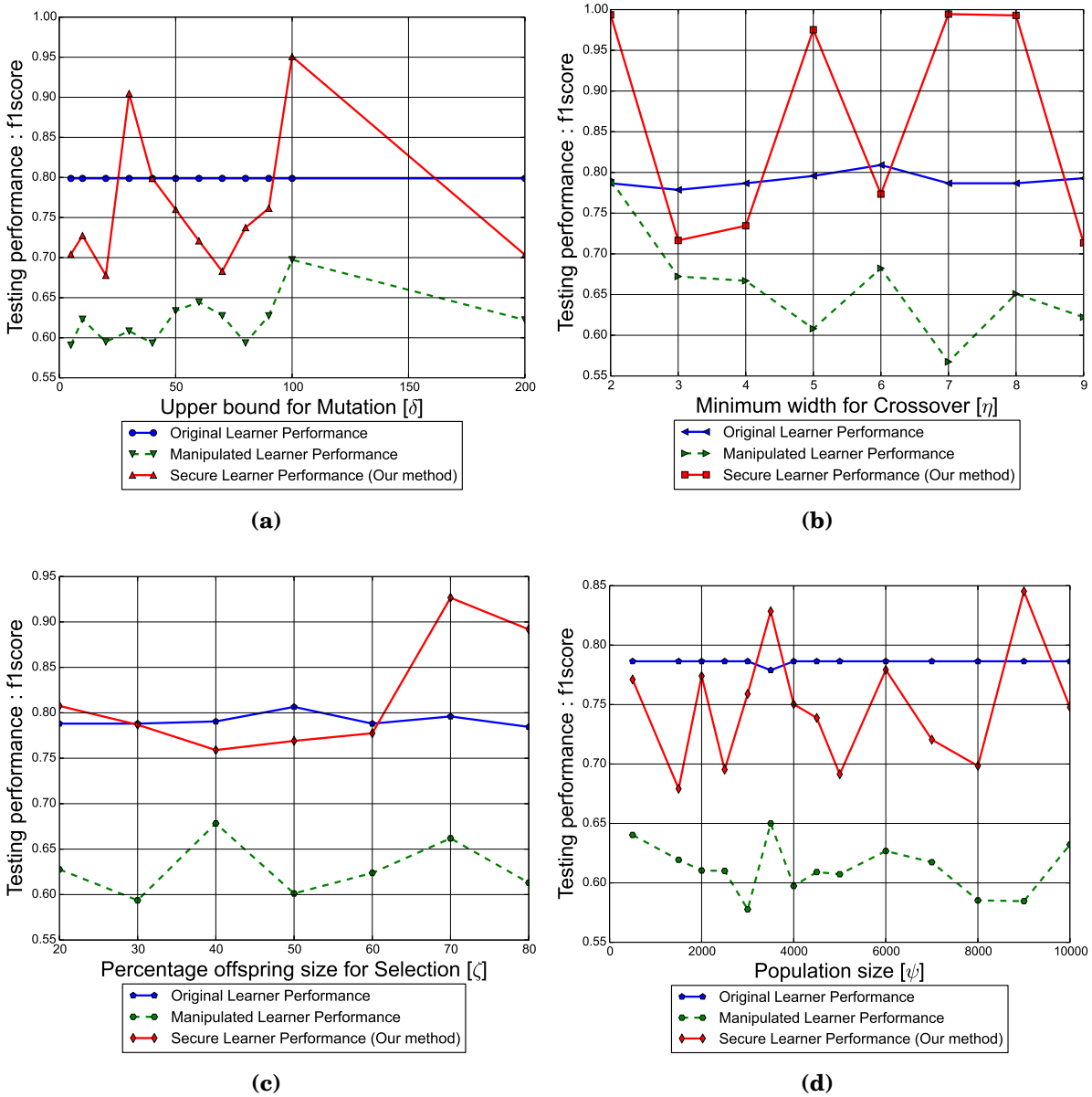


Figure 3.4: Testing performance with variations in evolutionary operators consisting of genetic parameters and annealing parameters. Genetic parameters are given in fig(a), fig(b), fig(c), and fig(d) for mutation step δ, crossover width η, selection size ζ, and population size ψ respectively. Annealing parameters are given in fig(e), fig(f), fig(g), and fig(h) for annealing step δ, annealing mask width η, annealing sample size ν, and annealing reduction rate ρ respectively. The manipulated learner has lower performance than the original learner. The secure learner has higher performance than the manipulated learner.

parameters for each genetic operator. The y-axis has the learner F1-score performance for the manipulated data.

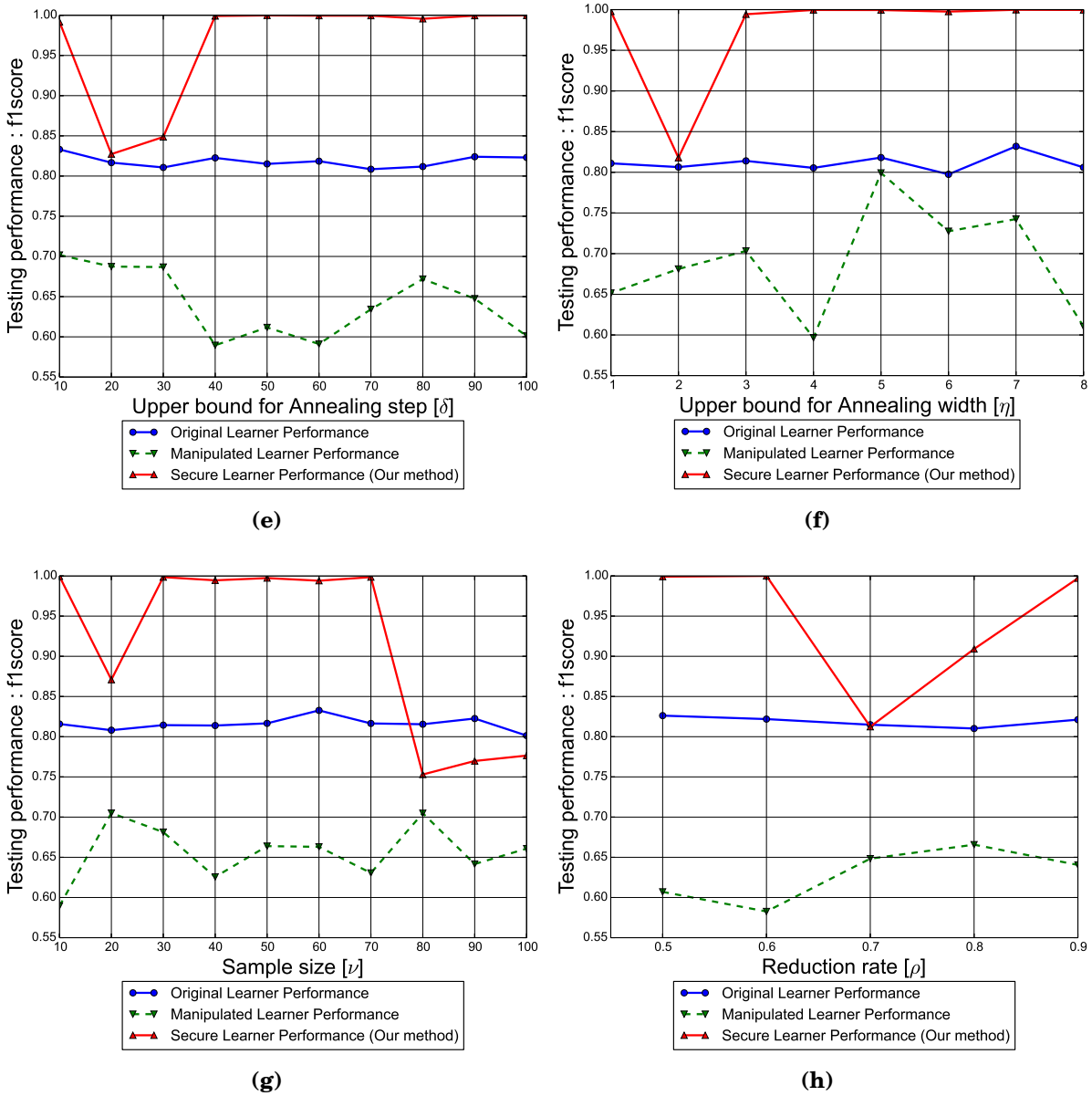


Figure 3.4: Testing performance with variations in evolutionary operators consisting of genetic parameters and annealing parameters. Genetic parameters are given in fig(a), fig(b), fig(c), and fig(d) for mutation step δ, crossover width η, selection size ζ, and population size ψ respectively. Annealing parameters are given in fig(e), fig(f), fig(g), and fig(h) for annealing step δ, annealing mask width η, annealing sample size ν, and annealing reduction rate ρ respectively. The manipulated learner has lower performance than the original learner. The secure learner has higher performance than the manipulated learner.

In the following, the genetic operators and parameters are described in more detail.

Population initialization In the initial $\alpha_{population}$ of the images, the pixel values

are randomly initialized around the meanimage of the positive class. The range for random pixel values is between the lower bound of RGB pixel value (-255) and the upper bound of RGB pixel value (+255). The size of the images is 32*32*3 as required by the CNN model. For the input images manipulated by adding α^*, the pixels with values greater than 255 are set to 255 and pixels with values less than 0 are set to 0.

Mutation operation A mask of randomly generated integers between a lower bound -δ (set to -50 by default) and upper bound +δ(set to +50 by default) for the step is added to the current image in the mutation operation. In Figure 3.4(a), the x-axis is varied by δ - the mutation step. From Figure 3.4(a) we conclude that the F1-score is minimized to 0.5 on the MNIST dataset for δ around 80.

Crossover operation For a three-dimensional 32*32*3 RGB image, the height and width indices are randomly selected. The starting index for height is selected between pixels 1 and 16 (half of the largest height). The ending index for height is selected between lower bound η (set to +2 by default) and η (set to +10 by default) from the corresponding starting index of height and upper bound 32. A similar random indexing scheme selects the starting width and ending width of the image. The slice of the starting and ending index of height/width over all the pixels in depth is then swapped between the two images in the crossover operation. In Figure 3.4(b), the x-axis is varied by η - the crossover width. From Figure 3.4(b) we conclude that the F1-score is minimized to 0.5 on the MNIST dataset for η around 7.

Selection operation The selection operation is an extension of random sampling without replacement. The parents for the next generation are randomly chosen from the current generation parents. A ζ (set to 0.5 by default) percentage of the current generation parents are selected to be the offspring for the next generation. The remaining candidates in the current generation of parents are preserved as parents for the next generation. The probability of selecting an offspring is proportional to the fitness values of the current parents. The selected offspring are then changed by crossover and mutation to get the parents for the next generation. Across every generation of the genetic algorithm, the size of the entire population (consisting of current offspring and parents) is fixed to the initial size of the parents. In Figure 3.4(c), the x-axis is varied by ζ - the selection size. From Figure 3.4(c) we conclude that the F1-score is minimized to 0.5 on the MNIST dataset for ζ around 30 %.

Population size In Figure 3.4(d), the x-axis is varied by ψ - the population size. ψ is supposed to have an effect on the randomization of the genetic operators. From Figure 3.4(d), we observe that the testing performance of the manipulated learner

decreases by around 20% from 0.79 to 0.60 for the F1-score. Then the testing performance of the secure learner trained on the manipulated data increases by around 15% from 0.6 to 0.75 for the F1-score. The highest decrease in the manipulated learner's performance is seen at population size 3000, 8000 and 9000. But the highest increase in the performance of the secure learner is seen at 9000. So we conclude that the secure learner performance increases with an increase in ψ. But the corresponding performance of the manipulated learner is seen to be periodically decreasing with ψ. ψ values of 3000 and 9000 seem to be the most suitable for the MNIST dataset.

We find higher values of $\lambda > 1$ are suitable for finding the most suitable genetic algorithm parameters. In all the figures we have used $\lambda = 10$ to minimize the F1-score and maximize the error term in the fitness function of Equation 3.5. For various settings of the genetic parameters, the game quickly converges into an α^* at Nash equilibrium in a maximum of 20 generations of the genetic algorithm. Randomization and the corresponding effect on the manipulated learner performance increases with attack strengths, iterations and population size in the game.

3.6.3 Simulated annealing validation in Sequential Game

In this section, we validate the adversarial data that is constructed by the annealing operator defined on the images. The testing performance for the annealing operator over steps, masks and its parameters settings on sample size, and reduction rate are reported in Figure 3.4. The x-axis shows variation in each parameter's values. The y-axis shows the learner F1-score performance for the original data and the manipulated data. In all the figures, we observe that the secure learner performance is higher than the manipulated learner performance and the manipulated learner performance is lower than the original learner performance.

In the following, the annealing operator and parameters are described in more detail.

Annealing step The annealing step parameter δ limits the upper and lower bounds for the pixel values added to the region of the image selected by the annealing operator. δ takes a default value of 20 and is varied between 5 and 100. The pixel values are randomly selected to be any integer value between $-\delta$ and δ. In Figure 3.4(e), the x-axis is varied by δ. From Figure 3.4(e) we conclude that a δ between 40 and 60 is most suitable for adversarial manipulation. Compared to the original F1-score performance, the drop in manipulated F1-score performance by varying δ varies from 15% to 30% whereas the gain in retrained F1-score performance varies from 15% to 50%. The annealing step parameter δ is similar to the mutation step parameter δ.

Annealing mask The annealing mask width parameter η limits the upper and lower bounds on the indices of the images that have been selected by the annealing operator for adversarial manipulation. η takes a default value of 2 for the lower bound and 10 for the upper bound for an image that is 32 pixels wide and 32 pixels high. The starting and ending indices for annealing are taken to be any random integer between the bounds set by η. A square region with the same starting and ending indices determined by η is then selected for adversarial manipulation. To generate appropriate candidate solutions, an additional condition is that the starting index for η is in the first half of the image and the selected square region has an area of at least 1 pixel. The final mask applied to generate candidate solutions is determined by the randomization resulting from η alongwith the nonzero pixel values found in positive and negative data. In Figure 3.4(f), the x-axis is varied by the difference in the upper bound and lower bound for η. From Figure 3.4(f) we conclude that a width of 4 is suitable for adversarial manipulation on the given training data. A maximum performance drop and gain of 20% and 50% is also observed in Figure 3.4(f). The annealing mask width parameter η is similar to the crossover width parameter η.

Annealing sample size The annealing sample size parameter ν determines the number of candidate solutions generated in each run of simulated annealing within each iteration of the game. ν takes a default value of 50. In Figure 3.4(g), ν is varied between 10 and 100. From Figure 3.4(g) we conclude that a sample size of 40 is suitable for adversarial manipulation. We also observe that there is a periodic trend in the manipulated performance across ν. A maximum performance drop and gain of 20% and 50% is also observed in Figure 3.4(g).

Annealing reduction rate The annealing sample size parameter ρ determines the rate of convergence of the simulated annealing runs within each iteration of the game. ρ takes a default value of 0.6. In Figure 3.4(h), ρ is varied between 0.5 and 0.9. A value of $\rho < 0.5$ ensures that the simulated annealing converges quickly whereas $\rho > 0.5$ ensures it converges slowly. From Figure 3.4(h) we conclude that a value of ρ between 0.5 and 0.6 seems best suited for adversarial manipulation. A maximum performance drop and gain of 25% and 60% is also observed in Figure 3.4(h).

3.6.4 Fitness function validation in Sequential Game

The adversarial manipulation α^* is found on convergence of the game. The game convergence criteria are subject to the fitness function Equation 3.5 used in the evolutionary algorithm. In this section, we report that by using an α^*, an adversary is able to affect the

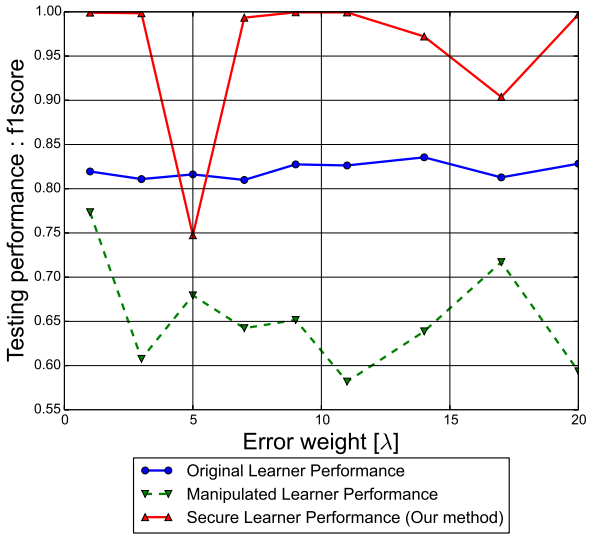


Figure 3.5: Testing performance with variation in error weight λ. The manipulated learner has lower performance than the original learner. The secure learner has higher performance than the manipulated learner.

testing performance of the learner across various parameter settings in the evolutionary algorithm and various combinations of positive and negative classes in the input MNIST database. t-statistics are calculated on the testing F1-score performance to check the effect of α^* across various classes and attack scenarios used in both the genetic algorithm and the simulated annealing algorithm.

For a converged α^* output by the game in either Algorithm 2 or Algorithm 7, F1-score measures the performance of our algorithm in terms of the testing data X_{test}, manipulated testing data $X_{test} + \alpha^*$ that are seen when the learner is trained on input data X_{train} and manipulated training data $X_{train} + \alpha^*$.

Fitness function error weight The error weight parameter λ determines the importance of the CNN error with respect to the cost of adversarial manipulation as simulated annealing generates candidate solutions within each iteration of the game. In Figure 3.5, λ is varied between 0.1 and 20. A value of $\lambda < 1$ ensures that simulated annealing gives less weight to CNN error rather than game cost to determine the best candidate for adversarial manipulation. A value of $\lambda > 1$ gives more weight to CNN error rather than game cost. A value of $\lambda = 1$ gives the same weight to CNN error and game cost. λ is empirically set for each labelled input. From Figure 3.5 we conclude that a value of $\lambda > 1$ is best suited for the current input. Specifically, λ between 10 and 15 seems best suited for the game. A maximum performance drop and gain of 20% and 40% is also observed in Figure 3.5. Table 3.2 and Table 3.3 shows the p-values for the

learner's F1-score before and after the game. Here t-statistic is an unpaired 2-sample t-test statistic. We conclude the following from the p-values on the performance of original learner $CNN_{original}$, manipulated learner $CNN_{manipulated-cnn}$ or $CNN_{manipulated}$ and secure learner CNN_{secure}.

- The manipulated learner $CNN_{manipulated}$ under attack has lower performance than the original learner $CNN_{original}$.

- The secure learner CNN_{secure} retrained from the game has higher performance than the manipulated learner $CNN_{manipulated}$ under attack.

- The secure learner CNN_{secure} has equal or higher performance than the original learner $CNN_{original}$.

- The low p-values in Table 3.2 and Table 3.3 allow us to reject the null hypothesis that the means of the performance measures are the same before and after adversarial data manipulations.

- From the low p-values (<0.05) of the t-statistic comparing original learner $CNN_{original}$ with manipulated learner $CNN_{manipulated}$, the t-statistic comparing manipulated learner $CNN_{manipulated}$ with secure learner CNN_{secure} and the Friedman test statistic in the paired test statistics of Table 3.2 we conclude that adversarial manipulations from the genetic algorithm affect learner performance with and without adversarial training data.

- From the low p-values (<0.05) of the t-statistic comparing original learner $CNN_{original}$ with manipulated learner $CNN_{manipulated}$, t-statistic comparing original learner $CNN_{original}$ with secure learner CNN_{secure}, the t-statistic comparing manipulated learner $CNN_{manipulated}$ with secure learner CNN_{secure} and the Friedman test statistic in paired test statistics of Table 3.3 we conclude that adversarial manipulations from simulated annealing algorithm not only affect learner performance with and without adversarial training data but also that the retrained learner has better performance and is comparable to the original learner.

- From the high p-values (>0.05) of t-statistic comparing original learner $CNN_{original}$ with secure learner CNN_{secure} in Table 3.2 and the low p-values (<0.05) of the t-statistic comparing original learner $CNN_{original}$ with secure learner CNN_{secure}

Genetic Parameter	Model performances t-statistics in Two-player games			
	CNN_o *vs* CNN_m	CNN_o *vs* CNN_s	CNN_m *vs* CNN_s	Friedman test
Upper bound for Mutation (δ)	8.0×10^{-16}	0.1395	2.5×10^{-5}	3.6×10^{-5}
Minimum width for Crossover (η)	0.0001	0.1446	0.0020	0.0098
Percentage offspring size for Selection (ζ)	4.6×10^{-7}	0.2393	0.0001	0.0111
Population size (ψ)	9.3×10^{-22}	0.00936	7.1×10^{-10}	5.7×10^{-06}

Table 3.2: Genetic Algorithm : p-values comparison before and after game by varying parameters for each genetic operator. t-statistics are computed between pairs of learner F1-score performance, manipulated learner F1-score performance and secure learner F1-score performance. CNN_o,CNN_m and CNN_s are short names for $CNN_{original}$,$CNN_{manipulated}$ and CNN_{secure} respectively.

Annealing Parameter	Model performances t-statistics in Two-player games			
	CNN_o *vs* CNN_m	CNN_o *vs* CNN_s	CNN_m *vs* CNN_s	Friedman test
Upper bound for Perturbation step (δ)	1.8×10^{-10}	2.1×10^{-6}	1.85×10^{-10}	4.6×10^{-5}
Upper bound for Perturbation index (η)	2.0×10^{-4}	4.6×10^{-6}	5.4×10^{-7}	3.4×10^{-4}
Reduction rate (ρ)	1.6×10^{-6}	1.0×10^{-2}	4.9×10^{-5}	1.5×10^{-2}
Sample size (ν)	6.8×10^{-11}	1.1×10^{-2}	1.37×10^{-6}	3.7×10^{-4}
Error weight (λ)	5.1×10^{-7}	2.0×10^{-4}	1.9×10^{-7}	3.0×10^{-4}

Table 3.3: Simulated Annealing Algorithm : p-values comparison before and after game by varying parameters for each annealing operator. t-statistics are computed between pairs of learner F1-score performance, manipulated learner F1-score performance and secure learner F1-score performance. CNN_o,CNN_m and CNN_s are short names for $CNN_{original}$,$CNN_{manipulated}$ and CNN_{secure} respectively.

in Table 3.3 we conclude that the learner with simulated annealing attack scenarios is more robust than the learner with genetic algorithm attack scenarios in the two-player two-label sequential game.

- From the low p-values across Table 3.2 and Table 3.3, we conclude that our game is not sensitive to the choice of an evolutionary algorithm. The statistical significance of our model generalizes across various parameter randomizations comparing the performances of original learner $CNN_{original}$, manipulated learner $CNN_{manipulated}$ and secure learner CNN_{secure}. In all the attack scenarios and two-label classification problems, we conclude that secure learner CNN_{secure} is robust to adversarial attacks proposed and simulated on the original learner $CNN_{original}$.

3.6.5 Evolutionary operations validation in Stochastic Game

In this section, we report the proposed model performances across various game types. Table 3.4 reports the performances for a sequential game and Table 3.5 reports the performances for a stochastic game. The sequential game is a special case of the stochastic game. The F1-scores in the tables are given for original learner $CNN_{original}$, manipulated learner $CNN_{manipulated}$ that is either a CNN based learner $CNN_{manipulated-cnn}$ or a GAN based learner $CNN_{manipulated-gan}$ on original and generated data respectively, secure learner CNN_{secure}. For the purposes of experimentation in a stochastic game, we assume five independent adversaries attack the learner. We also assume the original learner for baseline performance is a CNN model. The CNN is trained/tested on both the original data in the MNIST database and the generated data is obtained from an adversarial network. The lower digit in the class labels tuple of Table 3.4 and Table 3.5 is taken to be the positive label for adversarial manipulation. In the tables, we use the models defined in Section 3.2.

To obtain the generated data, we use four GANs, namely, Conditional DCGAN [194], IWGAN [99], BEGAN [26] and InfoGAN [54]. The data is generated for each pair of positive and negative labels. The CNN trained on the original MNIST data participates in the game whose output is the adversarial manipulation on each of the positive labels creating adversarial data. Then, a CNN trained on the generated data is validated against the adversarial data.

The stochastic game consists of multiple adversaries playing attack scenarios in terms of either genetic operators or annealing operators determining the training algorithm in the game. Following the formulation in Equation 3.8, by retraining the learner on all the adversarial manipulations, we demonstrate an effective learner robust to combined attacks from all the players. In Table 3.4 and Table 3.5, for both the original data and generated data, we observe that a learner tested on manipulated data $CNN_{manipulated-cnn}$,$CNN_{manipulated-gan}$ shows a significant decrease in performance compared to the learner $CNN_{original}$ trained and tested on original data.

By validating the adversarial data against the CNN trained on the output of various GANs, we observe that the effectiveness of the attack decreases with the increased robustness of the GANs. A more successful attack with larger error margins is possible when the CNN participating in the game is trained on original data as well as generated data. We also observe that GA attacks are better than SA attacks. Finally, multiplayer game attacks are better than two-player game attacks. After the game's convergence, the learner can be retrained on the manipulated data to find secure learner CNN_{secure}

F1-score: Genetic Operators in Two-player Game							F1-score: Annealing Operators in Two-player Game						Class Labels
CNN_o	CNN_m					CNN_s	CNN_o	CNN_m				CNN_s	
	CNN	*DCGAN*	*IWGAN*	*BEGAN*	*InfoGAN*			*CNN*	*DCGAN*	*IWGAN*	*BEGAN*	*InfoGAN*	
0.8696	0.7023	0.7881	0.6814	0.7737	0.8019	0.909	0.8464	0.692	0.6834	0.6548	0.7483	0.7759	0.9372	(2,8)
0.8193	0.6354	0.7502	0.7219	0.7487	0.8029	0.9186	0.9582	0.7094	0.763	0.7539	0.8957	0.9218	0.9346	(4,9)
0.9678	0.5288	0.6147	0.631	0.631	0.6311	0.8834	0.9816	0.5953	0.693	0.6319	0.8681	0.634	0.9983	(1,4)
0.8889	0.4881	0.7041	0.679	0.6787	0.6758	0.8141	0.9119	0.6395	0.6547	0.6751	0.6786	0.6757	0.9639	(5,8)
0.8971	0.5183	0.6812	0.6484	0.7887	0.7314	0.9226	0.8974	0.5575	0.6433	0.6479	0.8681	0.7316	0.9977	(3,8)
0.7995	0.6546	0.6592	0.7486	0.6919	0.7738	0.9987	0.8177	0.6181	0.7097	0.8011	0.651	0.6481	0.9991	(7,9)
0.9649	0.5813	0.761	0.7197	0.7995	0.7317	0.9555	0.96	0.6172	0.761	0.7595	0.7057	0.7946	0.83	(6,8)
0.9463	0.5702	0.8873	0.8879	0.8985	0.9299	0.9612	0.9609	0.611	0.8581	0.9057	0.9514	0.95	0.989	(2,6)
t-statistics	7.0×10^{-8}	1.2×10^{-4}	3.5×10^{-5}	3.1×10^{-4}	8.0×10^{-4}	Base	t-statistics	8.2×10^{-9}	3.8×10^{-6}	4.4×10^{-5}	3.0×10^{-3}	1.2×10^{-3}	Base	

Table 3.4: Two-player Two-Label Sequential Games Performance Evaluation - F1-score before and after two-player game across various combinations of handwritten digits. CNN_o,CNN_m and CNN_s are short names for $CNN_{original}$,$CNN_{manipulated}$ and CNN_{secure} respectively. We observe a consistent decrease in the manipulated learner CNN_m performance tested on adversarial data as compared to learner CNN_o performance. We also observe a consistent increase in the secure learner CNN_s performance compared to manipulated learner CNN_m.

weights that are secure to further adversarial manipulations on both original and generated data.

The last five rows in Table 3.4 and Table 3.5 report p-values for the t-statistics comparing secure learner performance with manipulated learner performance where the manipulated learner is trained on either the original data or the generated data. The attack scenarios for generating adversarial manipulations are determined by number and type of players participating in the game. The game is simulated over different combinations of the positive and negative class labels. The t-statistics also validate that our method is immune to various adversarial datasets. We demonstrate that the original CNN model as well as the GAN-based CNN model are vulnerable to proposed adversarial manipulations. The effectiveness of adversarial manipulations is demonstrated to generalize from the two-player game to the multiplayer game. More adversarial manipulations can be obtained by defining more adversarial scenarios comparing the original deep network model with the manipulated deep network model as well as the retrained deep network model. Better search algorithms and optimization conditions would lead to adversarial data manipulations that are effective on both original data distributions as well as generated data distributions.

F1-score: Genetic Operators in Multiplayer Game							F1-score: Annealing Operators in Multiplayer Game							Class Labels
CNN_o	CNN_m					CNN_s	CNN_o	CNN_m					CNN_s	
	CNN	*DCGAN*	*IWGAN*	*BEGAN*	*InfoGAN*			*CNN*	*DCGAN*	*IWGAN*	*BEGAN*	*InfoGAN*		
0.8651	0.6316	0.6748	0.707	0.7564	0.7717	0.8715	0.8466	0.6826	0.6572	0.6193	0.7166	0.7894	0.9474	(2,8)
0.8607	0.6027	0.7461	0.7466	0.7974	0.8154	0.829	0.8212	0.7123	0.7238	0.7387	0.775	0.8153	0.9023	(4,9)
0.9745	0.629	0.6034	0.6314	0.636	0.6316	0.833	0.9777	0.5938	0.6288	0.7005	0.7315	0.7332	0.8679	(1,4)
0.8294	0.6718	0.6825	0.6789	0.6871	0.69	0.8491	0.9096	0.5251	0.6452	0.6956	0.6879	0.6871	0.9998	(5,8)
0.8876	0.7455	0.6698	0.6526	0.7788	0.7315	0.9041	0.9007	0.6306	0.6689	0.6506	0.8283	0.7316	0.8583	(3,8)
0.8572	0.6836	0.6824	0.7898	0.7273	0.7799	0.9866	0.8484	0.7176	0.7136	0.7811	0.7905	0.8317	0.9885	(7,9)
0.9602	0.5827	0.7444	0.6992	0.7927	0.7183	0.8528	0.9523	0.5457	0.6824	0.7475	0.7318	0.7433	0.7697	(6,8)
0.925	0.6304	0.8649	0.8902	0.9131	0.91	0.936	0.9449	0.766	0.8332	0.9022	0.9161	0.9295	0.9364	(2,6)
t-statistics	4.5×10^{-7}	1.4×10^{-4}	5.4×10^{-4}	3.9×10^{-3}	3.2×10^{-3}	Base	t-statistics	1.6×10^{-5}	2.9×10^{-5}	6.2×10^{-4}	2.6×10^{-3}	5.2×10^{-3}	Base	

Table 3.5: Multiplayer Two-label Stochastic Games Performance Evaluation - F1-score before and after multiplayer game across various combinations of handwritten digits. CNN_o, CNN_m and CNN_s are short names for $CNN_{original}, CNN_{manipulated}$ and CNN_{secure} respectively. We observe a consistent decrease in the manipulated learner CNN_m performance tested on adversarial data as compared to learner CNN_o performance. We also observe a consistent increase in the secure learner CNN_s performance compared to manipulated learner CNN_m.

3.7. Findings Summary

In this chapter we demonstrated that deep learning classifiers for image recognition are vulnerable to data manipulations learnt by game-theoretic adversaries. Our adversaries engage the deep learning classifiers in sequential games and stochastic games. Such games measure cost of generating adversarial data that is statistically different from training data. We also demonstrate that the Nash equilibria in such games are found by solutions to optimization problems in an evolutionary computing framework. We have defined and validated the convergence behaviour of such evolutionary computing in terms of candidate solutions produced by genetic operators and annealing operators. Further we propose various baseline classification models to account for the effects of adversarial data in the training procedure of the image recognition classifiers. To validate statistical significance of our adversarial data manipulations, we conduct experiments on original data distribution as well as generated data distribution across two-label datasets.

In comparison to the existing adversarial training procedures in literature, our supervised learning models can adapt to continuous adversarial data manipulations. Further, to learn the changes to the training data distributions that can mislead the targeted classifiers, we focus on studying the attack (rather than defense) of the learning model. The proposed techniques are potentially useful for applications where data manipulation occurred and CNN is used as a classifier.

CHAPTER 4

GAME THEORETICAL ADVERSARIAL DEEP LEARNING WITH RANDOMIZATION STRATEGIES

In this chapter we discuss the problem formulation for the proposed adversarial learning algorithm with randomization strategies. A variational model is formulated for adversarial manipulations in two player sequential games.

4.1. The differences between adversarial data generated by randomization strategies and evolutionary strategies

The stochastic optimization problem and randomized strategy space for attacking multi-label classifiers with randomization strategies are different from those proposed in Chapter 3: (i) we formulate variable-sum Stackelberg games where adversarial cost function is defined on a attack strategy space of encoded data, while Chapter 3 proposed attack strategies are for a constant sum Stackelberg game in original data space; (ii) we propose a new simulated annealing algorithm to find optimal adversarial manipulations, attack parameters and payoff functions for multi-label classifiers while Chapter 3 proposed genetic and annealing operators attack two-label classifiers. (iii) By participating in a new Stackelberg game with the classifier, our game theoretical adversary repeatedly invokes annealing algorithm until repeatedly retrained classifier evaluates to

a decrease in adversarial payoff function over targeted class. By contrast, the annealing algorithm in Chapter 3 is invoked only once and converges to the optimal fitness function according to an exponential cooling schedule. (iv) During game we measure misclassification performance of the multi-label classifier in terms of true positive rate of targeted class converted into multiple negative classes. By contrast, Chapter 3 measured the misclassification performance of a two-label classifier in terms of recall of the positive class extended into recall of multiple positive classes in multiplayer game settings. (v) In comparision to adversarial manipulations found in Chapter 3 at Nash equilibrium, our game theoretical adversary finds new stochastic optima for attacking targetted classifier. In experimental validation of our attack scenarios we demonstrate that our adversarial attack successfully misleads multi-label classifiers trained on the two-label adversarial datasets produced in Chapter 3.

4.2. Game Formulation

In this section, we discuss problem formulation for two-player sequential Stackelberg game which is the foundation of our proposed adversarial learning algorithm.

4.2.1 Stackelberg game formulation

The key ingredients in a Stackelberg game [180] are (a) modelling players as the decision makers, (b) modelling actions (or series of actions) taken by players, and (c) modelling payoffs motivating players.

To model adversarial attack on CNNs, we assume an intelligent adversary as a leader player (L) interacting in a Stackelberg game with a follower player (F) – the CNN classifier. We then design data manipulations over a strategy space as the adversary's attack actions. Such data manipulations are performed on targetted classes. In response to each attack, the CNN is allowed to re-optimize weights on manipulated training data.

For misleading CNN's classification result, we design an objective function in a two-player sequential Stackelberg game based on an adversary payoff function and a classifier payoff function. The objective function is then optimized to produce adversarial data.

The leader L starts the game by making initial action (or move). The actions available to L and F are assumed to be over search spaces (or strategy spaces) A and W respectively. The outcome of an action is determined by L's and F's payoff functions $J_L \in \mathbb{R}$ and $J_F \in \mathbb{R}$,

respectively. Once CNN has been trained to learn optimal parameters w^* for all $w \in W$ on training data, the adversary's best action is formulated as α^* for all $\alpha \in A$ in a sequential game. The payoff function J_L for adversary is formulated as solving for α^* in Equation 4.1. The payoff function J_F for classifier is formulated as solving for w^* in Equation 4.2. In this research, A is taken to be data space of pixels in an image database and W is taken to be feature space of weights in a CNN classifier.

$$\alpha^* = argmax_{\alpha \in A} J_L(\alpha, w^*). \tag{4.1}$$

$$w^* = argmax_{w \in W} J_F(\alpha^*, w). \tag{4.2}$$

Equation 4.1 and Equation 4.2 can be combined into Equation 4.3 to form a sequential game with (α^*, w^*) expressed over adversarial manipulations α.

$$(\alpha^*, w^*) = argmax_{\alpha \in A} J_L(\alpha, argmax_{w \in W} J_F(\alpha, w)). \tag{4.3}$$

We assume the leader L's payoff is proportional to the follower F's payoff. The relation between L's and F's payoff functions is determined by a constant profit Φ for the game, variable profit $cost_L(\alpha)$ for the adversary and variable profit $cost_F(w)$ for the classifier. Further, the relative importance of adversary's cost $cost_L(\alpha)$ compared to classifier's cost $cost_F(w)$ in determining game solutions is controlled by a weighting parameter λ.

$$J_L + J_F = \Phi + \lambda * cost_L(\alpha) + cost_F(w). \tag{4.4}$$

To formulate a Stackelberg game using Equation 4.4, we express classifier's J_F in terms of adversary's J_L so that Equation 4.3 can be rewritten as a two-player game in Equation 4.5. By treating CNN as a blackbox model, we assume $cost_L(\alpha)$ is independent of $cost_F(w)$ in adversary's attack scenario. It is worth noting that the cost of re-optimizing a CNN, $cost_F(w)$, does not have a closed form expression for optimization. Then the overall objective of the game is:

$$\begin{aligned}(\alpha^*, w^*) = argmax_{\alpha \in A} J_L(\alpha, argmax_{w \in W}(\Phi + \lambda * cost_L(\alpha) \\ + cost_F(w) - J_L(\alpha, w))).\end{aligned} \tag{4.5}$$

In every game iteration, each player's move depends on the opponent's previous move. For a CNN learning optimal w^* for all $w \in W$ on training data X_{train}, the adversary generates adversarial manipulation α^* by searching over candidate solutions $\alpha \in A$ best suited for attacking w^* according to game model in Equation 4.5. The classifier is then allowed to re-optimize w^* on manipulated data $X_{train} + \alpha^*$ to defend against adversarial manipulation α^*. Game ends if the adversary's payoff J_L stops increasing with the

game iterations. At the end of the game, adversary converges onto optimal adversarial manipulation α^*.

Equation 4.5 is a bilevel stochastic optimization problem. We design adversary's payoff function J_L that can be solved in Equation 4.5 for optimal attack policy (α^*, w^*) where $cost_L(\alpha) = \|\alpha\|_F$.

$$J_L(\alpha, w) = error_F(w) - \lambda * cost_L(\alpha). \tag{4.6}$$

We define an attack scenario where adversary's payoff function J_L is given in Equation 4.6. In Equation 4.6, $error_F(w)$ is CNN's classification error for targetted classes and $cost_L(\alpha)$ is the adversary's manipulation cost for finding optimal solutions in Equation 4.5. Our intuition for the attack scenario with targetted classes is that an adversary aims to increase CNN's error $error_F(w)$ of misclassifying targetted classes while ensuring minimum change $cost_L(\alpha)$ to clean data. We measure $cost_L(\alpha)$ in terms of Frobenius norm $\|\alpha\|_F$ of manipulating tensor with same shape as encoded training data examples.

Since $error_F(w)$ is obtained from a blackbox CNN where we do not assume any knowledge of CNN's classification weights w, we empirically determine $cost_L(\alpha)$ across attack parameter settings in adversarial algorithm solving for α.

The $cost_L(\alpha)$ is enhanced by weighting term λ, which empirically evaluates relative importance of error $error_F(w)$ and cost $cost_L(\alpha)$ in adversarial attack scenarios. Setting low value to λ leads to low values for adversarial cost $\lambda * cost_L(\alpha)$ in comparison to (hopefully high) misclassification error $error_F(w)$. For training data X_{train}, λ allows us to control effect of $\|\alpha\|_F$ on adversarial data $X_{train} + \alpha$.

During game, we measure game model's performance $error_F(w)$ in terms of (percentage) true positive rate as $error_F(w) = (1 - tpr_{target}(w))$ for targetted class $target$. It is called adversarial attack performance in the model evaluations. After game convergence, $error_F(w)$ is measured in terms of (percentage) f_1-score as $error_F(w) = (1 - fscore_{overall}(w))$ for all classes $overall$ in manipulated data $X_{train} + \alpha^*$. It is called classification defence performance in the model evaluations.

We choose evaluation measure $tpr_{target}(w) = \frac{tp_{target}(w)}{p_{target}(w)}$ as attack performance metric since the adversary's payoff is optimized in data manipulations to positive class $target$. To attack a multi-label classifier, the adversary's goal is to generate illegitimate data for positive label $target$ that appears to be legitimate across many negative labels $overall$. At the same time, the classifier's goal of maintaining overall classification performance across all attacked labels is determined by $fscore_{overall}(w) = 2 \times \frac{tp_{target}(w)}{tp_{overall}(w) + fn_{overall}(w) + fp_{overall}(w)}$.

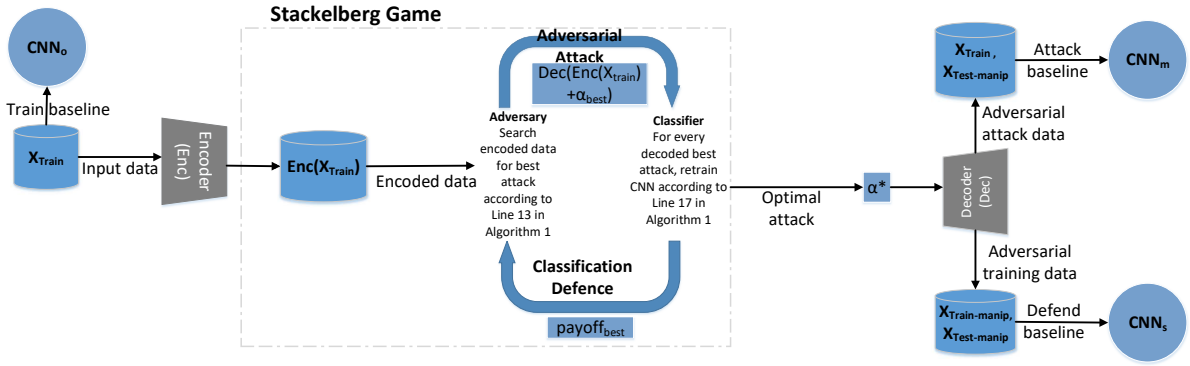


Figure 4.1: A flowchart illustrating the adversarial autoencoder based Stackelberg game-theoretic modelling.

To optimize adversarial manipulation $\alpha \in A$, we use a simulated annealing algorithm to solve the game model in Equation 4.5 with payoff function given in Equation 4.6. In an autoencoder network, adversarial manipulations α are generated on encoded data $Enc(X_{train})$, where Enc is the encoder function of an autoencoder network. Equation 4.6 is evaluated by decoding the perturbed data $Dec(Enc(X_{train}) + \alpha)$ that has been subject to adversarial manipulation α, where Dec is the decoder function of a autoencoder network.

4.2.2 Stackelberg Game Illustration

Figure 4.1 is a flowchart for our adversarial autoencoder based Stackelberg game model. A multi-label classifier $CNN_{original}$ (henceforth shortened as CNN_o) with weights $w^* \in W$. It is trained on labelled training data X_{train} and evaluated on labelled testing data X_{test} sourced from an image database. CNN_o participates in a two-player game with our game theoretical adversary. Adversary attacks CNN_o on a targetted positive label $target = pos$ by generating optimal attack $\alpha^* \in A$ at Nash equilibrium for every negative label $neg \in Neg$ that targetted positive label pos is manipulated into. In this research pos and Neg are class labels where $overall = pos \cup Neg$, and $A = Enc(X_{train})$ is determined by an autoencoder function Enc trained on X_{train}.

In each iteration of game, adversarial manipulation α_{best} is generated by the simulated annealing algorithm in Algorithm 10. For training data X_{train}, each α_{best} generates adversarial data $Enc(X_{train}) + \alpha_{best}$ in encoded space. It is then decoded as $Dec(Enc(X_{train}) + \alpha_{best})$ to be evaluated against CNN_o.

Upon convergence game outputs optimal α^* inferred for each pair of pos and neg. All α^*'s are then combined to effect a multi-label adversarial attack on CNN_o to output

Algorithm 9 Adversarial Autoencoder based Stackelberg Game

Input:
Labelled training data X_{train}, Labelled testing data X_{test} (for final evaluations only), Positive class label pos, Negative class labels Neg, Cost weighting term λ, Maximum step size $maxstep$, Upper bound for simulated annealing iteration count $maxiter$
Output:
Original performance $error_o$, Attack performance $error_m$, Defence performance $error_s$

Train CNN on X_{train} to output model CNN_o, $CNN_m = CNN_o$
Train Autoencoder on X_{train} to get Encoder function Enc and Decoder function Dec
for $neg \in Neg$ **do**
    $mask = mean(Enc(X_{train}[neg])) - mean(Enc(X_{train}[pos]))$
    exitgame = False, $payoff_{prev} = payoff_{best} = 0, \alpha^*[neg] = 0$
    while ¬ exitgame **do**
        Randomly generate α, a tensor with values in [0,$maxstep$] and $mask$ size and shape
        $\alpha = \alpha \odot mask$
        Evaluate CNN on $Dec(Enc(X_{train}[pos]) + \alpha) \cup X_{train}[neg]$ to find error $error_{prev}$
        $payoff_{prev} = error_{prev} - \lambda * \|\alpha\|_F$
        Calculate α_{best} and $payoff_{best}$ from simulated annealing sa in Algorithm 10
        If $payoff_{best} - payoff_{prev} > 0$ **then**
            exitgame = False
            $\alpha^*[neg] = \alpha_{best}$
            Retrain CNN on $Dec(Enc(X_{train}[pos])) + \alpha_{best}) \cup X_{train}[neg])$
        else
            exitgame = True
        $payoff_{prev} = payoff_{best}$
    end while
$X_{train-manip} = X_{train}$, $X_{test-manip} = X_{test}$
for $neg \in Neg$ **do**
    $X_{train-manip} = X_{train-manip} \cup Dec(Enc(X_{train}[pos]) + \alpha^*[neg]) \cup X_{train}[neg]$
    $X_{test-manip} = X_{test-manip} \cup Dec(Enc(X_{test}[pos]) + \alpha^*[neg]) \cup X_{test}[neg]$
Evaluate CNN_o on X_{test} to find error $error_o$
Evaluate CNN_m on $X_{test-manip}$ to find error $error_m$
Train CNN_m on $X_{train-manip}$ to output model CNN_s
Evaluate CNN_s on $X_{test-manip}$ to find error $error_s$
return ($error_o$,$error_m$,$error_s$)

manipulated classifier $CNN_{manipulated}$ (henceforth shortened as CNN_m). CNN_m is finally retrained into secure classifier CNN_{secure} (henceforth shortened as CNN_s) that is robust to multi-label adversarial attacks.

4.3. Our Proposed Algorithms

In this section, we present the algorithms for a two-player Stackelberg game defined by Equation 4.5.

Algorithm 9 is the training algorithm for Stackelberg game that solves for adversarial manipulations α^* in Equation 4.5. As input, Algorithm 9 requires labelled training data X_{train}, labelled testing data X_{test}, target positive class pos for adversarial attack and negative classes Neg to mislead a CNN classifying pos. A key difference between adversarial examples in literature and those produced in every move of our game's iteration is that Algorithm 9 re-optimizes CNN's optimal weights w^* such that adversarial attacks α^* increase adversary's payoffs J_L across game moves. Moreover, various blackbox attack scenarios on CNN are controlled by λ weighting term on adversarial cost in Equation 4.6.

Algorithm 10 presents the simulated annealing method, which is used to determine the best adversarial manipulation α_{best} in each game iteration.

4.3.1 Adversarial Learning Algorithm

Algorithm 9 initializes the game on line 3 by training CNN to get CNN_o (with optimal training weights w^*), and line 4 trains Autoencoder on X_{train} to get functions Enc and Dec. Adversary is assumed to target positive class pos to mislead classifier into misclassifying pos data into any negative class $neg \in Neg$. Two-player Stackelberg game loop from line 5 to line 20 creates adversarial data for attacking CNN. Between line 9 and line 11, we create adversarial data by adding random manipulation α to encoded positive data $Enc(X_{train}[pos])$.

On line 13, we use a simulated annealing algorithm described in Algorithm 10 to create candidate manipulation α_{best}. Between line 12 and line 20, we propose a game model's iteration to repeatedly attack and re-optimize CNN models weights w on manipulated training data $Enc(X_{train}[pos]) + \alpha_{best}$. Labelled testing data X_{test} is not used to create adversarial manipulations.

By end of game in line 20, we have created adversarial manipulations α^*, for each combination of positive label pos and negative label neg, that are in turn used to create manipulated training data $X_{train-manip}$ and manipulated testing data $X_{test-manip}$ between line 21 and line 24. For a given pos class label, on line 25 and line 26 we evaluate CNN_o on original testing data X_{test} and manipulated testing data $X_{test-manip}$ to output errors $error_{original}$, $error_{manipulated}$ (shortened as $error_o$, $error_m$) respectively. A successful adversary misleads classifier into misclassifying pos as $neg \in Neg$ when $error_m$ is greater than $error_o$.

4.3.2 Adversarial Manipulations Generation

For each combination of pos and neg data, we create an image mask $mask$ on line 6 from mean images. Changes to pixel values in α are limited by both pixel magnitudes and pixel positions in $mask$. With $mask$ we would like to find optimal change to positive class images such that they are misclassified as negative class images by current CNN model. In line 9, we generate a random adversarial manipulation α with pixel values in range $[0, maxstep]$ where $maxstep$ is a input parameter. On line 10 $mask$ is applied on α with an element-wise multiplication. Thus $mask$ and α are subject to same limits on pixel values.

Algorithm 10 Simulated Annealing with Random Restart

1: **function** SA($maxstep, mask, error_{prev}, payoff_{prev}, \alpha, maxiter, X_{train}, Enc, Dec$)
2: exitsearch = False, iter = 0, $payoff_{best} = error_{best} = 0$, jumpbound = 10
3: Initialize zeros tensor α_{best} with same shape as α
4: **while** ¬ exitsearch ∧ iter<maxiter **do**
5: iter = iter + 1
6: Randomly generate manipulation increment δ, a tensor with with values in [0,$maxstep$]
7: $\delta = \delta \odot mask$
8: $\alpha = \alpha + \delta$
9: Evaluate CNN on $Dec(Enc(X_{train}[pos]) + \alpha) \cup X_{train}[neg]$ to find error $error_{curr}$
10: $payoff_{curr} = error_{curr} - \lambda * \|\alpha\|_F$
11: **If** $payoff_{curr} > payoff_{best}$ **then**
12: $payoff_{best} = payoff_{curr}$
13: $\alpha_{best} = \alpha$
14: $error_{best} = error_{curr}$
15: **If** $payoff_{curr} - payoff_{prev} > 0$ **then**
16: exitsearch = False
17: **else**
18: **If** $error_{curr} - error_{prev} \leq 0$ **then**
19: Randomly select $jump$ in [0,jumpbound]
20:
21: **If** $|error_{curr} - error_{prev}| > jump$ **then**
22: exitsearch = False
23: **else**
24: exitsearch = True
25: $payoff_{prev} = payoff_{curr}$
26: **end while**
27: return $(payoff_{best}, \alpha_{best})$
28: **end function**

When α is available in current iteration of game, it is then used to initialize candidate manipulation. line 13 has simulated annealing algorithm sa from Algorithm 10 searching for best manipulation α_{best} and corresponding payoff $payoff_{best}$.

4.3.3 Simulated Annealing Algorithm

The simulated annealing algorithm sa in Algorithm 10 generates various candidate manipulatations α on encoded positive data $Enc(X_{train}[pos])$. It targets to output the best changed α called α_{best}. From function arguments, sa initializes candidate manipulation α given by current iteration of game. Finally, sa returns adversary's payoff $payoff_{best}$ and classifier's error $error_{best}$ corresponding to α_{best}.

Using $mask$ and $payoff_{prev}$ from line 4 to line 25 in Algorithm 10, α is optimized by randomly generated step increments δ. δ has least change to each encoded pixel in positive data such that adversarial manipulation α is able to mislead CNN. On line 7 $mask$ is applied on δ to limit changes to α pixels by position and magnitude of $mask$ pixels. For every update to α on line 8, we apply adversarial manipulation on line 9 (on encoded data) and line 10 to return payoff $payoff_{curr}$ and error $error_{curr}$ evaluating CNN. $error_{curr}$ is true positive rate $tpr_{target}(w)$ of target class pos under attack.

The looping condition from line 11 to line 14 ensures that incremental change to α continues as long as payoff $payoff_{curr}$ is increasing. The looping condition from line

15 to line 24 determines search behaviour when $payoff_{curr}$ starts decreasing. Line 15 is a deterministic condition to stop annealing search. Line 21 is a randomization condition to restart annealing search. Game parameters λ, $maxstep$, $maxiter$ determine convergence rate of the annealing algorithm in terms of the rate of change of $payoff_{best}$ across game iterations.

4.4. Experiments

In this section we discuss experimental validation and parameter settings for proposed adversarial learning algorithm. In a two-player Stackelberg game, various parameter settings correspond to various adversarial manipulations α^* that mislead CNN_o. Moreover we validate performance of various multi-label classifications produced by CNN models CNN_m trained on the MNIST database and tested on our data manipulations. We also compare defence performance of CNN models CNN_s retrained on data produced by our adversarial manipulations.

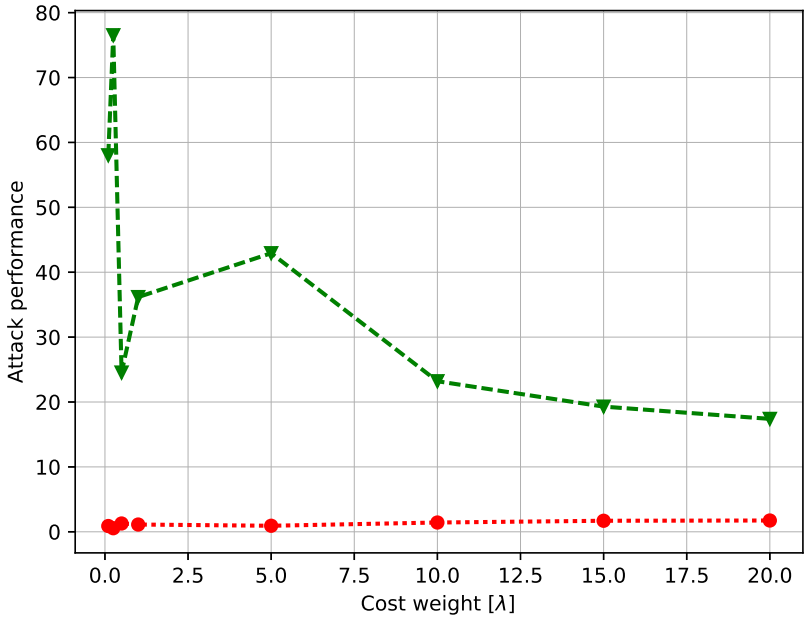
4.4.1 Classifier and Autoencoder Description

Pytorch[1] is used to create a CNN model that has deep representations of images using two convolution, one dropout and two fully connected layers. CNN has a rectified linear unit activation function. CNN has a log-softmax layer as loss function to make classification predictions. CNN is trained for 25 epochs on grayscale MNIST images to provide the baseline classification performance in our experiments. CNN's true positive rate for targeted class label is assumed to be adversary's target for Stackelberg game. We don't further fine tune the CNN learning processes since we assume a blackbox attack scenario where the adversarial cost function computation is independent of the classification cost function computation. Independently, we also use Pytorch to train a autoencoder model stacking two fully connected layers on MNIST images' encoder. Latent space of autoencoder is determined by parameter ρ – the code size of final layer in encoder.

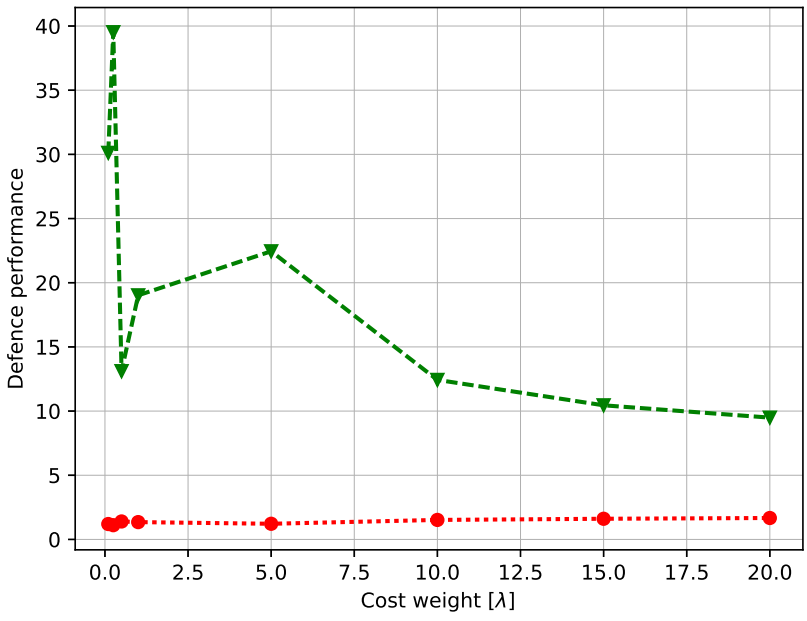
4.4.2 Attack and Defence Performance Validation with Fixed Target and Original Data

Given a targeted class label $target = pos$, adversary optimizes misclassification error (a.k.a., attack performance) expressed in terms of true positive rate $tpr_{target}(w)$ while

[1] `https://pytorch.org/docs/stable/index.html`



(a) Attack: varying λ



(b) Defence: varying λ

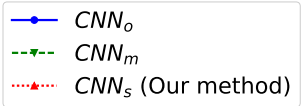
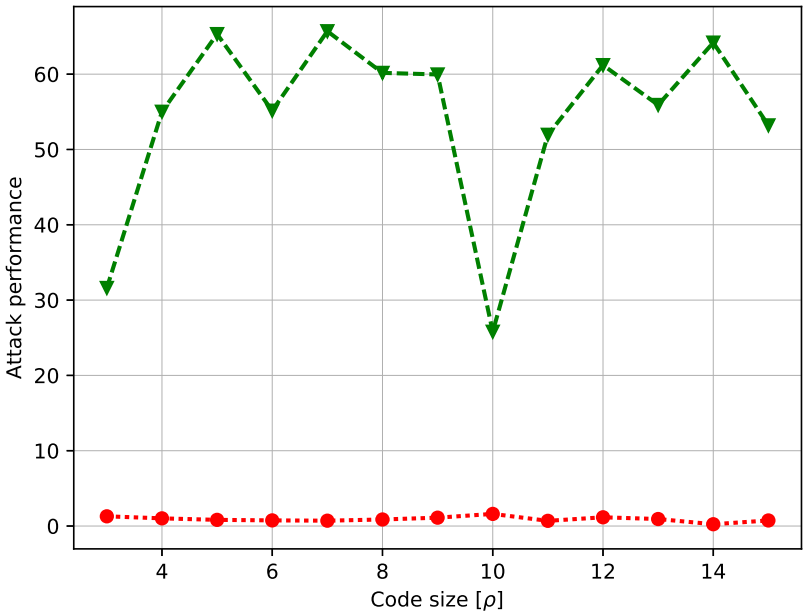
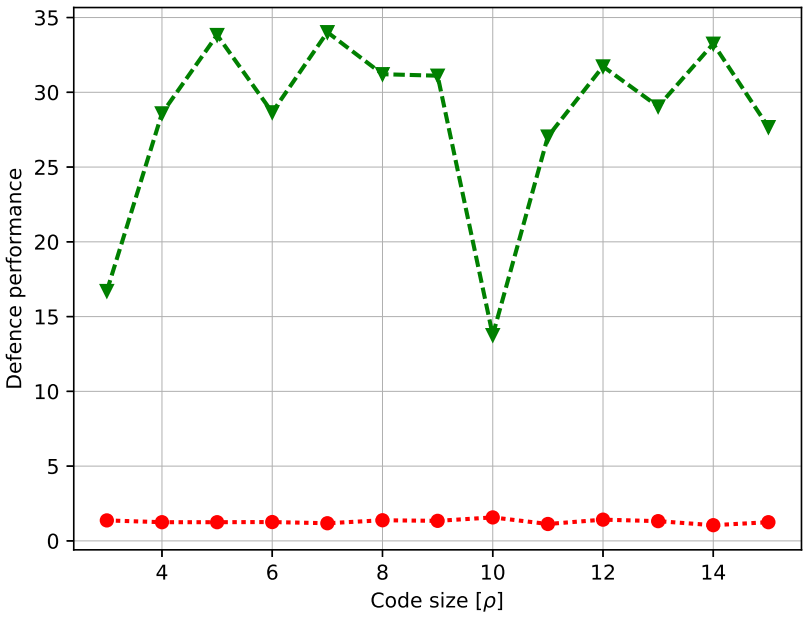


Figure 4.2: Testing performance (error) with variations in attack operators consisting of adversarial cost weight λ and autoencoder code size ρ.



(c) Attack: varying ρ



(d) Defence: varying ρ

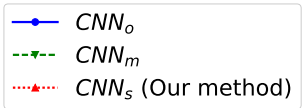


Figure 4.2: Testing performance (error) with variations in attack operators consisting of adversarial cost weight λ and autoencoder code size ρ.

Table 4.1: Autoencoder Attack Scenario: t-tests before and after game by varying parameters for target class "7".

	p-values from t-statistics in Stackelberg games		
Attack Parameter	CNN_o *vs* CNN_m	CNN_o *vs* CNN_s	CNN_m *vs* CNN_s
Cost weight (λ)	2.3×10^{-4}	3.3×10^{-2}	2.5×10^{-4}
Code size (ρ)	5.0×10^{-14}	6.9×10^{-6}	4.0×10^{-14}

classifier optimizes classification error (a.k.a., defence performance) expressed in terms of f_1-score $fscore_{overall}(w)$ for all class labels $overall = pos \cup Neg$. Figure 4.2(a) and Figure 4.2(c) compare CNN attack performance after adversarial manipulation with an example $pos = 7$. Figure 4.2(b) and Figure 4.2(d) compare CNN defence performance after adversarial training. The x-axis in Figure 4.2 gives parameter settings for adversarial attacks. The y-axis gives percentage error after adversarial attack. For attack parameters $\rho = 3$, $maxstep = 1/10$ and $maxiter = 100$ in Algorithm 10, the weighting term λ in Equation 4.6 is varied between 0.1 and 20. For same attack parameters, fixing $\lambda = 1$, Autoencoder's code size ρ is varied between 3 and 15.

In all outputs shown in Figure 4.2 we find that manipulated attack/defence performance $error_m$ of CNN_m is always greater than adversarially trained attack/defence performance $error_s$ of CNN_s. Same attack trend seen across CNN's attack performance and defence performance suggests that targeted positive class label $pos = 7$ most influences true positives $tp_{target}(w)$ numerator term in calculation of attack's true positive rate $tpr_{target}(w)$ and defence's f_1-score $fscore_{overall}(w)$.

Before the adversarial manipulation, the original performance of CNN_o, $error_o$, was found to be 0.86%. Table 4.1 compares p-values from pairwise t-tests comparing attack performances. Low p-values (<0.05) allow us to reject null hypothesis that the attack performances are same before and after adversarial manipulations. Thus adversarial manipulations in our model are statistically significant across various parameter randomizations in comparing attack performances of original classifier CNN_o, manipulated classifier CNN_m and secure classifier CNN_s.

4.4.3 Defence Performance Validation with Varying Target and Generated Data

Table 4.2 gives classifier's defence performance as classification error expressed in terms of f_1-score $fscore_{overall}(w)$. In last column of Table 4.2, we change pos across various target labels by fixing attack parameters $\lambda = 0.25$, $\rho = 8$, $maxstep = 1/50$.

The defence performance of models CNN_o, CNN_m and CNN_s is calculated in terms

Table 4.2: Comparisons on the defence to adversarial Nash equilibrium attacks

Classification error: Autoencoders attack in Stackelberg Game									Target Class
CNN_o	CNN_m							CNN_s (Our method)	
	CNN [133]	$DCGAN$ [194]	$IWGAN$ [99]	$DeepFool$ [173]	$FGSM$ [91]	CNN_{GA} [57]	CNN_{SA} [57]		
1.16	27.71	28.63	27.38	24.71	27.41	3.91	2.12	0.77	0
0.52	39.24	33.74	40.11	31.21	37.16	6.38	2.98	0.42	1
1.59	22.89	22.53	21.68	26.61	21.11	5.62	5.31	1.01	2
1.43	17.56	26.61	20.06	26.99	15.01	5.08	3.73	0.53	3
0.91	46.47	43.54	35.46	36.44	33.02	3.81	16.0	0.92	4
1.45	38.81	40.51	43.03	32.33	33.24	51.35	32.78	0.91	5
1.04	28.12	31.55	35.02	27.79	21.81	9.03	7.09	0.86	6
1.98	32.24	39.18	36.78	33.75	29.39	22.49	9.73	1.05	7
1.34	24.82	33.77	26.18	28.52	15.54	4.08	4.17	1.16	8
2.00	26.16	43.31	26.78	25.64	21.18	30.71	10.92	1.51	9
t-statistics	3.5×10^{-9}	1.8×10^{-11}	3.9×10^{-10}	6.7×10^{-15}	7.6×10^{-9}	1.6×10^{-2}	8.9×10^{-3}	Base	

of f_1-score $fscore_{overall}(w)$ as $error_o$, $error_m$ and $error_s$ respectively. We also compare our results against CNN models trained on adversarial examples produced by deep generative models misleading deep network classifiers. Specifically, we compare our CNN_m defence error after attacking CNN_o retrained on deep convolutional generative adversarial network (DCGAN) [194], improved Wasserstein generative adversarial network (IWGAN) [99], DeepFool [173], fast gradient sign method (FGSM) [91], genetic algorithm CNN_{GA} [57], annealing algorithm CNN_{SA} [57] outputs.

From Table 4.2, we observe that CNN_m after adversarial attack has higher defence error $error_m$ compared to original error $error_o$ of CNN_o before attack. Owing to our multi-step adversarial training, CNN_s achieves lower defence error $error_s$ than CNN_m defence error $error_m$. After game's convergence, classifier can be retrained on adversarial data to find secure model CNN_s that consistently has lower defence error $error_s$ than corresponding CNN_m defence error $error_m$ across varying target labels of both original data and generated data.

The low p-values (<0.05) in Table 4.2 allow us to reject null hypothesis that the defence performances are the same before and after adversarial trainings. The low p-values also show that our game-theoretic attack settings are statistically significant across CNN models trained on datasets output by deep learning models, deep generative models and game theoretical adversarial learning models in literature.

4.5. Findings Summary

In this chapter we demonstrate new Stackelberg games and Nash equilibria to create adversarial data for new attack scenarios on multilabel classifiers. Our game-theoretic adversary employs new randomization strategies for search and optimization of the

adversarial payoff function in the annealing algorithm. The Nash equilibrium is reached when adversary's payoff function cannot be further improved in the Stackelberg game.

For every candidate solution found by annealing algorithm over a fixed classification function, we further calculate the changes to adversarial cost with respect to adversarially trained classification function. Here we express the stopping condition for both annealing algorithm and Stackelberg game in terms of the changes to the adversary's payoff in participating in the game iterations.

The changes adversarial payoff is calculated on encoded data because we want to study changes to original data distribution in original dataset, encoded dataset and decoded dataset without assuming any knowledge about the target CNN's architecture. We are able to create a encoded representation for the adversarial data distribution across various modelling baselines for empirical validation of the misclassifications. Using an encoded space is a strategy where we can generate new synthetic samples (that never appeared in the training data), rather than make changes to data already existing in training, so that we can have the extra diversity of manipulated samples.

CHAPTER 5

GAME THEORETICAL ADVERSARIAL DEEP LEARNING WITH VARIATIONAL ADVERSARIES

In this chapter, we propose adversarial manipulations created by a variational adversary operation in a generative strategy space. In this research, we assume the adversary is the leader and the CNN follows by retraining the classifier. While participating in a Stackelberg game with our variational adversary, the classifier is assumed to behave like a blackbox model. The adversarial cost function is then optimized in a stochastic optimization manner by solving a variable-sum game. The adversarial manipulations are iteratively searched and optimized in the game until the adversarial payoff function starts decreasing for targeted class labels. Thus, we create two-label manipulations for a multi-label CNN classifier. At Nash equilibrium, the adversarial examples are able to attack the targeted classifier models that are trained on both original data distribution and generated data distributions. The adversarial manipulations found at our new stochastic optima are also able to mislead multi-label deep learning classifiers, which are trained on two-label adversarial data produced by existing (constant sum two-player) Stackelberg game models.

5.1. Game Formulation

In this section, we formulate a sequential variable-sum two-player Stackelberg game to mislead classifiers. Given training data distribution, our adversarial attack scenario

seeks to find data manipulations that generate changed data distributions misleading the classifier in the game theoretical model. Effective attack scenario is measured in terms of a classifier's misclassification error and an adversary's manipulation cost. Manipulation cost measures adversary's expenses to generate data manipulations in each game iteration. After multiple game iterations, these adversarial manipulations are optimized to find a Nash equilibrium between the converged adversary and classifier.

We formulate the adversarial payoff function on a convex strategy space determined by a Variational Autoencoder (VAE) model. The VAE model encodes training data and validation data into unsupervised mean and variance codecs. These codecs represent multivariate Gaussian distributions that are derived from data distributions in a latent space. We assume this latent space is the strategy space of our game. Please note that the VAE is part of our modelling and is not part of the classification model that will be under attack. At Nash equilibrium, we produce an output which is a vector of adversarial manipulations corresponding to VAE's encodings of training data. We assume a Convolutional Neural Network (CNN) to be the classifier of multi-label image databases, and a Simulated Annealing (SA) algorithm is utilized in doing stochastic search and optimization. To simulate a blackbox attack, we assume that the adversary has no prior knowledge of the classifier's training process. CNN is adversarially retrained according to an alternating least squares (ALS) algorithm until the game convergence to the Nash equilibrium. In every game iteration, SA computes convex adversarial data manipulations as local optima to ALS, which in-turn finds local optima to non-convex game theoretical model for characterizing our attack scenario.

In our Stackelberg game, we model the adversary acting as a leader player (L) attacking targeted classifier model. The classifier responds to adversarial manipulation by acting as a follower player (F), defending its classification performance by adversarial retraining. The attack-defence game plays between L and F proceeds according to a sequence of increasing payoff function values. Such a sequential Stackelberg game can converge to Nash equilibrium with optimal payoff function values for both players. The optimal payoff brings an optimal data manipulation for the adversary.

All the potential manipulations/moves of L are assumed to be over a strategy space A^M, A^Σ, where M and Σ are mean and variance parameter spaces determined by a VAE model. The moves of F are assumed to be over classifier's parameter space W, which is learnt on training data distribution. During each game iteration, the move of L and F is determined by their real-valued payoff functions $J_L \in R$ and $J_F \in R$, respectively. To ease understanding, the variables notation used in our problems and algorithms are

Table 5.1: Table of Notation

Variable Name	Variable Description	Variable Name	Variable Description	Variable Name	Variable Description
L	Adversary as Leader in Stackelberg Game	$\mathbb{A}_*^\mu$	Multi-label adversarial manipulations on data mean parameters	$cost_F$	F's classification cost function
F	Classifier as Follower in Stackelberg Game	$\mathbb{A}_*^\sigma$	Multi-label adversarial manipulations on data variance parameters	α^μ	Adversarial manipulation over means in L's strategy space
M	training data mean parameter space	μ_{pos}, σ_{pos}	pos training data's mean and variance parameters	α_*^μ	L's best move for manipulating data mean parameters
Σ	training data variance parameter space	μ_{neg}, σ_{neg}	neg training data's mean and variance parameters	α_*^σ	L's best move for manipulating data variance parameters
A^M, A^Σ	Leader L's strategy space	δ_μ	Maximum possible difference between mean parameters of negative and positive training data	α^σ	Adversarial manipulation over variances in L's strategy space
W	F's training parameter space	δ_σ	Maximum possible difference between variance parameters of negative and positive training data	α_{next}	Candidate adversarial manipulation generated in SA iteration
$payoff_{next}$	L's payoff function value when data is manipulated for α_{next}	$payoff_{curr}$	L's payoff function value in current iteration	$\alpha_{initial}$	Initial adversarial manipulated in SA
J_L	L's adversarial payoff function	$error_{curr}$	F's misclassification error value in current iteration	$\alpha_{optimal}$	Optimal adversarial manipulated returned by SA
$cost_L$	L's adversarial cost function	$payoff_{best}$	L's best move payoff function value	α_{best}	Best adversarial manipulation element found by SA across mean and variance parameters
J_F	F's classification payoff function	$error_{best}$	F's best move misclassification error value	$error_{next}$	F's misclassification error value when data is manipulated for α_{next}

summarized in Table 5.1.

In every game play, the value of J_L and J_F depends on candidate adversarial manipulations $\alpha^\mu \in A^M, \alpha^\sigma \in A^\Sigma$ and candidate classifier weights $w \in W$. Adversary's best move

finds a best strategy $\alpha^{\mu}_{*}, \alpha^{\sigma}_{*}$ with best payoff J_L as expressed in Eq. (5.1).

$$\alpha^{\mu}_{*}, \alpha^{\sigma}_{*} = argmax_{(\alpha^{\mu} \in A^{M}, \alpha^{\sigma} \in A^{\Sigma})} J_L((\alpha^{\mu}, \alpha^{\sigma}), w). \tag{5.1}$$

In response to adversarial manipulation, the best move of retrained classifier then converges onto a robust $w^* \in W$ with best payoff J_F as expressed in Eq. (5.2).

$$w^* = argmax_{w \in W} J_F((\alpha^{\mu}, \alpha^{\sigma}), w). \tag{5.2}$$

Substituting Eq. (5.2) in Eq. (5.1) we have Eq. (5.3) in terms of the adversarial payoff function $J_L((\alpha^{\mu}, \alpha^{\sigma}), w)$ and classification payoff function $J_F((\alpha^{\mu}, \alpha^{\sigma}), w)$.

$$\begin{aligned}(\alpha^{\mu}_{*}, \alpha^{\sigma}_{*}, w^*) = & argmax_{(\alpha^{\mu} \in A^{M}, \alpha^{\sigma} \in A^{\Sigma})} J_L\big((\alpha^{\mu}, \alpha^{\sigma}), \\ & argmax_{w \in W} J_F((\alpha^{\mu}, \alpha^{\sigma}), w) \big).\end{aligned} \tag{5.3}$$

Eq. (5.3) defines the Nash equlibrium in terms of final adversarial manipulations $\alpha^{\mu}_{*}, \alpha^{\sigma}_{*}$ and final classifier weights w^* that will be found after all the plays of the Stackelberg game.

In our blackbox attack scenario, we assume CNN misclassifying a set of positive labels Pos into another set of negative labels Neg. For $pos \in Pos$ and $neg \in Neg$, we map every label pair (pos, neg) to a sequential two-player Stackelberg game, i.e., solving Eq. (5.3) for multi-label adversarial manipulations $\mathbb{A}^{\mu}_{*}[pos, neg] = \alpha^{\mu}_{*}, \mathbb{A}^{\sigma}_{*}[pos, neg] = \alpha^{\sigma}_{*}$.

To solve Eq. (5.3), we assume that the movements between adversary and classifier interact in a sequence of game plays defined by a variable-sum Stackelberg game in Eq. (5.4)

$$J_L + J_F = \Phi + \lambda \times cost_L(\alpha^{\mu}, \alpha^{\sigma}) + cost_F(w). \tag{5.4}$$

where $\Phi \in R$ is a proportionality constant to increase adversary's payoff at the expense of classifier's payoff J_F. $cost_L(\alpha^{\mu}, \alpha^{\sigma})$ is the adversarial cost function, which typically has a closed form expression for game theoretical optimization and depends on application-specific adversarial data distributions. $\lambda \in R$ is a constant parameter to balance the weight between $cost_L$ and $cost_F$. $cost_F(w)$ is the classifier's cost for doing adversarial retraining across multiple game plays. Typical $cost_F(w)$ has no closed form expression for game theoretical optimization. In our research, $cost_F(w)$ replies on CNN's retraining performance and optimization strategies, such as, retraining runtime, number of adversaries, player's attack and defence budgets, training data size and classifier weights initialization. The interplay between adversary's strategy spaces and classifier's decision boundaries empirically determines the effect of $cost_L$ on $cost_F$. In blackbox attack

scenario, we assume that the adversary's cost function $cost_L$ is independent from the classifier weights w. Similarly, the classifier's cost function $cost_F$ is independent from the adversarial manipulations $\alpha^\mu, \alpha^\sigma$.

Substituting J_F from Eq. (5.4) into Eq. (5.3) defines bilevel non-convex objective function for variational Stackelberg game:

$$\begin{aligned}(\alpha^\mu_*, \alpha^\sigma_*, w^*) = argmax_{(\alpha^\mu \in A^M, \alpha^\sigma \in A^\Sigma)} & J_L\big((\alpha^\mu, \alpha^\sigma), \\ & argmax_{w \in W}(\Phi + \lambda \times cost_L(\alpha^\mu, \alpha^\sigma) \\ & + cost_F(w) - J_L((\alpha^\mu, \alpha^\sigma), w))\big).\end{aligned} \tag{5.5}$$

Eq. (5.5) characterizes our blackbox attack scenario. The objective function is computed in terms of the adversarial payoff function J_L, which determines the candidate adversarial manipulations $\alpha^\mu, \alpha^\sigma$. J_L converges to Nash equilibrium in a two-player multi-label sequential Stackelberg variable-sum game. Nash equilibrium outputs game theoretical adversarial manipulations $(\alpha^\mu_*, \alpha^\sigma_*)$ and the corresponding secure classifier weights w^*.

To solve Eq. (5.5), we design adversarial payoff function J_L in Eq. (5.6) with $cost_L(\alpha^\mu, \alpha^\sigma) = (\|\alpha^\mu\|_F + \|\alpha^\sigma\|_F)$.

$$J_L((\alpha^\mu, \alpha^\sigma), w) = error_F(w) - \lambda * cost_L(\alpha^\mu, \alpha^\sigma). \tag{5.6}$$

The strategy space A^M, A^Σ for $(\alpha^\mu, \alpha^\sigma)$ is a randomized convex data space determined by a VAE, i.e., a multivariate Gaussian mixture model. J_L in Eq. (5.6) is optimized by a SA algorithm that does stochastic search and local optimization for $\alpha^\mu_*, \alpha^\sigma_*$ in every game play. $error_F(w)$ denotes CNN's misclassification performance for positive labels $pos \in Pos$ to be manipulated into negative labels $neg \in Neg$. λ controls relative importance of adversarial cost function $cost_L$ compared to misclassification error $error_F(w)$.

During game model training evaluations, we define $error_F(w)$ in terms of true positive rate: $error_F(w) = error_{pos}(w) = (1 - tpr_{pos}(w))$ and call it attack performance. During game model testing evaluations, we define $error_F(w)$ in terms of (percentage) classification f_1-score for Pos as $error_F(w) = error_{Pos}(w) = (1 - fscore_{Pos}(w))$ and call it defence performance. Here $tpr_{pos}(w) = \frac{tp_{pos}(w)}{p_{pos}(w)}$ and $fscore_{Pos}(w) = 2 \times \frac{tp_{Pos}(w)}{tp_{Pos \cup Neg}(w) + fn_{Pos \cup Neg}(w) + fp_{Pos \cup Neg}(w)}$. To simplify the experimental setup we have assumed $Pos = pos$ and $Neg = neg$ for evaluating defence performances in Section **??**. However the CNN classifier is trained on all possible class labels $(pos, neg) \in (Pos \times Neg)$.

Eq. (5.6) is sub-problem of Eq. (5.5) which is the objective function for game theoretical optimizations. Eq. (5.6) is solved repeatedly by the adversary until convergence of the game iterations. In every game iteration, the changes to attack parameters $(\alpha^\mu, \alpha^\sigma)$

adversarially influence the updates to classification parameters w. Such a game's convergence condition is called the Nash equilibrium. It is a balance of maximum increases to payoff functions values J_L and J_F solving Eq. (5.5) until convergence.

Across the Stackelberg game iterations, we design the SA to optimize the $cost_L$ such that the leader L gains in changes to its payoff J_L is in proportion to the payoff changes lost by the follower F in changes to its payoff J_F. In Eq. (5.4), we also allow the follower F to assume an independent blackbox cost $cost_F$ of retraining to engage the leader L in a Stackelberg game. In Eq. (5.6), the pure profit for the adversary (acting as leader L) is determined by $cost_L$. While trying to minimize $cost_L$ the adversary's target is to maximize the misclassification error $error_F$ of a CNN (acting as follower F). $error_F$ inturn depends on the CNN's architecture and loss function which we assume are unknown to the adversary according to a blackbox attack scenario. By including $error_F$ in Eq. (5.6), we only seek to model the adversarial manipulations to training data distributions. We define Eq. (5.6) with reference to misclassification error $error_F$ but not misclassification cost $cost_F$ because we assume that the adversary targets the CNN by manipulating its input data without knowing any more detail of its retraining procedure for participating in the game. By analyzing the convergence of the adversarial cost function $cost_L$ but not the classification cost function $cost_F$, we analyze the stochastic optima of a sequential Stackelberg game rather than a simultaneous Stackelberg game. We solve the sequential Stackelberg game of Eq. (5.4) by optimizing the adversary L's cost $cost_L$ for leading the sequential game.

In Eq. (5.6), $error_F$ can also be interpreted as a regularization term on $cost_L$ to be optimized in Eq. (5.4). Here λ controls the relative importance of adversarial cost function $cost_L$ compared to misclassification error $error_F(w)$ for creating adversarial manipulations $\alpha_*^\mu, \alpha_*^\sigma$ in each game iteration. The tradeoff between decreasing cost $cost_L$ and increasing error $error_F(w)$, with respect to optimization parameters solving for our blackbox attack scenario, plays out across all of the search iterations described in our game theoretical algorithms of Section . L's strategy space in Algorithm 11 and Algorithm 12 is defined on the original data distributions and encoded data distributions leading to global optima solving for the Nash equilibrium of the Stackelberg game. L's strategy space in Algorithm 13 is determined by the SA parameters solving for the local optima in step and direction of each adversarial manipulation. When we consider all the adversarial manipulations learnt by SA parameters in a variational model, we solve for a stochastic optimization problem that is nonlinear and non-convex in the terms of the objective function in Eq. (5.4).

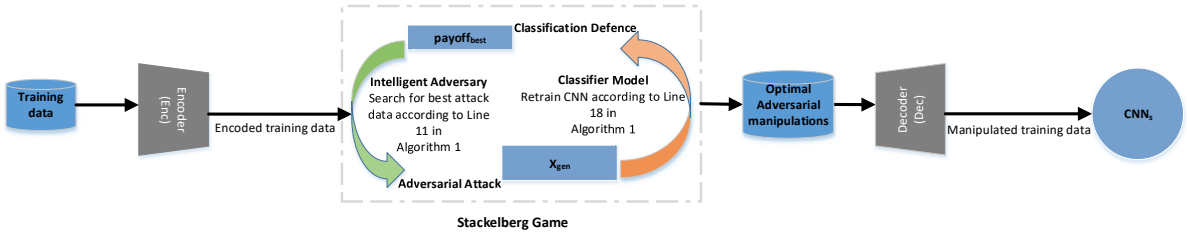


Figure 5.1: A flowchart illustrating the variational Stackelberg game theoretical adversarial learning.

Alongwith SA parameter settings described above, VAE's encoder function Enc and decoder function Dec determine our game theoretical attack scenarios. Given training data X_{train}, we generate adversarial data X_{gen} via:

$$X_{gen} = Dec(\mu_{pos} + \alpha^{\mu}, \sigma_{pos} + \alpha^{\sigma}) \cup X_{train}[neg], \tag{5.7}$$

where $\mu_{pos}, \sigma_{pos} = Enc(X_{train}[pos])$. The Eq. 5.7 first encodes positive data distribution $X_{train}[pos]$ into multivariate Gaussian distribution's parameters $(\mu_{pos}, \sigma_{pos})$. Then it adds adversarial manipulations $(\alpha^{\mu}, \alpha^{\sigma})$ to $(\mu_{pos}, \sigma_{pos})$. After that, the encoded adversarial data is decoded by Dec into original data space. The distribution of negative data $X_{train}[neg]$ is unchanged since we target pos class performance of CNN classifier.

5.1.1 Overall Structure of Our Model

Figure 5.1 is a flow chart of our adversarial learning process that accounts for the presence of a variational adversary in supervised learning. The final outcome of our adversarial learning is a CNN classification model CNN_{secure} (henceforth shortened as CNN_s) that is robust to the adversarial attacks.

We generate the adversarial data in a two-player Stackelberg game between the adversary and the classifier. The adversary creates a variational model by searching for adversarial manipulations on encoded training data. Every statistical parameter of the encoded training data is searched according to a Simulated Annealing (SA) procedure. The aggregation of adversarial manipulations to all statistical parameters in the encoded training data is optimized according to an Alternating Least Squares (ALS) procedure. The ALS optimization is invoked at each time when the adversary generates adversarial data X_{gen} in the Stackelberg game. X_{gen} acts as a validation data for the classifier under attack. For every X_{gen}, the classifier re-optimizes its training weights to update itself.

The result of such a game-theoretic interaction between the learner's and classifier's best moves is quantified by the adversary's payoff $payoff_{best}$. The adversary engages

the classifier in the Stackelberg game as long as the $payoff_{best}$ increases. A decrease of $payoff_{best}$ indicates that Nash equilibrium exit condition has been reached in the Stackelberg game. At the end of the game, adversary has optimal adversarial manipulations from the most recent X_{gen}. Such manipulations are applied on the training data to obtain attacked training data. Then the classifier's learning process adds the attacked data into the original training data so that the CNN_s can be optimally retrained by our adversarial attacks. While the CNN classifier is trained in the original data space, the adversary generates data manipulations in the encoded data space. A variational representation of the encoded data space allows the adversary to propose a generative model for the adversarial manipulations.

5.1.2 The differences between our method and GANs

Although both GAN and our method are based on the framework of game theory and both of them are seeking for the Nash equilibrium, they have some great differences. In this section, we summarize these differences into three aspects, including the model construction, model optimization and the optimization results.

Model construction. Firstly, our variational Stackelberg game is a variable-sum problem according to Eq. (5.4), while the GANs construct a constant-sum game. Secondly, our method defines the adversary as the game leader, whereas the GAN is led by a Generator. Lastly, the GANs define attack scenarios to discover generative models underlying given data distribution, while we optimize adversarial payoff functions with evolutionary attack parameters defining our attack scenarios in randomized strategy spaces.

Model optimization. Firstly, GANs solve a convex optimization problem with gradient-based optimization algorithms. By contrast, we solve a stochastic optimization problem with a simulated annealing algorithm. Specifically, our game's Nash equilibrium is computed by solving non-convex optimization problem in Eq. (5.5). Secondly, during game-theoretic adversarial training, we query CNN about the attack performance $error_{pos}(w)$, while GANs query CNN to distinguish between "real" data and "fake" data.

Optimization results. This is the most significant difference between GANs and our model. Firstly, GAN's objective at the Nash equilibrium is to learn a generative model that mimicks the original distribution of data, while our method learns the optimal adversarial manipulations $(\alpha_*^{\mu}, \alpha_*^{\sigma})$ that are not the original true distribution of the data but are *manipulations* to the original distribution. Secondly, GAN's discriminator at Nash equilibrium is unable to classify between labels, while our classifier is robust to ad-

versarial manipulation and its defence performance is measured by $error_{Pos}(w)$. Thirdly, in our blackbox attack scenario, different proposals on adversarial payoff functions and adversarial cost functions in Eq. (5.6) lead to different Nash equilibria for Eq. (5.5) and corresponding adversarial manipulations in Eq. (5.3). In contrast, a GAN always tries to converge to training data distribution.

5.1.3 Variational Game and Adversarial Examples Illustration

To demonstrate examples of successful attacks, Figure 5.2 shows adversarially manipulated samples generated by our model. For the two images in each subfigure of Figure 5.2, the left one is the original image while the right is the manipulated image. The original images shown for specific target classes are from the MNIST database and the VGGFace2 database. We evaluate adversary's payoff in Eq. (5.6) over data distributions of many such manipulated and misclassified images.

Specifically, Figure 5.2(a) to Figure 5.2(e) show adversarially manipulated MNIST images that have been misclassified to the class label 8. In Figure 5.2(a) and Figure 5.2(d), handwritten digits 2 and 6 have been manipulated by deleting pixels to smoothen sharp edges. In Figure 5.2(b) and Figure 5.2(c), handwritten digits 3 and 5 have been manipulated by adding and deleting pixels to change orientation. In Figure 5.2(e), handwritten digit 9 has been manipulated by adding pixels targeting particular aspects of the image geometry that are similar between images of 9 and 8.

Figure 5.2(f) to Figure 5.2(j) show adversarially manipulated VGGFace2 images that have been misclassified as belonging to the class label "Jackie Chan". In Figure 5.2(f) and Figure 5.2(h) images of "Andy Lau" and "Zhang Jingchu" have been manipulated by blurring parts of the facial features while leaving the remaining background unchanged. In Figure 5.2(g) and Figure 5.2(i) images of "Ajay Devgan" and "Aishwarya Rai Bachchan" have been manipulated by changing the shapes of sunglasses and hand covering the targeted classes facial features. In Figure 5.2(j), image of "Jada Pinkett Smith" has been manipulated by decreasing the distinction between the colours of the image foreground and the image background.

5.2. Variational Stackelberg Game Method

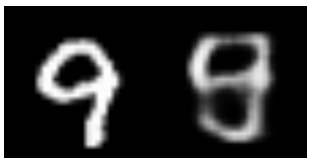
The pseudocode for two-player game which outputs multi-label adversarial manipulations $(\mathbb{A}_*^{\mu}, \mathbb{A}_*^{\sigma})$ is given in Algorithm 11. Each pair (pos, neg) of positive labels $pos \in Pos$ and

(a) 2 misclassified by CNN as 8

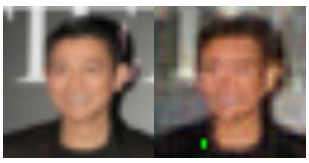
(b) 3 misclassified by CNN as 8

(c) 5 misclassified by CNN as 8

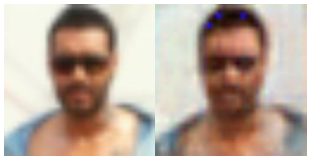
(d) 6 misclassified by CNN as 8



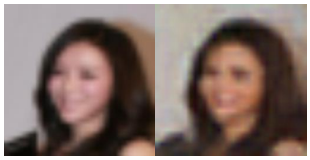
(e) 9 misclassified by CNN as 8



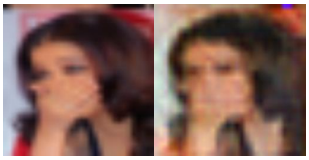
(f) Andy Lau misclassified by CNN as Jackie Chan



(g) Ajay Devgan misclassified by CNN as Jackie Chan



(h) Zhang Jingchu misclassified by CNN as Jackie Chan



(i) Aishwarya Rai Bachchan misclassified by CNN as Jackie Chan

(j) Jada Pinkett Smith misclassified by CNN as Jackie Chan

Figure 5.2: Examples of transformed images found at Nash equilibrium in a Stackelberg game. For the two images in each subfigure, the left one is the original image and the right is the manipulated image.

negative labels $neg \in Neg$ executes a separate two-player game that misleads classifier into misclassifying pos data as neg data. The input to Algorithm 11 is labelled training data X_{train} consisting of grayscale images. A labelled testing data X_{test} is separately used to evaluate $(\mathbb{A}_*^\mu, \mathbb{A}_*^\sigma)$ at the end of each game. Algorithm 11 also defines some input parameters to characterize the game theoretical adversary and find an optimal attack from the strategy space. Candidate adversarial manipulations $(\alpha^\mu, \alpha^\sigma)$ search and optimization over the adversarial payoff function $J_L = payoff_{best}$ is described in

the Alternating Least Squares algorithm of Algorithm 12 and Simulated Annealing algorithm of Algorithm 13.

Line 2 of Algorithm 11 trains a classifier model CNN on X_{train} to initialize three models $CNN_{original}$, $CNN_{manipulated}$ and CNN_{secure} (henceforth shortened as CNN_o, CNN_m and CNN_s, respectively) with same weights that provide a high classification performance on X_{test}. During the game, adversary repeatedly attacks CNN classifier with data manipulations $(\alpha_*^\mu, \alpha_*^\sigma)$. As a response, CNN classifier is allowed retraining its weights w^* by using both X_{gen} and X_{train} on line 18. At the end of the game, adversary converges to data manipulations $(\mathbb{A}_*^\mu[pos,neg], \mathbb{A}_*^\sigma[pos,neg])$, which gives the highest value for adversarial payoff function $J_L = payoff_{best}$.

Algorithm 11 Variational Stackelberg Game Algorithm

Input:

1: Training data X_{train}, Labelled testing data X_{test} (for final evaluations only), Positive and negative training labels Pos and Neg, Encoding space code size $s = 50$, Cost weighting constant $\lambda = 1$, Maximum number of steps in simulated annealing $N = 10$, Lower bound for step interval in simulated annealing $l = 1$, Upper bound for step interval in simulated annealing $h = 5$, Step decrement $d = 0.5$ on h in simulated annealing, Upper bound for iteration count in simulated annealing $maxiter = 100$, Iteration error threshold's $maxerror\_game = 15$, $maxerror\_als = 10$, $maxerror\_sa = 5$ for Stackelberg Game, Alternating Least Squares, Simulated Annealing respectively

Output: Adversarial manipulations in variational game $\mathbb{A}_*^\mu, \mathbb{A}_*^\sigma$, Original performance $error_o$, Attack performance $error_m$, Defence performance $error_s$

2: Train CNN on X_{train} to output model CNN_o. Initialize CNN_m, CNN_s with CNN_o's weights.
3: Train Variational Autoencoder VAE on X_{train} to get Encoder function Enc, Decoder function Dec.
4: **for** $(pos,neg) \in (Pos \times Neg)$ **do**
5: $\mu_{neg}, \sigma_{neg} = Enc(X_{train}[neg])$, $\mu_{pos}, \sigma_{pos} = Enc(X_{train}[pos])$
6: $\delta_\mu = mean(\mu_{neg}) - mean(\mu_{pos})$, $\delta_\sigma = mean(\sigma_{neg}) - mean(\sigma_{pos})$
7: Initialize zeros tensors α_*^μ and α_*^σ with same shape as $mean(\mu_{pos})$ and $mean(\sigma_{pos})$ respectively
8: exitgame = False, $payoff_{curr} = error_{curr} = -\infty$, $CNN = CNN_o$
9: **while** exitgame = False **do**
10: Generate adversarial data X_{gen} by substituting $(\alpha_*^\mu, \alpha_*^\sigma)$ in Eq. (5.7). Evaluate CNN on X_{gen} to find error $error_{curr}$. Substitute $error_{curr}$ in Eq. (5.6) to calculate $payoff_{curr}$ for $(\alpha_*^\mu, \alpha_*^\sigma)$.
11: Update α_*^μ and α_*^σ from alternating least squares ALS in Algorithm 12
12: Generate adversarial data X_{gen} by substituting $(\alpha_*^\mu, \alpha_*^\sigma)$ in Eq. (5.7). Evaluate CNN on X_{gen} to find error $error_{best}$. Substitute $error_{best}$ in Eq. (5.6) to calculate $payoff_{best}$ for $(\alpha_*^\mu, \alpha_*^\sigma)$.
13: **If** $payoff_{best} - payoff_{curr} > 0$ **then**
14: **If** $error_{best} > maxerror\_game$ **then**
15: exitgame = True, break //*Nash equilibrium achieved*
16: **else**
17: exitgame = False
18: Re-optimize CNN weights on manipulated data and training data $X_{gen} \cup X_{train}$
19: $X_{train-manip} = X_{train-manip} \cup X_{gen}$
20: **else**
21: exitgame = True
22: **end while**
23: $\mathbb{A}_*^\mu[pos,neg], \mathbb{A}_*^\sigma[pos,neg] = \alpha_*^\mu, \alpha_*^\sigma$
24: **for** $(pos,neg) \in (Pos \times Neg)$ **do**
25: Simulate adversarial attacks by $\mathbb{A}_*^\mu[pos,neg], \mathbb{A}_*^\sigma[pos,neg]$ on testing data $X_{test}[pos,neg]$ to output manipulated testing data $X_{test-manip}[pos,neg]$
26: Evaluate CNN_o on $X_{test}[pos,neg]$ to find original model error $error_o[pos,neg]$ for targeted label pos in attack scenario
27: Evaluate CNN_m on $X_{test-manip}[pos,neg] \cup X_{test}[pos,neg]$ to find manipulated model error $error_m[pos,neg]$ for targeted label pos in attack scenario
28: Train CNN_m on $X_{train-manip}[pos,neg]$ to output secure model CNN_s
29: Evaluate CNN_s on $X_{test-manip}[pos,neg] \cup X_{test}[pos,neg]$ to find secure model error $error_s[pos,neg]$ for targeted label pos in attack scenario
30: return $(\mathbb{A}_*^\mu, \mathbb{A}_*^\sigma, error_o, error_m, error_s)$

Then, CNN_o is evaluated on testing data X_{test} to output original performance

$error_{original}$ (henceforth shortened as $error_o$) on line 25. On line 26, $CNN_{manipulated}$ is evaluted on manipulated testing data $X_{test-manip}$ as well as testing data X_{test} to output attack performance $error_{pos}(w) = error_{manipulated}$ (henceforth shortened as $error_m$). On line 27, CNN_m is retrained on manipulated training data $X_{train-manip}$ to output secure classifier CNN_s. On line 28, CNN_s is evaluated on manipulated testing data $X_{test-manip}$ and testing data X_{test} to output defence performance $error_{Pos}(w) = error_s$ (henceforth shortened as $error_s$). Notably, $error_o$, $error_m$, $error_s$ are calculated for each $(pos, neg) \in (Pos \times Neg)$.

Before the game begins, CNN_o's weights are learnt on training data X_{train} and CNN_m's weights are duplicate of CNN_o's weights. After the game ends, CNN_m weights and CNN_o weights remain unchanged. In the presence of an adversary, CNN_s weights are more robust than CNN_o weights. Classifier model is called CNN. CNN is initially trained on line 2, then it is adversarially trained on line 18 for every adversarial attack. CNN is discarded after game ends for each label pair (pos, neg).

The adversary's strategy space in Algorithm 12 is determined by an Encoder function Enc and a Decoder function Dec of a VAE model. The VAE encodes input data in terms of a mean vector μ and a variance vector σ to represent a multivariate Gaussian distributions in the latent space. In Algorithm 12, the size of latent space is given as attack scenario parameter s. On line 5 of Algorithm 12, Enc is applied on positive data $X_{train}[pos]$ and negative data $X_{train}[neg]$ to output positive mean vector μ_{pos} and negative mean vector μ_{neg}; positive variance vector σ_{pos} and negative variance vector σ_{neg}, respectively. These vectors are used on line 6 to derive vectors δ_{mu} and δ_{sigma} that measure amount of change in latent space to entirely convert encoded positive data $Enc(X_{train}[pos])$ into encoded negative data $Enc(X_{train}[neg])$.

The adversarial learning targets to search for small random changes $(\alpha_*^\mu, \alpha_*^\sigma)$ to attack the encoded positive data $(\mu_{pos}, \sigma_{pos})$ on line 10 and line 12 such that CNN predicts a negative label neg for the manipulated positive data $(\mu_{pos} + \alpha_*^\mu, \sigma_{pos} + \alpha_*^\sigma)$ after it is decoded into training data space. Since search for $(\alpha_*^\mu, \alpha_*^\sigma)$ is in VAE's latent space, the adversary's strategy space for attacker's payoff calculation in the two-player game is the VAE's latent space. However, the classifier's strategy space for defender's payoff calculation is the original data space, which is the same as reconstructed data space.

Line 10 evaluates current CNN on X_{gen} to find classification error $error_{curr}$. Then it computes adversarial payoff function value $payoff_{curr}$ by combining $error_{curr}$ with adversarial cost function, where the adversarial cost function is defined as the Forbenius norm of corresponding adversarial manipulation $(\alpha_*^\mu, \alpha_*^\sigma)$. λ is a weight constant that

Algorithm 12 Alternating Least Squares

1: **function** ALS($\alpha^{\mu}_{*}, \alpha^{\sigma}_{*}, \delta_{\mu}, \delta_{\sigma}, payoff_{curr}, error_{curr}, \mu_{pos}, \sigma_{pos}, s$)
2: exitsearch = False
3: **while** ¬ exitsearch **do**
4: $payoff_{best}, error_{best} = payoff_{curr}, error_{curr}$
5: $\alpha^{\mu}_{best}, \alpha^{\sigma}_{best} = \alpha^{\mu}_{*}, \alpha^{\sigma}_{*}$
6: By fixing α^{σ}_{best}, update α^{μ}_{best}, $payoff_{best}$, $error_{best}$ from simulated annealing SA in Algorithm 13 for given δ_{μ} and μ_{pos}
7: By fixing α^{μ}_{best}, update α^{σ}_{best},$payoff_{best}$, $error_{best}$ from simulated annealing SA in Algorithm 13 where δ_{σ} replaces δ_{μ} and σ_{pos} replaces μ_{pos}
8: **If** $payoff_{best} - payoff_{curr} > 0$ **then**
9: **If** $error_{best} > maxerror\_als$ **then**
10: exitsearch = True, break
11: **else**
12: exitsearch = False
13: **else**
14: exitsearch = True
15: $payoff_{curr}, error_{curr} = payoff_{best}, error_{best}$
16: $\alpha^{\mu}_{*}, \alpha^{\sigma}_{*} = \alpha^{\mu}_{best}, \alpha^{\sigma}_{best}$
17: **end while**
18: return $(\alpha^{\mu}_{*}, \alpha^{\sigma}_{*})$
19: **end function**

controls the contribution of adversarial cost function when calculating the adversarial payoff function. In a given game execution and training data distribution, λ is set by adversary to increase payoff function by increasing classifier's error and decreasing adversary's cost across game plays.

On line 11, the Alternating Least Squares method (see Algorithm 12) updates candidate adversarial manipulations $(\alpha^{\mu}_{*}, \alpha^{\sigma}_{*})$. Line 12 updates classifier error $error_{best}$ and adversarial payoff $payoff_{best}$ corresponding to adversarial data X_{gen}. The looping condition on line 13 continues game plays as long as $payoff_{best}$ is greater than $payoff_{curr}$. All game plays end on line 22 when it converging to the optimal adversarial manipulation $(\alpha^{\mu}_{*}, \alpha^{\sigma}_{*})$ for label pair (pos, neg).

5.2.1 Alternating Least Squares Algorithm

Algorithm 12 illustrates the Alternating Least Squares (ALS) search procedure, which optimizes the adversarial manipulations by changing the elements of vectors $\alpha^{\mu}_{best}, \alpha^{\sigma}_{best}$. By using the Simulated Annealing (SA) method (see Algorithm 13) on line 6 and line 7, ALS can alternatively change each element of mean vector α^{μ}_{best} and variance vector α^{σ}_{best}, while fixing all the other elements. ALS optimizations are repeated until adversarial payoff function value $payoff_{best}$ stops increasing in comparison to current payoff function value $payoff_{curr}$. Starting with unchanged $(\alpha^{\mu}_{best}, \alpha^{\sigma}_{best})$ on line 5 of Algorithm 12, we empirically assume an attack scenario where the variance vector α^{σ}_{best} is changed only after mean vector α^{μ}_{best} has been changed in elemental sequence. The maximum number of elements to change is determined by VAE's code size s. After executing multiple search

Algorithm 13 Simulated Annealing

1: **function** SA($\alpha_{best}, \delta_\mu, payoff_{curr}, error_{curr}, \mu_{pos}, s$)
2: $\alpha_{optimal} = \alpha_{initial} = \alpha_{best}, payoff_{optimal} = error_{optimal} = -\infty$
3: **for each** $i \in [0, s]$
4: $\alpha_{next} = \alpha_{initial}$
5: $\epsilon = \frac{\delta_\mu[i]}{N}$
6: exitoptimize = False, $iter = 0, jumped = False$
7: **while** $\neg$ exitoptimize $\wedge$ iter $<$ $maxiter$ **do**
8: $manip_{next} \mathrel{+}= \epsilon \times random(l, h - d \times iter)$
9: $\alpha_{next}[i] \mathrel{+}= manip_{next}$
10: Generate adversarial data X_{gen} by substituting α_{next} in Eq. 5.7. Evaluate CNN on X_{gen} to find error $error_{next}$. Substitute $error_{next}$ in Eq. 5.6 to calculate $payoff_{next}$ for α_{next}.
11: **If** $payoff_{next} - payoff_{curr} > 0$ **then**
12: $payoff_{curr} = payoff_{next}, error_{curr} = error_{next}$
13: **If** $error_{next} > maxerror\_sa$ **then**
14: exitoptimize = True, break
15: **else**
16: exitoptimize = False
17: **else**
18: **If** $error_{next} - error_{curr} > 0 \wedge jumped == False$ **then**
19: exitoptimize = False, $jumped = True$
20: $manip_{next} \mathrel{+}= \epsilon \times random(l, h)$
21: **else**
22: exitoptimize = True
23: $iter \mathrel{+}= 1$
24: **end while**
25: $\alpha_{best} = \alpha_{initial}, \alpha_{best}[i] = \alpha_{next}[i]$
26: Generate adversarial data X_{gen} by substituting α_{best} in Eq. 5.7. Evaluate CNN on X_{gen} to find error $error_{best}$. Substitute $error_{best}$ in Eq. 5.6 to calculate $payoff_{best}$ for α_{best}.
27: **If** $payoff_{best} > payoff_{optimal}$ **then**
28: $\alpha_{optimal} = \alpha_{best}$
29: $payoff_{optimal} = payoff_{best}, error_{optimal} = error_{best}$
30: **end for**
31: return $\alpha_{optimal}, payoff_{optimal}, error_{optimal}$
32: **end function**

iterations, ALS finds optimally changed $(\alpha_*^\mu, \alpha_*^\sigma)$ on line 18 of Algorithm 12. Thresholding $error_{best}$ with input parameter $maxerror\_als$ on line 10 ensures that the ALS search procedure exits as soon as least squares fit of $payoff_{best}$ is found by alternating SA optimizers, operating on the geometric surface of the adversarial payoff function J_L.

5.2.2 Simulated Annealing Algorithm

Algorithm 13 is the derivative-free stochastic optimization procedure for finding adversarial manipulations in terms of simulated annealing (SA) steps to i-th elements of vectors $(\alpha_{best}^\mu, \alpha_{best}^\sigma)$. The step size in SA is set by ϵ on line 5. The bounds for ϵ depend on i-th element of δ_μ defined by Algorithm 11 in the latent space of the VAE. The magnitude of ϵ is determined by the parameter N dividing $\delta_\mu[i]$ into multiple increments. The direction of SA is set by the sign of $\delta_\mu[i]$ determining additive distance between negative data and positive data. The initial candidate for SA is $\alpha_{next}[i]$ where α_{next} has been initialized to candidate adversarial manipulation α_{best}. α_{best} is determined by line 5 of Algorithm 12.

For each $\alpha_{next}[i]$, candidate adversarial manipulations α_{next} are updated on line

9 where $manip_{next}$ is a random multiple of ϵ steps on line 8. The upper bound for selecting the random multiple of steps decreases with SA's iterations to reduce step size across SA iterations. After minimizing adversarial cost function by a greedy optimization strategy, we find a $manip_{next}$ that is close to 0 but is able to mislead CNN in $payoff_{next}$ calculation on line 10. Moreover, $payoff_{next}$ is expected to have low cost of optimization in adversary's strategy space (determined by VAE model) for large classification error in classifier's strategy space (which is the same as the original data space).

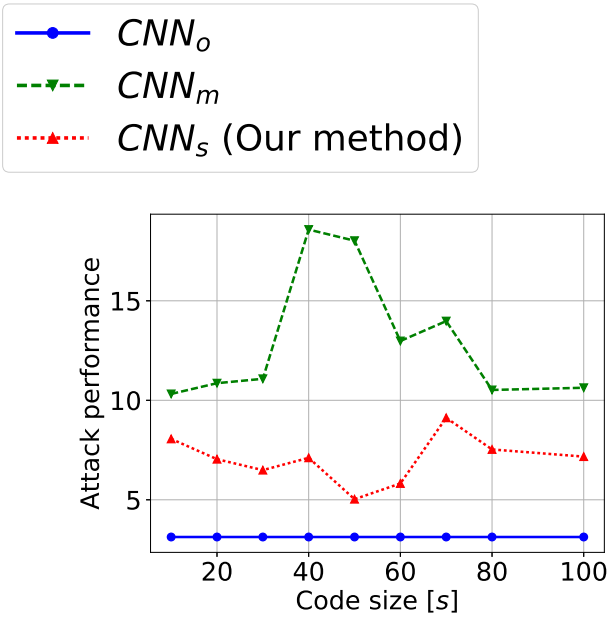
SA's convergence criteria are VAE's encoding space code size s, maximum number of SA steps N, adversarial cost function's weighting constant λ. The SA's optimization condition of increasing payoff $payoff_{next}$ is on line 11. The SA's jump condition of increasing error and exit condition of decreasing payoff as well as error are on line 18 and line 21, respectively. To find a successful adversarial manipulation, SA's optimization condition is expected to be frequently true while SA's jump condition is rarely true. On line 13, SA's optimization of $error_{next}$ is thresholded by parameter $maxerror\_sa$. Line 24 updates only the i-th element of α_{best} with i-th element of α_{next} found by SA. On line 25, $payoff_{best}$ and $error_{best}$ are calculated to obtain adversary's payoff and classifier's error for i-th adversarial manipulation. The sorting condition on line 26 finds $\alpha_{optimal}$ with highest adversarial payoff produced across all the s SA optimizers in stochastic optimization procedure of Algorithm 13.

5.3. Experiments

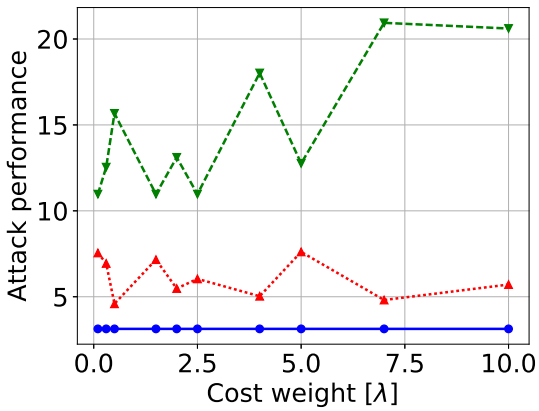
In this section, we validate the game theoretical adversary's attack scenarios in terms of CNN classifer models' performance validation. The attack scenarios are expressed in terms of stochastic optimization parameters in Algorithm 13. The number of times the adversary invokes Algorithm 13 is determined by the VAE's encoder codesize s. For each label pair (pos, neg), the rate of convergence of Algorithm 13 onto optimal payoff is determined by adversarial cost function's weighting constant λ. The step size of annealing operation in Algorithm 13 has upper bound N.

5.3.1 Classifier and Autoencoder Description

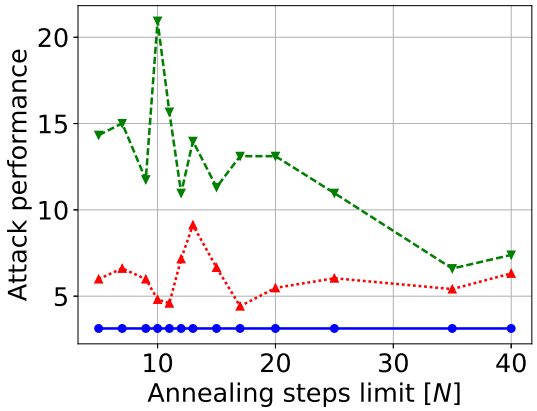
To create training data, testing data and validation data for game theoretical adversary, we conduct experiments on the handwritten images of MNIST database [141] and VGGFace2 database of human faces [48]. During the game, the adversary targets the



(a) Attack: varying s

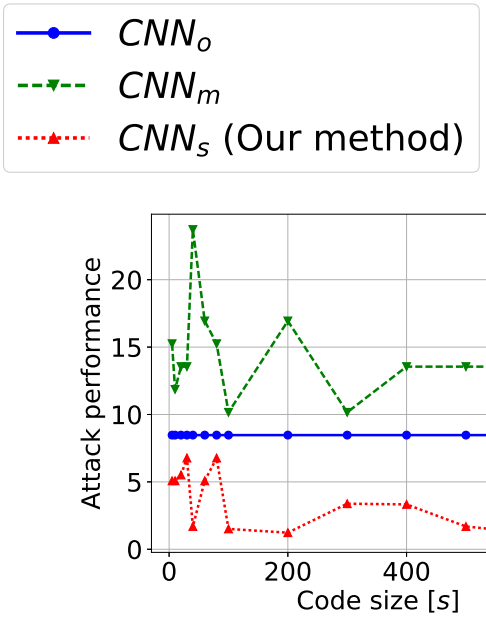


(b) Attack: varying λ

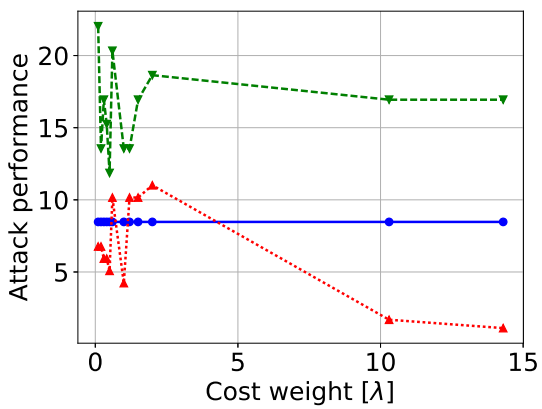


(c) Attack: varying N

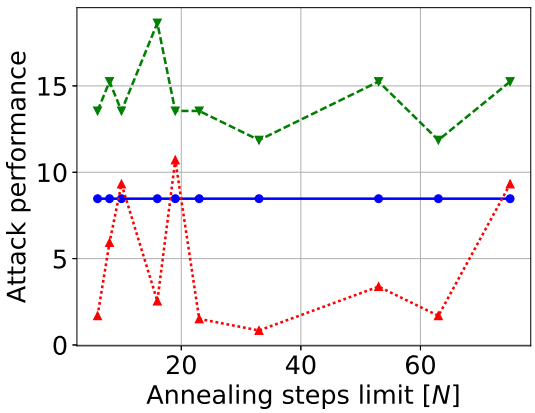
Figure 5.3: MNIST Testing performance (error) with variations in attack parameters consisting of encoder code size s, adversarial cost weight λ and annealing steps limit N.



(a) Attack: varying s



(b) Attack: varying λ



(c) Attack: varying N

Figure 5.4: VGGFace2 Testing performance (error) with variations in attack parameters consisting of encoder code size s, adversarial cost weight λ and annealing steps limit N.

targeted class label pos's true positive rate, which is also called the attack performance. After the game, adversarial data is used to measure CNN's f_1-score for varying pos label, which is defined as the defence performance. All CNNs and VAEs are implemented in Pytorch[1] platform. In each VAE, encoder has fully connected layers representing input data in terms of a multivariate Gaussian distribution with mean vector and variance vector of size s. The VAE's decoder architecture is a mirror of its encoder.

MNIST CNN and VAE: Following the loss function defined by Krizhevsky *et al.* [133], the CNN trained for MNIST has two convolution layers and two fully connected layers leading upto a crossentropy loss function with dropout regularization. The CNN has a learning rate of 0.01 and momentum of 0.5. It is trained for 25 epochs. Following the loss function defined by Kingma *et al.* [124], the MNIST VAE's encoder has two convolution layers. All the input images are in the size of 28×28 pixels. The VAE is trained for 50 epochs.

VGGFace2 CNN and VAE: Following the loss function defined by He *et al.* [106], the CNN trained for VGGFace2 is a pretrained model obtained from torchvision package[2]. The CNN has a learning rate of 0.01 and momentum of 0.9. It is trained for 100 epochs. Following the loss function defined by Hou *et al.* [109], the VGGFace2 VAE's encoder has four convolution layers with batch normalization. All the input images are in the size of 32×32 pixels. The VAE is trained for 1000 epochs.

5.3.2 Attack Performance Validation

Figure 5.3 and Figure 5.4 show the variation in (percentage) attack performance for CNN_o, CNN_m, CNN_s on the y-axis when $pos = 5$ and $neg = 8$ in MNIST database and $pos = AndyLau$ and $neg = JackieChan$ in VGGFace2 database respectively. The x-axis in Figure 5.3(a) and Figure 5.4(a) varies s the code size of mean vector μ_{pos} and variance vector σ_{pos} encoding positive data. s is varied between 10 and 100 for MNIST database. s is varied between 5 and 700 for VGGFace2 database. The x-axis in Figure 5.3(b) and Figure 5.4(b) varies λ the weighting constant on adversarial payoff function. λ is varied between 0.1 and 10 for MNIST database. λ is varied between 0.1 and 15 for VGGFace2 database. The x-axis in Figure 5.3(c) and Figure 5.4(c) varies N the maximum number of steps taken by the adversary applying ALS of Algorithm 12 to optimize the step change to each element of the encoded data parameters μ_{pos} and σ_{pos}. N is varied between 5 and 40 for MNIST database. N is varied between 6 and 75 for VGGFace2 database.

[1] `https://pytorch.org/docs/stable/index.html`

[2] `https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html`

Table 5.2: Alternating Least Squares Attack Scenario: t-tests by varying parameters in Variational Stackelberg games. The small p-values from the statistical tests demonstrate the superiority of our model.

Attack Parameter	p-values from t-statistics for target class "5"			p-values from t-statistics for target class "Andy Lau"		
	CNN_o *vs* CNN_m	CNN_o *vs* CNN_s	CNN_m *vs* CNN_s	CNN_o *vs* CNN_m	CNN_o *vs* CNN_s	CNN_m *vs* CNN_s
Code size (s)	9.9×10^{-8}	3.6×10^{-8}	9.5×10^{-5}	1.9×10^{-4}	4.5×10^{-9}	4.1×10^{-8}
Cost weight (λ)	2.7×10^{-8}	1.8×10^{-7}	3.2×10^{-6}	6.5×10^{-9}	6.1×10^{-2}	1.4×10^{-7}
Annealing steps limit (N)	1.5×10^{-9}	1.1×10^{-8}	2.0×10^{-6}	3.4×10^{-8}	5.6×10^{-3}	1.4×10^{-6}

Table 5.3: MNIST Comparisions on the defence to adversarial Nash equilibrium attacks.

Classification error: Adversarial attack in Variational Stackelberg Game									Class Labels (pos, neg)
$CNN_{original}$	$CNN_{manipulated}$							CNN_{secure} (*Our method*)	
	CNN	*DCGAN* [194]	*IWGAN* [99]	*DeepFool* [173]	*FGSM* [91]	CNN_{GA} [57]	CNN_{SA} [57]		
1.12	10.01	40.16	39.41	32.06	34.66	25.93	25.81	6.44	(2,8)
1.54	54.74	47.12	39.84	46.15	37.71	27.88	28.76	5.11	(3,8)
3.86	24.16	26.18	26.75	26.59	26.57	28.58	26.82	6.33	(4,8)
1.87	8.78	50.68	53.31	56.71	35.91	27.94	28.49	5.31	(5,8)
1.21	12.41	39.41	36.16	47.19	35.58	25.77	26.16	5.39	(6,8)
4.89	23.91	28.78	27.48	27.03	27.28	27.65	27.81	4.82	(7,8)
2.53	16.97	41.21	37.22	54.88	36.85	26.86	27.34	3.58	(9,8)
t-statistics	1.9×10^{-2}	3.7×10^{-7}	6.8×10^{-7}	7.6×10^{-6}	2.0×10^{-9}	3.5×10^{-14}	4.6×10^{-14}	Base	

Table 5.4: VGGFace2 Comparisons on the defence to adversarial Nash equilibrium attacks

Classification error: Adversarial attack in Variational Stackelberg Game									Class Labels (pos,neg)
$CNN_{original}$	$CNN_{manipulated}$							CNN_{secure} (*Our method*)	
	CNN	*DCGAN* [194]	*IWGAN* [99]	*DeepFool* [173]	*FGSM* [91]	CNN_{GA} [57]	CNN_{SA} [57]		
5.25	44.44	62.21	45.34	45.99	46.13	32.37	92.0	26.53	(Andy Lau, Jackie Chan)
20.19	37.71	73.33	56.31	26.45	33.81	22.07	61.34	13.09	(Zhang Jingchu, Jackie Chan)
20.11	35.19	27.64	31.13	32.64	34.91	33.88	26.27	16.66	(Ajay Devgan, Jackie Chan)
8.08	31.41	71.84	35.89	20.38	22.03	23.28	17.51	12.31	(Aishwarya Rai Bachchan, Jackie Chan)
21.92	32.62	81.94	31.75	26.27	32.17	33.01	84.51	11.74	(Amy Smart, Jackie Chan)
16.77	28.13	21.61	26.08	56.14	54.36	95.08	18.24	20.52	(Jada Pinkett Smith, Jackie Chan)
2.24	35.09	87.5	34.49	85.91	79.83	100	94.49	11.07	(Bruce Willis, Jackie Chan)
24.44	31.31	67.93	27.17	28.35	36.67	96.69	26.08	9.57	(Oprah Winfrey, Jackie Chan)
t-statistics	4.8×10^{-6}	1.1×10^{-4}	1.7×10^{-4}	7.3×10^{-3}	1.0×10^{-3}	8.1×10^{-3}	8.7×10^{-3}	Base	

5.3.3 Defence Performance Validation

While s determines the invocation times of SA in Algorithm 13, λ determines the change rate of $payoff_{best}$ crossing iterations in Algorithm 13. N sets the infinitesimal step size ϵ in Algorithm 13.

For each attack parameter value (and adversarial dataset) fixed on the x-axis, each data point triplets marked on y-axis. Figure 5.3 and Figure 5.4 show that the manipulated CNN (CNN_m) has higher error than the original CNN (CNN_o), and secure CNN

(CNN_s) has lower error than manipulated CNN (CNN_m). Thus adversarial data has successfully misled CNN_o into misclassifying pos data as neg data, which is measured by true positive rate of pos in CNN's predictions. The exact change in error between CNN_m and CNN_o is determined by the game's iterations thresholds, i.e., $maxerror\_game$, $maxerror\_als$ and $maxerror\_sa$, which act as the exit condition on the game's convergence criteria. These optimization criteria estimate optimal adversarial manipulations such that adversarial cost function tends to relatively low value and classification error function yields a relatively high value.

Table 5.2 shows the results of statistical hypothesis testing across different attack performances, such as, CNN_o, CNN_m and CNN_s, under all the attack parameters values and datasets as mentioned in Figure 5.3 and Figure 5.4. Each row in Table 5.2 lists the attack parameter for statistical testing. Each column in Table 5.2 gives results of a two-sample t-test for each pair of CNN_o, CNN_m, CNN_s classifiers. The null hypothesis in each t-test states that the classifier performances are the same before and after adversarial manipulation. Assuming a normal distribution for the attack performances calculated over attack scenarios, the alternative hypothesis in each t-test states that across attack parameter settings (and adversarial datasets), the manipulated classifier has higher error than original classifier and the secure classifier has lower error than manipulated classifier.

Table 5.3 and Table 5.4 give the CNN defence performances calculated from f_1-score with varying (pos, neg) label pairs. Across the table rows, the attack parameters are fixed to target-dependent tuples, such as $s = 50$, $\lambda = 50$ and $N = 30$. Varying target class pos from 2 to 9 allows us to create different adversarial data manipulations on both the original data and the generated data. The defence performance of original CNN (CNN_o) and secure CNN (CNN_s) are our baseline for performance validation in Table 5.3 and Table 5.4. The defence performances of manipulated CNNs (CNN_m) in Table 5.3 and Table 5.4 are calculated on the original data distribution as well as the generated data distribution. CNN_o's defence performance is created on training data of either original data distribution or generated data distribution. By adversarially manipulating the corresponding testing data, CNN_m's defence performance is created. CNN_s's defence performance is created by training on adversarially manipulated training data and testing on adversarially manipulated testing data.

The original training data and original testing data is cross-validation data created from the MNIST database of handwritten digits [141] and VGGFace2 database of human faces [48]. The generated data is created from the outputs of Deep Convolutional

Generative Adversarial Network (DCGAN) [194] and Improved Wasserstein Generative Adversarial Network (IWGAN) [99], which are generative adversarial networks (GANs) trained to produce images. The generated data is also created from existing adversarial examples in deep learning networks, i.e., DeepFool [173] and Fast Gradient Sign Method (FGSM) [91]; game theoretical adversaries genetic algorithm CNN_{GA} [57] and annealing algorithm CNN_{SA} [57]. The p-values from two-sample t-tests in the last row of Table 5.3 compare CNN_m's performances with CNN_s (which acts as the classification baseline for statistical comparisons).

The null hypothesis for t-test is that our adversarial manipulations are unable to attack existing adversarial classifiers where classifier's defence performance is calculated in terms of f_1-score of targeted class labels. The alternative hypothesis for t-test is that our adversarial manipulations are able to attack classifiers trained on existing sources for original data, generated data and adversarial data. Moreover, our secure classifier is robust to the proposed data manipulations in comparision to the existing adversarial classifiers.

Therefore, across multiple two-label two-player sequential variable-sum Stackelberg games played over both original data and generated data, CNN_m has greater error than CNN_o error in Table 5.3 and Table 5.4. At the same time CNN_s has lesser error than CNN_m. From low p-values (<0.05) in Table 5.3 and Table 5.4 we are able to reject the null hypothesis in t-tests comparing CNN_m with CNN_s. The errors thresholds $maxerror\_game$, $maxerror\_als$, $maxerror\_sa$, which manipulates training data in the game's iterations, allow us to control the amount of adversarial data that is injected into the testing data to achieve desired defence performance after game's convergence. On a given dataset, a desired value for $maxerror\_game$, $maxerror\_als$, $maxerror\_sa$ can be obtained by experimenting with parameter settings for λ, s, N, respectively. Varying the negative label neg is expected to create new values for the defence performances in Table 5.3, but not change the conclusions from t-tests over same set of positive labels pos.

From p-values of the t-tests, we also conclude that we are able to create attack data for the adversarial examples generated by GANs and adversarial networks. In comparision to the attack scenarios in Chivukula and Liu [57], our improved search and convergence criteria in the changed optimization problem are able to find better adversarial manipulations at the Nash equilibrium. These manipulations are able to attack the game theoretical adversarial learning proposed by Chivukula and Liu [57]. Finally we are then able to produce a more robust CNN_s classifier in comparision to

existing literature.

5.4. Findings Summary

Unlike existing literature in adversarial learning, our game theoretical adversaries do not solve a optimization problem with with well defined gradients of the objective function. Thus the theory of search and optimization in our adversarial algorithms cannot guarantee the convergence of the game-theoretic modelling to the global optimum finding Nash equilibrium. This observation motivates us to empirically improve the search procedure to solve for a better local optimum with the help of better computation procedures in the misclassification modelling baselines. The new Nash equilibria are empiricially computed from a balance of players' payoff functions that influences each player oppositely until convergence.

Our game theoretical objective function does not support backpropagation of errors through non-convex layers and losess in CNN and VAE. In this chapter we propose a computational intelligence based annealing algorithm to create a generative model for the optimal data manipulations. The annealing algorithm is expected to find better solutions than simple greedy heuristics that craft adversarial data with minimum possible change to original data distribution.

In this chapter we study and improve the optima solving Stackelberg games included in the training procedure of predictive models. We are able to find optimum that create new adversarial datasets for statistical significance testing in our experiments. To create the adversarial data from original data we have studied variational adversaries participating in the Stackelberg game with the image recognition models. Such adversaries are able to create a representation model for the adversarial manipulations in an encoded search space of probability vectors. In comparison to the existing adversarial training procedures in literature, we create adversarial data with deep generative models. Our variational training procedures are better able to regularize the adversarial cost function and optimize the adversarial payoff function occurring as sub-problems in the game theoretical learning objective of the image recognition models. In the experiments, we demonstrate the effect of SA and ALS attack parameters on the local search and global optimization of the proposed (sequential variable-sum two-player) game theoretical learning objective function.

In model validation criteria for solving the optimization problems in the algorithms, we study convergence of an adversarial cost function (rather than a classification cost

function) because we solve a sequential game where the adversary is the leader and the classifier follows the leader in the game (rather than a simultaneous game where the players move simultaneously). The interplay of adversarial cost function and classification error function has also been elaborated.

In this research on robust classification, similar to the empirical evaluations of GANs which are potentially a general learning framework but used mostly images for experiments (although text-learning-based variations of GANs do exist separately), our present focus is on CNNs classifying images. In future work we do want to extend this work to text data and natural language processing problems, which will require different types of feature manipulation strategies and substantially different learning models.

We retrained GAN using the manipulated data and evaluated its robustness against the adversarial attacks in the experiments. We have made comparisons against models retrained on manipulated data using GAN models, the DeepFool model, and also the Fast Grandient Sign methods. We have discussed the differences in game theory between GANs and our method. GAN at its optima is to best distinguish synthetic images from true ones that belong to the same class label (i.e., a generative learning problem), rather than to classify labels of synthetic images at the highest accuracy (i.e., a label classification problem). From this perspective, the GAN may not be expected to perform well when an image is deliberately and subtly changed to mislead the classifier to misclassify it to a targeted wrong label.

We formulated the adversarial payoff function on a convex strategy space determined by a Variational Autoencoder (VAE) model. For future investigations, we are interested in the effect of encoded vectors on manipulated data with discontinuities in the latent space.

Part II

CHAPTER 6

DISCOVERING GRANGER-CAUSAL FEATURES WITH DEEP LEARNING NETWORKS

Causal inference is a central theme in computational sciences that construct mathematical models for causation. In statistics, causality is defined over conditional dependencies modelled between features in the data. Such conditional dependencies are used to construct data distributions linking causes with effects in causal relations defined on data features. Such causal relations are useful for feature discovery in machine learning. The impact and risk of including causal relations or causal features is validated by domain knowledge.

Causality is generally defined on logical formalizations of different classes of knowledge, reasoning and complexity in data. Causality also depends on features and representations, patterns and noise from ground truth data generated in an application domain. Depending on a particular definition of causality, causal relations identify causal features for machine learning.

The Granger-Sargent statistic and the Granger-Wald statistic are commonly used to discover Granger-causal features on time-domain and frequency-domain formulations of Granger causality [100]. In this chapter we discover Granger-causal features by measuring model improvement in deep networks. Our models are useful for simulating time-dependent observations in application domains with neural computations in deep learning.

Deep learning is a class of neural networks that learn hierarchical feature repre-

sentations approximating non-linear functions. In data-driven analytics applications, deep learning has been used to visualize, store, process and predict information. In supervised deep learning, the information is typically modelled as statistical correlations and variable associations. Introducing causality methods into supervised deep learning creates analytics models for data-driven decision making in an application domain where causal features are separated from spurious features.

In this chapter, we analyze time series data distributions with the help of deep learning networks to discover causal relations and causal graphs from Granger causality tests [93]. To derive data representations, the deep networks are trained to optimize squared error loss functions between actual data and predicted output. The corresponding analytics predictions are tested and validated with statistical significance tests on regression errors. We also extend unrestricted models in Granger causality for supervised feature discovery with bivariate regression as well as supervised causal inference with multivariate regression. Theoretically, the deep network architecture and its squared error loss function determine empirical risk in our regression models.

6.1. Our Proposed Algorithms

We predict stock prices in financial markets with Deep Neural Networks (DNNs) for discriminative learning based regression models and Recurrent Neural Networks (RNNs) for sequence learning based regression models. Outputs from bivariate regression models are used to search Granger-causal features in multivariate time series data.

6.1.1 Empirical Risk training in Deep Learning Networks

Suppose a regression model for stock y having actual value $y(t)$ at time t predicts $\hat{y}(t;\boldsymbol{\alpha})$ parameterized by regression parameters $\boldsymbol{\alpha}$ belonging to parameter space A. In computational learning theory, the regression model is analyzed in terms of expected risk $E(L(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$, which is defined as expected value of the loss function $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$, learning probability density function $P(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ underlying the data [10]:

(6.1) Expected Risk : $E(L(\hat{y}(t;\boldsymbol{\alpha}),y(t))) = \int d(\hat{y}(t;\boldsymbol{\alpha}))d(y(t))L(\hat{y}(t;\boldsymbol{\alpha}),y(t))P(\hat{y}(t;\boldsymbol{\alpha}),y(t))$

The expected risk $E(L(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$ is posed as a regression model when loss function $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ is defined on squared errors computed between $\hat{y}(t;\boldsymbol{\alpha})$ and $y(t)$. If the regression model defining $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ is learning a training dataset of finite size m, then expected risk $E(L(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$ is called empirical risk [88] $\hat{E}(L(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$.

$$\text{Empirical Risk} : \hat{E}_{y(t)\sim P(\hat{y}(t;\boldsymbol{\alpha}),y(t))}(L(\hat{y}(t;\boldsymbol{\alpha}),y(t))) = \frac{\Sigma_{i=1}^{m} L(\hat{y}(t;\boldsymbol{\alpha})^{(i)},y(t)^{(i)})}{m} \tag{6.2}$$

The computational complexity of empirical risk $\hat{E}(L(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$ is determined by the computational complexity of $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ which in turn is determined by the regression model's feature selection and model validation. Thus, our intuition is that introducing causal features into deep networks not only minimizes empirical risk but also minimizes regression error.

In our deep network based regression models, regression error is minimized by the weights $\boldsymbol{\alpha}$ learnt on Squared Error (SE) Loss function $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ as in Equation 6.3:

$$\text{SE Loss} : L(\hat{y}(t;\boldsymbol{\alpha}),y(t)) = (\hat{y}(t;\boldsymbol{\alpha}) - y(t))^2. \tag{6.3}$$

$L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ is determined by the deep network's data representation $P(\hat{y}(t;\boldsymbol{\alpha}),y(t)))$ of actual data $y(t)$. For training data of size m, the total loss function $L_{MSE}(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ is given in Equation 6.4:

$$\text{MSE Loss} : L_{MSE}(\hat{y}(t;\boldsymbol{\alpha}),y(t)) = \frac{\Sigma_{i=1}^{m} L(\hat{y}(t;\boldsymbol{\alpha})^{(i)},y(t)^{(i)})}{m}. \tag{6.4}$$

By training a deep network model, we use either a DNN or RNN to minimize empirical risk in Equation 6.2. The backpropagation training algorithm solves for $\boldsymbol{\alpha}$ in Equation 6.4 with a stochastic gradient descent procedure finding best model fit on $P(\hat{y}(t;\boldsymbol{\alpha}),y(t))$.

6.1.2 Granger Causality testing in Deep Learning Networks

Causal features can be discovered by changing loss function $L(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ in Equation 6.2 according to data representation $P(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ in deep learning networks conditioned on actual past data $y(t-j), j = 1,2,...,p$ with p lags. In the deep network, $P(\hat{y}(t;\boldsymbol{\alpha}),y(t)) = P(\hat{y}(t;\boldsymbol{\alpha})|y(t-j))$ is the conditional probability of predicting regression value $\hat{y}(t)$ or it parameterized version $\hat{y}(t;\boldsymbol{\alpha})$ for stock y.

If another stock x at time point $x(t)$ with q lagged values $x(t-k), k = 1,2,...,q$, indicates the occurrence of $y(t)$ then we create a deep network conditioned on not only $y(t-j), j = 1,2,...,p$ but also $x(t-k), k = 1,2,...,q$. Then, $P(\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta}),y(t)) = P(\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta})|y(t-j),x(t-k))$ is conditional probability of predicting regression value $\hat{y}(t)$ or it parameterized version $\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta})$ for stock y parameterized by regression parameters tensors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ belonging to deep network parameter spaces A and B respectively.

From data representations $P(\hat{y}(t;\boldsymbol{\alpha}),y(t))$ and $P(\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta}),y(t))$ defined above, we devise following Granger causality test using Equation 6.3 to predict $\hat{y}(t;\boldsymbol{\alpha})$ and $\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta})$ as dependent test variables for $y(t-j),x(t-k)$ as independent test variables.

$$\text{restricted model: } \hat{y}(t;\boldsymbol{\alpha}) = L(P(\hat{y}(t;\boldsymbol{\alpha}), y(t))) = L(P(\hat{y}(t;\boldsymbol{\alpha})|y(t-j))). \tag{6.5}$$

$$\begin{aligned} \text{unrestricted model: } \hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta}) &= L(P(\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta}), y(t))) \\ &= L(P(\hat{y}(t;\boldsymbol{\alpha};\boldsymbol{\beta})|y(t-j), x(t-k))). \end{aligned} \tag{6.6}$$

The null hypothesis of no Granger causality is rejected if and only if $x(t-k)$ has been retained along with $y(t-j)$ in the $\hat{y}(t)$ regression according to an F-test on Root Mean Squared Errors (RMSEs) between $\hat{y}(t)$ and $y(t)$. The F-test in Definition 7 [93] determines the Granger causality relation between stocks x and y where RMSE is computed for unrestricted regression as $RMSE_{ur}$ and restricted regression as $RMSE_r$.

Definition 7. *F-statistic* $= \frac{\frac{RMSE_r - RMSE_{ur}}{q-p}}{\frac{RMSE_{ur}}{n-q}}$

To compute causal features over N multivariate time series $X = \{X(t)^u\}, u \in [1,N], t \in [1,n]$ selected from N stock prices at n time points in financial markets, we repeat the F-test for every pair of stocks x and y. In each F-test, the null hypothesis is that the sample means of predictions are equal and the regression parameters $\boldsymbol{\beta}$ are zero. The alternative hypothesis is that there is significant variation between the sample means of predictions for some non-zero $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The null hypothesis is rejected if p-value on F-test has a significance level less than 0.05. If the null hypothesis is rejected, deep network features $y(t-j)$ and $x(t-k)$ Granger cause predicted output $\hat{y}(t)$ with actual stock price $y(t)$. In experiments with deep learning networks, a Granger-causal feature is represented by the causal relation $x \rightarrow y$ for stocks x and y.

6.1.3 Multivariate Regression validation with Deep Learning Networks

While we introduced theory to identify single Granger-causes in the previous subsection, in this subsection we explain the discovery of multiple Granger-causes for a given stock. Incorporating multiple Granger-causal features into the F-test allows us to improve deep learning with causal reasoning on multivariate time-dependent data.

Therefore, we discover multiple Granger-causal features with multivariate regression in an unrestricted model. The multiple Granger-causal features discovery process validates the single Granger-causal features and predictions. We extend the unrestricted model for bivariate regression in Equation 6.6 to the unrestricted model for multivariate regression as in Equation 6.7.

$$\begin{aligned} \text{multivariate unrestricted model: } \hat{y}(t;\{\boldsymbol{\alpha}^{\boldsymbol{w}}\}) &= L(P(\hat{y}(t;\{\boldsymbol{\alpha}^{\boldsymbol{w}}\}), y(t))) \\ &= L(P(\hat{y}(t;\{\boldsymbol{\alpha}^{\boldsymbol{w}}\})|y(t-j), \{x(t-k)^w\})). \end{aligned} \tag{6.7}$$

Equation 6.7 predicts $\hat{y}(t)$ by discovering statistically significant Granger-causal features $\{x^w\} \rightarrow y, w \in [1,N]$ from multivariate regression. As detailed in Algorithm 15 in the next subsection, Granger causality test of Equation 6.5 is applied to all pairs of restricted and unrestricted models that differ in one independent variable x^w discovered by bivariate regression. A feature selection procedure for multivariate regression searches candidate feature sets in the power set of the set $\{x^w\}$. The optimal feature set is determined by $\{\boldsymbol{\alpha}^{\boldsymbol{w}}\}$ with minimum RMSE $RMSE_{mv}$.

In bivariate regression, the single Granger-causal features are discovered by a DNN-based and RNN-based regression model. In multivariate regression, multiple Granger-causal features are discovered by a DNN-based regression model.

6.1.4 Deep Learning Networks based Regression Models

Algorithm 14 gives learning algorithm implementing Equation 6.5 and Equation 6.6 for loss function in Equation 6.4. The algorithm requires a multivariate time series $X = \{X(t)^u\}$ to predict regression model's causal graph G_{MSE} of Granger-causal features and corresponding regression errors $RMSE_r$, $RMSE_{ur}$, $RMSE_{mv}$ for the restricted model, the unrestricted model and the multivariate model participating in Granger causality.

Algorithm 14 executes from Line 1 to Line 16 for every pair of time series $y(t), x(t) \in X$ with lags p, q. Line 6 prepares crossvalidation data for training deep network on Line 8 which depends on the prediction $\hat{y}(t)$ from Granger causality models in Line 7. $\hat{y}(t)$ is predicted as a complex nonlinear combination of features $y(t-j)$ and $x(t-k)$ in Line 9. On Line 13 and Line 15, bivariate regression errors $RMSE_r$, $RMSE_{ur}$ are computed on actual time point $y(t)$ and predicted time point $\hat{y}(t)$. Line 11 applies F-test to discover Granger causality relations in Line 16. The null hypothesis of not finding Granger-causal features is rejected at 5% significance level. The corresponding Granger-causal graph G_{MSE} is searched on Line 19 to improve multivariate regression errors $RMSE_{mv}$ on Line 20. In Algorithm 14, while loop from Line 2 to Line 16 discovers single Granger-causal features G_{MSE} with bivariate regression, loop from Line 17 to Line 20 discovers multiple Granger-causal features C_{mv} with multivariate regression. Algorithm 14 ends on Line 21 by returning Granger-causal features G_{MSE}, C_{mv} as well as their regression errors $RMSE_r$, $RMSE_{ur}$ and $RMSE_{mv}$.

Algorithm 14 Discovery of Granger-causal features using deep learning networks

Input: Multivariate time series : $X = \{X(t)^u\}, u \in [1,N], t \in [1,n]$; Granger causality lags $p,q \in \mathbb{Z}$;
Output: Predictive model output : Bivariate Granger-causal features graph G_{MSE}; Multivariate Granger-causal features set C_{mv}; Bivariate regression errors $RMSE_r, RMSE_{ur}$ for restricted and unrestricted model; Multivariate regression errors $RMSE_{mv}$ for unrestricted model;

1: $G_{MSE} = C_{mv} = \Phi, RMSE_r = RMSE_{ur} = RMSE_{mv} = \Phi$
2: **for** $u \in [1,N]$ **do**
3: $y(t) = X(t)^u$
4: **for** $v \in [1,N]$ **and** $v \neq u$ **do**
5: $x(t) = X(t)^v$
6: Create preprocessed and lagged cross validation data $y(t-j), x(t-k)$ with lags p,q from time series $y(t), x(t), t \in [1,n]$
7: Construct restricted and unrestricted regression model on actual data $y(t), x(t)$ according to Equation 6.5 and Equation 6.6.
8: Construct MSE loss predictions $\hat{y}(t)$ from Equation 6.4 for DNN as well as RNN networks.
9: Calculate regression errors $RMSE_r$ and $RMSE_{ur}$ for each $\hat{y}(t)$ and $y(t)$.
10: From Definition 7, compute $F\text{-}statistic$ over $RMSE_r$ and $RMSE_{ur}$.
11: **if** $F\text{-}statistic > 0.05$ **then**
12: **if** model is restricted **then**
13: Update bivariate regression error, $RMSE_r[u][v] = RMSE_r$, for restricted model
14: **else**
15: Update bivariate regression error, $RMSE_{ur}[u][v] = RMSE_{ur}$, for unrestricted model
16: Update Granger-causal features, $G_{MSE}[u] = G_{MSE}[u] \cup x(t) \rightarrow y(t)$, for bivariate regression
17: **for** $u \in [1,N]$ **do**
18: $y(t) = X(t)^u$
19: Retrieve bivariate Granger-causal features $\{x(t)^w\}$ for u from G_{MSE}
20: $RMSE_{mv}[u], C_{mv}[u]$ = multivar_granger$(y(t), \{x(t)^w\}, RMSE_{ur})$ to compute multivariate regression outputs.
21: **return** $RMSE_r, RMSE_{ur}, RMSE_{mv}, G_{MSE}, C_{mv}$

Algorithm 15 called on Line 20 of Algorithm 14 gives the search procedure implementing Equation 6.7. Algorithm 15 requires unrestricted model error $RMSE_{ur}$ found for bivariate regression predicting $y(t)$ from single Granger-causal features $\{x(t)^w\}$. The causal relations discovered between $\{x(t)^w\}$ are in Granger-causal graph G_{MSE}. Algorithm 15 then returns unrestricted model error $RMSE_{mv}$ from multivariate regression as well as corresponding multiple Granger-causal features set c_{mv} discovered by multivariate regression network. For all predicted $\{X(t)^u\}$, Granger-causal feature sets C_{mv} stored on Line 20 are the optimal Granger-causal feature sets discovered across many multivariate regression networks. The loop from Line 4 to Line 19 in Algorithm 15 uses two sets of selected_causes and candidate_causes to generate and evaluate candidate Granger-causal feature sets for unrestricted model in multivariate regression. On Line 2, selected_causes are initialized to Granger-causal features $\{x(t)^w\}$ discovered in G_{MSE} of Algorithm 14. On Line 6, Cartesian product of selected_causes and bivariate Granger-causal features $\{x(t)^w\}$ generates candidate_causes. $Candidate\_RMSE_r$, $Candidate\_RMSE_{ur}$ are used to track regression errors of restricted and unrestricted models built from candidate_causes. On Line 10, initially a restricted model in multi-

Algorithm 15 Search procedure for constructing multivariate Granger-causal graphs

Input: Effect time series $y(t)$; Bivariate Granger-causal features $\{x(t)^w\}$; Bivariate Regression errors $RMSE_{ur}$ for unrestricted model
Output: Optimal Granger-causal feature set c_{mv} and multivariate regression error $RMSE_{mv}$ for unrestricted model
1: **function** MULTIVAR_GRANGER($y(t), \{x(t)^w\}, RMSE_{ur}$)
2: Initialize selected_causes to Bivariate Granger-causal features $\{x(t)^w\}$
3: $iter = 0, Candidate\_RMSE_r = Candidate\_RMSE_{ur} = \Phi$
4: **while** selected_causes $\neq \Phi$ **do**
5: $iter$ += 1
6: Generate candidate_causes, candidate_causes = $\{x(t)^w\}\times$ selected_causes, from previous iteration's selected_causes
7: Reset selected_causes to Φ in current iteration
8: **for** each candidate cause $c \in$candidate_causes **do**
9: **if** $iter == 1$ **then**
10: Set restricted model error $Candidate\_RMSE_r[c] = RMSE_{ur}[c]$
11: **else**
12: Set restricted model error $Candidate\_RMSE_r[c] = Candidate\_RMSE_{ur}[c \setminus \{x(t)^w\}]$
13: Construct multivariate unrestricted regression model on actual data $y(t)$ and $\{x(t)^w\}$ according to Equation 6.7
14: Construct MSE loss predictions $\hat{y}(t)$ from Equation 6.4 for DNN networks.
15: Calculate regression error $Candidate\_RMSE_{ur}[c]$ for all $\hat{y}(t)$ and $y(t)$.
16: From Definition 7, compute $F\text{-}statistic$ over $Candidate\_RMSE_r[c]$ and $Candidate\_RMSE_{ur}[c]$.
17: **if** $F\text{-}statistic > 0.05$ **then**
18: Update Granger-causal features: selected_causes = selected_causes $\cup$ c
19: **end while**
20: Among unrestricted models $Candidate\_RMSE_{ur}$, find optimal Granger-causal feature set c_{mv} with minimum multivariate regression error $RMSE_{mv}$
21: **return** $RMSE_{mv}, c_{mv}$
22: **end function**

variate regression is assumed to be the same as the unrestricted model in bivariate regression. Later as the loop from Line 4 to Line 19 crosses more than one iteration as tracked by counter $iter$, the restricted model is evaluated against Granger-causal features $c \setminus \{x(t)^w\}$ on Line 12 while the unrestricted model is evaluated against Granger-causal features c on Line 13. In any giver iteration $iter$, the restricted and unrestricted models differ by only one of the Granger-causal features present in $\{x(t)^w\}$. The multivariate regression error $Candidate\_RMSE_{ur}$ is computed for each candidate c at Lines 13-15. If the corresponding F-statistic is greater than a predefined threshold on Line 17, then the candidate c is found to be a legitimate Granger-causal feature for subsequent processing with multivariate regression. Such a c is updated to selected_causes on Line 18. For every new iteration $iter$, selected_causes are reset to the empty set on Line 7 immediately after being used to generate candidate_causes on Line 6. This loop convergence condition ensures that larger Granger-causal feature sets are generated across iterations. On convergence, no further selected_causes are available for processing. Algorithm 15 terminates the search procedure by returning the optimal Granger-causal feature set c_{mv} that minimizes multivariate regression error $RMSE_{mv}$.

6.2. Experiments

Table 6.1: Companies Listing

Abbreviation	Company Name	Abbreviation	Company Name
AAPL	Apple Inc.	MCD	McDonald's Corporation
ABT	Abbott Laboratories	MSFT	Microsoft Corporation
AEM	Agnico Eagle Mines Limited	ORCL	Oracle Corporation
AFG	American Financial Group, Inc.	WWD	Woodward, Inc.
APA	Apache Corporation	T	AT&T Inc.
CAT	Caterpillar Inc.	UTX	United Technologies Corporation

In this section we discuss the empirical validation of Granger-causal features in deep learning networks regression models. Table 6.1 lists the stocks from different financial sectors in Standard & Poor's 500 - a stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ. The stocks daily closing prices were obtained from Yahoo Finance website [1]. The data is obtained for a period of 21 years from 26-07-1996 to 25-07-2017.

The regression model's feature learning is determined by deep network structure weights $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\{\boldsymbol{\alpha}^{\boldsymbol{w}}\}$ with MSE loss function. Deep network structure is designed to minimize bivariate/multivariate regression errors and maximize significant Granger causes in the unrestricted model. We treat the regression model as a time-dependent data-based model with causal lags p, q set to a default value of 200 days. 5285 days of time points are used to create the crossvalidation data. Each data record has delayed prices time series predicting current price of a given stock. For fair comparison of baseline models, we split 30% of crossvalidatiton data into testing data while remaining 70% of crossvalidatiton data is taken to be training data.

On bivariate data, we treat regression modelling problem as a discriminative learning problem in DNNs as well as a sequence learning problem in RNNs to show that discovered Granger-causal features are not specific to a given network structure. On multivariate data, we treat regression modelling problem as a discriminative learning problem in DNNs to validate generalization capability of proposed feature discovery procedure. The regression errors for discovering Granger-causal features are also compared with those from a Autoregressive Integrated Moving Average (ARIMA) regression model. A grid search procedure is used to select ARIMA training parameters. Number of training epochs in DNN is set to a default value 50 over a total of 12 stocks. The DNN has three hidden layers consisting of dense activation units and dropout regularization units. It is

[1] `https://finance.yahoo.com/`

Table 6.2: RMSEs with MSE loss for bivariate regression. DNN is selected as the best network structure for Granger causality.

Abbreviation	ARIMA	LSTM	GRU	DNN
AAPL	0.807	1.449	1.475	0.504
ABT	0.748	0.461	0.469	0.626
AEM	1.643	1.115	1.107	0.143
AFG	0.795	0.580	0.588	0.485
APA	2.795	1.558	1.520	0.145
CAT	1.254	1.474	1.452	0.106
MCD	0.319	0.981	0.994	0.425
MSFT	1.555	0.597	0.606	0.361
ORCL	0.190	0.521	0.520	0.497
T	0.786	0.335	0.339	0.078
UTX	0.209	1.110	1.113	0.297
WWD	0.237	0.817	0.819	0.311
t-test	1.24×10^{-2}	2.21×10^{-4}	1.89×10^{-4}	Base

implemented in Keras [2] – a Tensorflow based API for deep learning. All time series are subject to min-max normalization before training.

We experiment with two variants of RNN with Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) activation units. The number of training epochs in RNN is set to a default value 15. The LSTM has one hidden layer consisting of LSTM activation unit with 50 neurons. The GRU has three hidden layers consisting of GRU activation units with 25 neurons. Dropout units are the regularization units. LSTM as well as GRU state is reset after each training epoch. The LSTM and GRU are trained for 200 time steps - one record at a time - over lagged data. The time series data is differenced and scaled to a range of [-1,1]. For multivariate regression, all the identified single Granger-causal features are used as input. On multivariate testing data, regression values are predicted one time step at a time.

6.2.1 Single Granger-causes validation

For each company's price time series, autoregression models RMSEs are reported in Table 6.2. From t-test statistics in Table 6.2, we find DNN generally has better performance than competitive models. So we choose DNN as the regression model for discovering Granger-causal features with bivariate regression in Table 6.3 as well as multivariate regression in Figure 6.1. For experimental validation of our algorithms, we also report Granger-causal features discovered by a GRU model in Table 6.4.

Table 6.3 and Table 6.4 report RMSEs for restricted model $RMSE_r$ and unrestricted model $RMSE_{ur}$. $RMSE_{ur}$ is consistently lower than $RMSE_r$ for Granger causality models given in Equation 6.5 and Equation 6.6 respectively. Each row in Table 6.3

[2] `https://www.tensorflow.org/api_docs/python/tf/contrib/keras`

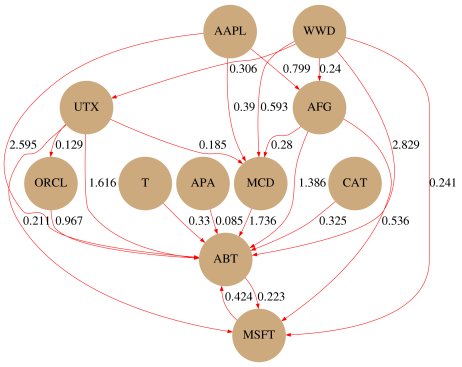
Table 6.3: RMSEs with MSE loss for Granger-causal feature discovery. The rows show causal relations with the restricted model and the unrestricted model RMSEs $RMSE_r$ and $RMSE_{ur}$ in bivariate regression with DNN.

Causal Relation			$RMSE_r$ Restricted model (DNN without causes)	$RMSE_{ur}$ Unrestricted model (our model with single cause)	Causal Relation			$RMSE_r$ Restricted model (DNN without causes)	$RMSE_{ur}$ Unrestricted model (our model with single cause)
AAPL	→	ABT	0.626	0.198	AAPL	→	AFG	0.485	0.293
AFG	→	ABT	0.626	0.191	WWD	→	AFG	0.485	0.396
APA	→	ABT	0.626	0.477	AAPL	→	MCD	0.425	0.315
CAT	→	ABT	0.626	0.372	AFG	→	MCD	0.425	0.418
MCD	→	ABT	0.626	0.261	UTX	→	MCD	0.425	0.365
MSFT	→	ABT	0.626	0.501	WWD	→	MCD	0.425	0.353
ORCL	→	ABT	0.626	0.362	ABT	→	MSFT	0.361	0.295
T	→	ABT	0.626	0.535	AFG	→	MSFT	0.361	0.249
UTX	→	ABT	0.626	0.271	UTX	→	MSFT	0.361	0.297
WWD	→	ABT	0.626	0.184	WWD	→	MSFT	0.361	0.183
WWD	→	UTX	0.297	0.219	UTX	→	ORCL	0.497	0.202
t-test			1.17×10^{-6}	Base	t-test			1.17×10^{-6}	Base

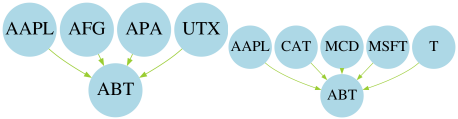
Table 6.4: RMSEs with MSE loss for Granger-causal feature discovery. The rows show causal relations with the restricted model and the unrestricted model RMSEs $RMSE_r$ and $RMSE_{ur}$ in bivariate regression with RNN.

Causal Relation			$RMSE_r$ Restricted model (RNN without causes)	$RMSE_{ur}$ Unrestricted model (our model with single cause)	Causal Relation			$RMSE_r$ Restricted model (RNN without causes)	$RMSE_{ur}$ Unrestricted model (our model with single cause)
ABT	→	AAPL	1.475	0.403	AFG	→	APA	1.529	0.788
AFG	→	AAPL	1.475	0.781	MCD	→	APA	1.571	0.944
MCD	→	AAPL	1.475	0.936	MSFT	→	APA	1.522	0.859
MSFT	→	AAPL	1.475	0.851	ORCL	→	APA	1.551	0.732
ORCL	→	AAPL	1.475	0.726	T	→	APA	1.527	0.741
T	→	AAPL	1.475	0.734	UTX	→	APA	1.522	1.021
UTX	→	AAPL	1.475	1.012	WWD	→	APA	1.526	0.846
WWD	→	AAPL	1.475	0.839	ABT	→	CAT	1.445	0.474
ABT	→	AEM	1.107	0.458	AFG	→	CAT	1.445	0.918
ABT	→	AFG	0.588	0.303	MSFT	→	CAT	1.445	1.001
ABT	→	APA	1.545	0.407	ORCL	→	CAT	1.444	0.853
ABT	→	MCD	0.994	0.382	T	→	CAT	1.445	0.863
ORCL	→	MCD	0.994	0.687	WWD	→	CAT	1.444	0.986
T	→	MCD	0.994	0.695	ABT	→	UTX	1.113	0.421
ABT	→	ORCL	0.521	0.257	ORCL	→	UTX	1.113	0.756
ABT	→	T	0.338	0.231	T	→	UTX	1.114	0.765
ABT	→	MSFT	0.606	0.256	ABT	→	WWD	0.821	0.397
t-test			3.21×10^{-11}	Base	t-test			3.21×10^{-11}	Base

and Table 6.4 shows pairwise causal relations and their RMSEs. From t-test p-value statistic comparing RMSEs with and without Granger-causal features in Table 6.3 and Table 6.4, we conclude that unrestricted model shows non-trivial reduction in RMSE compared to restricted model for any random pair of stocks involved in Granger causality.



(a) Granger-causal graph with F-test statistics computed on RMSEs $RMSE_r$,$RMSE_{ur}$ in the restricted and unrestricted models for bivariate regression



(b) Two sets of Granger-causal features discovered by Algorithm 15 for predicting stock price ABT with multivariate regression. They reduce restricted model $RMSE_r$ from 0.626 to unrestricted model $RMSE_{mv}$ 0.141 and 0.178 respectively.

Figure 6.1: Granger-causal features, F-statistics on RMSEs $RMSE_r$,$RMSE_{ur}$ and multivariate regression RMSEs $RMSE_{mv}$ for the unrestricted model with DNN. The edge directions indicate the causal relations between pairs of stocks and the edge weights show the corresponding F-test statistic given in Definition 7.

Figure 6.1(a) represents Granger-causal features discovered from bivariate regression as a causal graph between time series of stock prices represented by vertices where F-test statistics represented by edges show the strength of Granger causality.

Thus Table 6.3 and Table 6.4 validate our proposal to use Granger causality in feature selection for deep networks based regression models. We also observe that the proposed feature discovery process and supervised learning process are robust to any particular deep network structure.

6.2.2 Multiple Granger-causes validation

Figure 6.1(a) shows the Granger-causal graph with directed weighted edges that are outcomes of Definition 7. It indicates Granger-causal relations discovered for bivariate regression. The edge weights are F-test statistics for all the unrestricted models that reduce RMSEs in bivariate regression. For example, the causal relation $UTX \rightarrow ORCL$ indicates that UTX causes ORCL or ORCL is caused by UTX with F-test statistic 0.129. This relation has been selected in the Granger-causal graph because the unrestricted model including UTX prices in ORCL price prediction leads to a RMSE reduction from 0.497 to 0.202 according to Table 6.3. Figure 6.1(a) shows all the causalities identified on the training data. We do not assume causalities change at every time point. From

Figure 6.1(a), we not only can identify causal features but also indicate the strength of causality.

Figure 6.1(b) shows the top ranked Granger-causal features discovered by Algorithm 15 from Figure 6.1(a). These causal features are suitable for multivariate regression. In Figure 6.1(a), vertices like APA without Granger-causes, ORCL and MSFT with one and two Granger-causes result in no output from Algorithm 15. For vertices like ABT with non-zero single Granger-causes, Algorithm 15 identifies multiple Granger-causes. For ABT, Algorithm 15 outputs a total of 69 Granger-causes which reduces RMSE $RMSE_{mv}$ in multivariate regression models from $RMSE_r = 0.626$ in the restricted model to $RMSE_{mv} \in [0.141, 0.541]$ in the unrestricted model. In Figure 6.1(b), the multivariate Granger-cause {AAPL, AFG, APA, UTX} has regression error of $RMSE_{mv} = 0.141$ while {AAPL, CAT, MCD, MSFT, T} has regression error of $RMSE_{mv} = 0.178$ in the unrestricted model. Of the 69 causes, the longest but not optimal Granger-causes are found to be {AAPL, APA, CAT, MCD, ORCL, T, UTX, WWD} with $RMSE_{mv} = 0.162$ and {AAPL, APA, CAT, MCD, MSFT, T, UTX, WWD} with $RMSE_{mv} = 0.174$ in the unrestricted model. We also find two Granger-causes of length 8, six Granger-causes of length 7 and ten Granger-causes of length 6. From Figure 6.1(b), we observe that multivariate regression on Granger-causal features results in a better unrestricted model than bivariate regression on Granger-causal features. In bivariate regression as well as multivariate regression, while F-test statistics on RMSEs validate our feature selection on regression errors, t-test statistics on RMSEs support our model validation on regression errors.

6.3. Findings Summary

In this chapter we derive data representation about the structure of a complex system from regression networks that measure the information learnt by deep networks. With multiple regression we measure relevant information operating at multiple scales within the complex system. For our chosen loss functions, we find Granger-causal structures as optimal solutions for the empirical risk of the complex system in the sense of decision-theoretic rationality. The empirical risk of regression modelling is formulated in terms of the mean squared error loss function in DNNs and RNNs.

From an application standpoint, our deep network regression models capture temporal dependencies over multivariate time series data. We treat time series analysis as a discriminative learning regression problem with a DNN as well as a sequence learning regression problem with a RNN. Then we use deep networks regression to

extract the Granger-causal features in time series. The DNN's learning process then reduces overall variance in the regression errors to provide more confidence in the use of model predictions. In general, the DNN's loss function learns moments and cumulants of the time dependent data distributions in the regression model. The Granger-causal structures giving direct and indirect cause-effect relations in an application scenario of stock market analysis.

Our learning model can be further customized to temporal continuity and information uncertainty dynamics in the underlying multivariate time series data distributions. Here adversarial data would consist of anticausal features exploiting irregularities, dissimilarities and anomalies in the input data distribution. In future research investigations we shall integrate causal feature learning with adversarial learning in the context of explainable artificial intelligence.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1. Research Summary

Our aim with this research was to improve the robustness of deep learning models with game-theoretic modelling. Through a series of Stackelberg games, we generated data about how adversaries manipulate training data to mislead CNN classification results. With this knowledge, we were able to augment the training data with the manipulated data within an adversarial training process to produce a series of secure CNNs.

We defined both two-player and multiplayer Stackelberg games. The two-player games were held between a data miner in charge of designing a learning algorithm and an intelligent adversary in charge of designing an adversarial algorithm. The games mimicked the parameter tuning actions of a machine learning algorithm. The adversary had to modify its attack strategy to avoid detection by the learning algorithm, while the learning algorithm had to update its model based on newly discovered threats from the adversary. Equilibrium was reached when neither the learner nor the adversary had further incentive to play the game, and the games ended with payoffs to each player based on their objectives and actions. These objectives could be further improved by identifying the key decision-theoretic ingredients in a game, such as: (a) which players were the decision-makers; (b) the actions or series of actions taken by players; and (c) determining how different payoffs motivated player actions.

In terms of the adversarial algorithm design, we needed to address issues such as: (i) adversarial data generation in a single iteration of the game; (ii) adversarial

attack scenarios in multiple iterations of the game; and (iii) adversarial performance measurement when the training data label is known and the learner's prediction label is generated to minimize the adversary's cost and maximize the classification error.

We also formulated stochastic optimization problems for adversarial learning with two-player sequential games, two-player stochastic games, and multiplayer stochastic games over deep learning networks. By producing incremental (and additive as well as non-additive) changes to training data distributions in the games, we were able to find the local optima and use them as manipulations by adversary(s). In each iteration of the game, the adversary's strategy spaces and attack scenarios are determined by a payoff function with both evolutionary and variational attack parameters. The optimal attack policy is revealed by searching for the attack parameters in the original data as well as in the encoded data. The experiments demonstrate the correctness and performance of the proposed adversarial algorithms.

In labelling the generated adversarial data, our aim was to improve the supervised estimations of the adversary rather than the learner. While we mostly generated illegitimate adversarial data that looked like legitimate learner data, we were also interested in legitimate adversarial data that looked like illegitimate learner data. However, legitimate adversarial data that looked like legitimate learner data was not of interest.

To test adversarial learning hypotheses on the class labels and the decision boundaries in classification loss functions, we also propose adversarial payoff functions. The proposed payoff functions optimize the search for data manipulations on a original pixel data space as well as a latent data space representing pixel distributions by a Gaussian mixture model. The payoff functions were then optimised by the parameter settings in simulated annealing, variational learning and generative learning algorithms. The payoff functions are optimized by the parameter settings in simulated annealing and generative learning algorithms. The CNN's misclassification performance at the time of Nash equilibrium was measured in terms of t-statistics hypothesised over recall, true positive rate, and the f_1-score of targeted class labels.

We experimented with adversarial payoff functions over randomised strategy spaces by changing the Stackelberg game formulation. Here, the attack scenarios over the strategy spaces determined the convergence criteria of Stackelberg games over multi-label datasets. In Nash equilibrium, the game converged on adversarial manipulations that affect testing performance across targeted labels in both two-label and multi-label classification models. The results led us to a proposal for a secure learner that is immune to that type of adversarial attack, and an empirical analysis confirms that this

classification model is significantly more robust than a traditional CNN or GAN under attack by an adversary.

From our experiments and analyses, we are confident that our proposed adversarial learning algorithms create classifiers that are robust to the targeted attacks. However, in future, we plan to experiment with whitebox attacks. These types of attacks involve deep generative networks and will require multiplayer games and payoff functions suited to interclass discrimination. In these types of scenarios, attack policies can be formulated as multiple EM-like steps that attempt to estimate an adversary's cost functions (sparse as well as dense) as a blackbox attack and a learner's loss functions as a whitebox attack. Introducing metric learning models that measure image representation and information divergence between legitimate and illegitimate data will allow us to explore whitebox substitution attacks on the expected behaviour of loss functions in multi-label classification networks.

In future work, we shall also explore dependence between randomization in our adversarial manipulations and optimization in our game formulation. At present, the game-theoretic stochastic optima (solving for adversarial data) are determined by the convergence of the adversarial cost function rather than a classification cost function. However, multi-label classification cost functions in a multiplayer strategy space of pure strategies as well as mixed strategies could be another fruitful avenue of research. Assuming a whitebox attack scenario on a CNN classifier would mean we could guide the parameter settings in a genetic algorithm and an SA algorithm into application-dependent adversarial data distributions.

An attack scenario with interactions between multiple cooperating adversaries gives rise to Coalitional Games in the one-leader-multiple-follower style with either single or multiple adversarial objectives. We are also interested in randomization strategies for robust optimization in Multiplayer Games that can be decomposed into Prediction Games or Stackelberg Games. Here, some of the relevant game-theoretic formulations are to be found in literature on Evolutionary Games, Matrix Games, Robust Games, Fuzzy Games, Markov Games and Bayesian Games. The dimensionality of multi-label data in these kinds of games can be tackled by training an adversarial algorithm for dataflow and control-flow parallelisation with multiple processing units. Guided search operators in evolutionary learning might lead to attack scenarios with parallelisation for stochastic optimization in adversarial step magnitude and direction estimations.

Further, we might also experiment with various classification functions by changing the deep learning architecture, or experiment with multi-objective optimization methods

by changing the evolutionary operators in the fitness function evaluations of stochastic optimization and constraint-driven games. It is possible that our variational adversaries with Gaussian mixture models could be improved with customised probabilistic models, such as Multinomial Mixture Models and Mixture Density Networks for image data representation. User validation criteria of the generated adversarial data can be represented by deep networks such as the Spatial Translation Network.

In Chapter 6, we discussed the information-theoretic dependence between the adversary's conditional data likelihood estimation and the classifier's empirical risk minimisation when training a deep network-based regression model. Our regression models augment and discover Granger-causal features that are able to significantly improve multivariate regression performance on multivariate financial time series data. We also constructed Granger-causal graphs to capture the temporal dependencies in multivariate data. With real stock market data, we demonstrated that our theoretical model significantly outperforms the existing deep learning-based regression models.

As future work on the discovery of Granger-causal features, we plan to combine multiple data sources to extract regularised features for cost-sensitive concept learning and big data pattern detection. The proposed network architectures and loss functions can be extended for nonlinear algorithm-oriented approaches to robust regression, and the sensitivity of our custom loss function could be improved with patterns constructed for application-dependent model selection. Here, time-frequency analysis methods and deep generative models might be helpful for feature engineering and learning generalisations in specific application domains.

Another focus of future work could be causal inference formulations that connect an adversarial learning environment to a targeted learner. For example, combining formal concept analysis with Granger causality could help us to design a better deep learning environment. Combining deep learning features with rule mining concepts may allow us to explore time-dependent descriptive modelling for time series prediction. In turn, these novel causal inference mechanisms could lead to: alternative statistical significance tests for supervised feature discovery; new causal graphs to improve the cost-sensitive concept learning of our custom loss functions; or better estimations for discriminative deep regression learning with custom data representations and custom loss functions.

7.2. Applications of Game Theoretical Adversarial Learning Models

In Chapter 4, we addressed attack settings in multi-label classifiers with a two-player sequential game. Further, by changing the game formulation, we experimented with adversarial payoff functions over randomised strategy spaces. In future, we plan to extend this work to machine learning applications with multiple adversarial objectives, such as protocol verification and performance modelling with complex systems.

The two-player multi-label version of the game can be generalised to a multi-player multi-label game by associating one conditional generative network with each adversary. The adversarial cost term in the corresponding fitness function would need to be determined through new game formulations. However, according to game theory, once the fitness function was identified, the interactions between the adversary's costs and classification costs would allow us to introduce multiple adversarial objectives.

We also plan to investigate the more challenging multiplayer game scenarios where adversaries simultaneously attack multiple labels. In this multiplayer adversarial learning problem, we want to simulate manipulations that transform a targeted positive label into any one of many negative labels. The successful attack scenarios in a strategy space of this type would then inform multiplayer games with mixed strategies that have two or more labels as manipulation targets given a single learner. In these scenarios, it is likely that the randomization of strategies and payoffs in the game formulation would affect the weight regularizations and the decision boundaries of the learner.

7.2.0.1 Multiplayer Games

Game-theoretic players start with beliefs about their opponent's strategies, and each player's best response in a game is subsequently assessed in relation to the best strategy of their opponent. In the context of game-theoretic formulations, payoff functions measure these player-driven optimizations to improve training and inference in machine learning. Payoff functions are effectively mathematical definitions that quantify the degree of each player's preferences toward the actions they have available to them in the game. In machine learning, they also explain the impact of uncertainty with reference to a distribution of outcomes, and, in the sense of decision-theoretic rationality, payoff functions maximise the expected utility or the average utility for each player in the game.

In the following paragraphs, we review some of the ideas that emerged from the stochastic optimizations in our multiplayer games. These games involved one leader and multiple follower formulations with both single or multiple objectives for the adversary(s), and each is suitable for mathematical programming with multi-label classification problems. Typically, these game optimization designs solve:a) maximisation problems over the adversary's payoff function; and b) minimisation problems over the learner's payoff function. In the following we survey maximisation problems in adversary's payoff functions expressed in terms of the improvements to the strategy spaces in adversarial cost functions.

Normal Form Game and Extensive Form Game The standard Normal Form or Matrix Form representations for game explicitly define players, actions, and their payoffs. Extensive Form representations have timing information represented by trees of possible actions in the game at various points in time. These trees can also keep track of what each player knows when he or she makes each decision. The decisions can be made either sequentially or non-sequentially.

At Nash equilibrium, every player must choose their optimal response to maximise their chance of winning the game given the information they know about the other players' actions. Over time, various dynamics will underlie the setting of the game. These dynamics: (i) must be learned through machine learning; and (ii) must evolve over time according to a stochastic optimization.

Extensive form games make the temporal structure of the game explicit by defining the players, actions, choice nodes, terminal nodes, and successor functions for every edge in the game tree. The utility functions are then defined for each player on the terminal nodes. A subgame corresponds to a subtree in the game tree. The Nash equilibrium in the subgame is called subgame perfect Nash equilibrium, which is computed by an optimization algorithm like Backward Induction.

In imperfect information extensive form games, each player's choice nodes are partitioned into information sets such that players cannot distinguish between choice nodes in the same information set. Imperfect information extensive-form games can be expressed as Normal Form games.

Pure Strategy Game and Mixed Strategy Game The best response is the response that increases the player's payoff given the predicted actions of all the other players. At Nash equilibrium, players must look for stable action profiles or a 'pure strategy' for all

the players given the best response by each player. In a pure strategy game, a player also has the opportunity to reason about everything they have seen in the game before acting.

When pure strategy Nash equilibrium does not exist, 'pure strategy' Nash equilibria is contrasted with games giving 'mixed strategy' of best responses. A mixed strategy for a player can be thought of as a probability distribution over the actions available to that player. In this context, a player's pure strategy is then a special case of a player's mixed strategy where only one action can be taken with a positive probability. By comparison, a mixed strategy supposes that the player could take more than one action with a positive probability.

The set of actions with positive probability are called in support of the mixed strategy. With mixed strategy support for each player, the definition of the payoff function may be extended to replace the deterministic actions of each player. Mixed strategies are probabilistic mixtures of pure strategies, while mixed strategies are defined as randomizations of pure strategies, and behavioural strategies are randomizations of information sets of mixed strategies.

Bayesian Game Bayesian Games are a new form of game that auctions reasoning with uncertain information. The auctions are designed to account for the intangible aspects of a game, such as supply-demand, the social value of a commodity in a market at a given time, etc.

In an auction, the players are not sure about what a good is worth to the other participants in the auction, but this information is critical to strategic reasoning in the auction. Thus, Bayesian Games are a mathematical formalism for modelling this kind of uncertainty about a player's payoff function. No player in the game has full information of all the players, actions, or payoffs motivating the game. However, all players do have well-defined beliefs about what is possible in the world, which are modelled as posterior probability distributions that are derived from conditioning a common public prior on individual private signals. The individual private signals for each player are defined by a partition structure of equivalence classes over the games, and the common prior is decided by nature and the ground truth.

Thus, players in a Bayesian Game must reason about their opponents without fully knowing their opponent's best strategies. In the game, each player only has access to common prior, their own equivalence classes, and their opponents equivalence classes. Hence, a Bayesian Game can be defined as a probability distribution over multiple normal form games, all of which share the same players and action sets but not the same

payoffs.

In Bayesian games, players also reason with uncertain information about the payoff functions using epistemic mathematical notion. Epistemic notation is supposed to capture everything related to a player - their private signals, their posterior beliefs, etc. Bayesian game strategies are specified in terms of the players, actions, probability distribution over strategy types and player's utility functions over finite sets.

Bayesian equilibrium is a Nash equilibrium solution concept for Bayesian games. Bayesian equilibrium determines the best responses for maximizing the expected utility for each player, which is calculated from the actions and epistemic types of the other players. In a pure strategy Bayesian game, each player picks an action for each epistemic type. In a mixed strategy Bayesian game, each player picks a probability distribution over all actions expressed as a function of all players' epistemic types. Thus, Bayesian equilibrium moves us one step closer to real-world applications with uncertainty as to strategies and payoffs.

Coalitional Game Games can be categorised into non-cooperative games and coalitional games based on mathematical notions of competition and coordination. The constant-sum and variable-sum games in our thesis are non-cooperative games.

The basic modelling unit in a coalitional game is a group of players acting together; individual choices are disregarded. The focus of the study then becomes one of how the coalition can improve their payoff by working together. The coalition is assigned a single utility value, which is distributed among the members of the coalition, and fairness is defined by the Shapley value. The idea of fairness is that the members should receive a share of the payout that is proportional to their marginal contribution. The computational problem then is to design a weighting system for the contribution of members to the payoffs in relation to the utility value. Based on the concept of Shapley value, games can model complex multilateral bargaining and coalition formation strategies without specifying the particulars of a normal form or extensive form game.

Therefore, depending on whether the strategy space consists of pure strategies or mixed strategies over normal form games or extensive form games, a multiplayer game can be modelled as one or more of a set of non-cooperative games, Bayesian games, or coalitional games. Further extensions toward robust optimizations for game-theoretic adversarial learning with fuzzy games, evolutionary games, prediction games and robust games would involve the use of computational algorithms and models found in algorithmic and stochastic game theory.

The above summarises our research contributions and our projections for future research.

BIBLIOGRAPHY

[1] D. ADLER, *Genetic algorithms and simulated annealing: a marriage proposal*, in IEEE International Conference on Neural Networks, March 1993, pp. 1104–1109 vol.2.

[2] E. A.E., *Evolutionary algorithms and constraint satisfaction: Definitions, survey, methodology, and research directions*, in Kallel L., Naudts B., Rogers A. (eds) Theoretical Aspects of Evolutionary Computing., Springer, Berlin, Heidelberg, 2001.

[3] N. AKHTAR AND A. S. MIAN, *Threat of adversarial attacks on deep learning in computer vision: A survey*, IEEE Access, 6 (2018), pp. 14410–14430.

[4] J. V. ALAIN BENSOUSSAN, JENS FREHSE, *Nash and stackelberg differential games*, Chinese Annals of Mathematics, Series B, 33 (2012), pp. 317–332.

[5] E. ALBA AND M. TOMASSINI, *Parallelism and evolutionary algorithms*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 443–462.

[6] H. ALKHATIB, P. FARABOSCHI, E. FRACHTENBERG, H. KASAHARA, D. LANGE, P. LAPLANTE, A. MERCHANT, D. MILOJICIC, AND K. SCHWAN, *IEEE CS 2022 report (draft)*, tech. rep., IEEE Computer Society, February 2014.

[7] T. ALPCAN, B. I. P. RUBINSTEIN, AND C. LECKIE, *Large-scale strategic games and adversarial machine learning*, in 2016 IEEE 55th Conference on Decision and Control (CDC), 2016.

[8] P.-O. AMBLARD AND O. J. MICHEL, *The relation between granger causality and directed information theory: A review*, Entropy, 15 (2012), pp. 113–143.

[9] M. ANCONA, C. ÖZTIRELI, AND M. H. GROSS, *Explaining deep neural networks with a polynomial time algorithm for shapley value approximation*, in Proceed-

ings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019, pp. 272–281.

[10] N. ANCONA, D. MARINAZZO, AND S. STRAMAGLIA, *Radial basis function approach to nonlinear granger causality of time series*, Phys. Rev. E, 70 (2004), p. 056221.

[11] A. ARNOLD, Y. LIU, AND N. ABE, *Temporal causal modeling with graphical granger methods*, in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 66–75.

[12] A. ATTAR, R. M. RAD, AND R. E. ATANI, *A survey of image spamming and filtering techniques*, Artificial Intelligence Review, 40 (2013), pp. 71–105.

[13] T. BACK, F. HOFFMEISTER, AND H.-P. SCHWEFEL, *A survey of evolution strategies*, in Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, pp. 2–9.

[14] D. BAEHRENS, T. SCHROETER, S. HARMELING, M. KAWANABE, K. HANSEN, AND K.-R. MÜLLER, *How to explain individual classification decisions*, J. Mach. Learn. Res., 11 (2010), p. 1803,Äì1831.

[15] M. T. BAHADORI AND Y. LIU, *Granger causality analysis in irregular time series*, in Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, 2012, pp. 660–671.

[16] ——, *An examination of practical granger causality inference*, in Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 467–475.

[17] S. BALUJA AND I. FISCHER, *Learning to attack: Adversarial transformation networks*, in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[18] S. BANDARU, A. H. C. NG, AND K. DEB, *Data mining methods for knowledge discovery in multi-objective optimization: Part A - survey*, Expert Syst. Appl., 70 (2017), pp. 139–159.

[19] ——, *Data mining methods for knowledge discovery in multi-objective optimization: Part B - new developments and applications*, Expert Syst. Appl., 70 (2017), pp. 119–138.

[20] S. BANDYOPADHYAY, S. K. PAL, AND C. MURTHY, *Simulated annealing based pattern classification*, Information Sciences, 109 (1998), pp. 165 – 184.

[21] M. BARRENO, B. NELSON, R. SEARS, A. D. JOSEPH, AND J. D. TYGAR, *Can machine learning be secure?*, in Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS '06, New York, NY, USA, 2006, ACM, pp. 16–25.

[22] A. BARTH, B. I. RUBINSTEIN, M. SUNDARARAJAN, J. C. MITCHELL, D. SONG, AND P. L. BARTLETT, *A learning-based approach to reactive security*, International Conference on Financial Cryptography and Data Security, (2010), pp. 192–206.

[23] T. BASAR AND J. MOON, *Riccati equations in nash and stackelberg differential and dynamic games*, IFAC-PapersOnLine, 50 (2017), pp. 9547 – 9554.
20th IFAC World Congress.

[24] O. BASTANI, Y. IOANNOU, L. LAMPROPOULOS, D. VYTINIOTIS, A. V. NORI, AND A. CRIMINISI, *Measuring neural net robustness with constraints*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS,Äô16, Red Hook, NY, USA, 2016, Curran Associates Inc., p. 2621,Äì2629.

[25] V. BEHZADAN AND A. MUNIR, *Vulnerability of deep reinforcement learning to policy induction attacks*, in Machine Learning and Data Mining in Pattern Recognition - 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings, 2017, pp. 262–275.

[26] D. BERTHELOT, T. SCHUMM, AND L. METZ, *BEGAN: boundary equilibrium generative adversarial networks*, CoRR, abs/1703.10717 (2017).

[27] D. P. BERTSEKAS, *Stochastic optimization problems with nondifferentiable cost functionals*, J. OPTIM. THEORY APPL, 12 (1973), pp. 218–231.

[28] H.-G. BEYER AND H.-P. SCHWEFEL, *Evolution strategies - A comprehensive introduction*, Natural Computing, 1 (2002), pp. 3–52.

[29] B. BIGGIO, I. CORONA, D. MAIORCA, B. NELSON, N. ŠRNDIĆ, P. LASKOV, G. GIACINTO, AND F. ROLI, *Evasion attacks against machine learning at test time*, in Machine Learning and Knowledge Discovery in Databases, H. Blockeel,

K. Kersting, S. Nijssen, and F. Železný, eds., Berlin, Heidelberg, 2013, Springer Berlin Heidelberg, pp. 387–402.

[30] B. BIGGIO, G. FUMERA, I. PILLAI, AND F. ROLI, *A survey and experimental evaluation of image spam filtering techniques*, Pattern Recogn. Lett., 32 (2011), pp. 1436–1446.

[31] B. BIGGIO, G. FUMERA, AND F. ROLI, *Multiple classifier systems for adversarial classification tasks*, in Multiple Classifier Systems, J. A. Benediktsson, J. Kittler, and F. Roli, eds., Berlin, Heidelberg, 2009, Springer Berlin Heidelberg, pp. 132–141.

[32] B. BIGGIO, G. FUMERA, AND F. ROLI, *Multiple classifier systems for robust classifier design in adversarial environments*, Journal of Machine Learning and Cybernetics, 1 (2010), p. 27–41.

[33] B. BIGGIO, G. FUMERA, AND F. ROLI, *Multiple classifier systems for robust classifier design in adversarial environments*, International Journal of Machine Learning and Cybernetics, 1 (2010), pp. 27–41.

[34] ——, *Security evaluation of pattern classifiers under attack*, IEEE transactions on knowledge and data engineering, 26 (2014), pp. 984–996.

[35] B. BIGGIO, G. FUMERA, P. RUSSU, L. DIDACI, AND F. ROLI, *Adversarial biometric recognition: A review on biometric system security from the adversarial machine learning perspective*, IEEE Signal Processing Magazine, 32 (2015), pp. 31–41.

[36] B. BIGGIO, B. NELSON, AND P. LASKOV, *Poisoning attacks against support vector machines*, (2012), pp. 1467–1474.

[37] ——, *Poisoning attacks against support vector machines*, in Proceedings of the 29th International Coference on International Conference on Machine Learning, ICML'12, USA, 2012, Omnipress, pp. 1467–1474.

[38] B. BIGGIO, I. PILLAI, S. ROTA BULO, D. ARIU, M. PELILLO, AND F. ROLI, *Is data clustering in adversarial settings secure?*, in Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, AISec '13, New York, NY, USA, 2013, ACM, pp. 87–98.

[39] B. BIGGIO AND F. ROLI, *Wild patterns: Ten years after the rise of adversarial machine learning*, in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, New York, NY, USA, 2018, ACM, pp. 2154–2156.

[40] C. M. BISHOP, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[41] W. BRENDEL, J. RAUBER, AND M. BETHGE, *Decision-based adversarial attacks: Reliable attacks against black-box machine learning models*, in International Conference on Learning Representations, 2018.

[42] A. BRESSAN, *Noncooperative differential games*, Milan Journal of Mathematics, 79 (2011), pp. 357–427.

[43] C. BROWNE, E. J. POWLEY, D. WHITEHOUSE, S. M. LUCAS, P. I. COWLING, P. ROHLFSHAGEN, S. TAVENER, D. P. LIEBANA, S. SAMOTHRAKIS, AND S. COLTON, *A survey of monte carlo tree search methods*, IEEE Trans. Comput. Intellig. and AI in Games, 4 (2012), pp. 1–43.

[44] M. BRÜCKNER, *Prediction games: machine learning in the presence of an adversary*, PhD thesis, University of Potsdam, 2012.

[45] M. BRÜCKNER, C. KANZOW, AND T. SCHEFFER, *Static prediction games for adversarial learning problems*, J. Mach. Learn. Res., (2012).

[46] A. BUJA, W. STUETZLE, AND Y. SHEN, *Loss functions for binary class probability estimation and classification: Structure and applications, manuscript, available at www-stat.wharton.upenn.edu/$\tilde{b}$uja*, 2005.

[47] E. CANTU-PAZ, *A survey of parallel genetic algorithms*, CALCULATEURS PARALLELES, 10 (1998).

[48] Q. CAO, L. SHEN, W. XIE, O. M. PARKHI, AND A. ZISSERMAN, *Vggface2: A dataset for recognising faces across pose and age*, in International Conference on Automatic Face and Gesture Recognition, 2018.

[49] N. CARLINI AND D. WAGNER, *Adversarial examples are not easily detected: Bypassing ten detection methods*, in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec '17, New York, NY, USA, 2017, ACM, pp. 3–14.

[50] N. CESA-BIANCHI AND G. LUGOSI, *Prediction and Playing Games*, Cambridge University Press, 2006.

[51] ——, *Prediction, Learning, and Games*, Cambridge University Press, New York, NY, USA, 2006.

[52] A. CHATTOPADHYAY, P. MANUPRIYA, A. SARKAR, AND V. N. BALASUBRAMANIAN, *Neural network attributions: A causal perspective*, in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, 2019, pp. 981–990.

[53] S. CHEN, M. XUE, L. FAN, S. HAO, L. XU, H. ZHU, AND B. LI, *Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach*, Computers and Security, 73 (2018), pp. 326–344.

[54] X. CHEN, Y. DUAN, R. HOUTHOOFT, J. SCHULMAN, I. SUTSKEVER, AND P. ABBEEL, *Infogan: Interpretable representation learning by information maximizing generative adversarial nets*, in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2016, pp. 2172–2180.

[55] M. CHENG, J. YI, H. ZHANG, P. CHEN, AND C. HSIEH, *Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples*, CoRR, abs/1803.01128 (2018).

[56] V. CHERKASSKY AND F. MULIER, *Learning from data: concepts, theory, and methods*, Wiley series on adaptive and learning systems for signal processing, communications, and control, John Wiley & Sons, 2007.

[57] A. CHIVUKULA AND W. LIU, *Adversarial deep learning models with multiple adversaries*, IEEE Transactions on Knowledge and Data Engineering, 31 (2019), pp. 1066–1079.

[58] G. CLAESKENS, N. L. HJORT, ET AL., *Model selection and model averaging*, vol. 330, Cambridge University Press Cambridge, 2008.

[59] P. COMON, X. LUCIANI, AND A. L. F. DE ALMEIDA, *Tensor decompositions, alternating least squares and other tales*, Journal of Chemometrics, 23 (2009), pp. 393–405.

[60] I. CORONA, G. GIACINTO, AND F. ROLI, *Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues*, Inf. Sci., 239 (2013), pp. 201–225.

[61] G. DAI, J. XIE, AND Y. FANG, *Metric-based generative adversarial network*, in Proceedings of the 2017 ACM on Multimedia Conference, MM '17, New York, NY, USA, 2017, ACM, pp. 672–680.

[62] N. DALVI, P. DOMINGOS, MAUSAM, S. SANGHAI, AND D. VERMA, *Adversarial classification*, in Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, New York, NY, USA, 2004, ACM, pp. 99–108.

[63] S. DAS AND P. N. SUGANTHAN, *Differential evolution: A survey of the state-of-the-art*, IEEE Transactions on Evolutionary Computation, 15 (2011), pp. 4–31.

[64] P. DASGUPTA, J. B. COLLINS, AND A. BUHMAN, *Gray-box techniques for adversarial text generation*, in Proceedings of the AAAI Symposium on Adversary-Aware Learning Techniques and Trends in Cybersecurity (ALEC 2018) co-located with the Association for the Advancement of Artificial Intelligence 2018 Fall Symposium Series (AAAI-FSS 2018), Arlington, Virginia, USA, October 18-20, 2018., 2018, pp. 17–23.

[65] S. J. DELANY, P. CUNNINGHAM, A. TSYMBAL, AND L. COYLE, *A case-based technique for tracking concept drift in spam filtering*, Knowl.-Based Syst., 18 (2005), pp. 187–195.

[66] L. DEMETRIO, B. BIGGIO, G. LAGORIO, F. ROLI, AND A. ARMANDO, *Explaining vulnerabilities of deep learning to adversarial malware binaries*, in Proceedings of the Third Italian Conference on Cyber Security, Pisa, Italy, February 13-15, 2019, 2019.

[67] A. DEMONTIS, P. RUSSU, B. BIGGIO, G. FUMERA, AND F. ROLI, *On security and sparsity of linear classifiers for adversarial settings*, in Joint IAPR Int'l Workshop on Structural, Syntactic, and Statistical Pattern Recognition, vol. 10029 of LNCS, Merida, Mexico, 2016, Springer International Publishing, Springer International Publishing, pp. 322–332.

[68] L. DENG, *Three classes of deep learning architectures and their applications: A tutorial survey*, APSIPA Transactions on Signal and Information Processing, (2012).

[69] S. DIRK, *Parallel evolutionary algorithms*, Springer Handbooks., (2015).

[70] R. O. DUDA, P. E. HART, AND D. G. STORK, *Pattern Classification (2Nd Edition)*, Wiley-Interscience, 2000.

[71] J. EBRAHIMI, D. LOWD, AND D. DOU, *On adversarial examples for character-level neural machine translation*, in Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018, 2018, pp. 653–663.

[72] M. EICHLER, *Causal inference with multiple time series: principles and problems*, Phil. Trans. R. Soc. A, 371 (2013), p. 20110613.

[73] K. EYKHOLT, I. EVTIMOV, E. FERNANDES, B. LI, A. RAHMATI, C. XIAO, A. PRAKASH, T. KOHNO, AND D. SONG, *Robust physical-world attacks on deep learning visual classification*, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2018, pp. 1625–1634.

[74] U. FAYYAD, G. PIATETSKY-SHAPIRO, AND P. SMYTH, *The kdd process for extracting useful knowledge from volumes of data*, Commun. ACM, 39 (1996), pp. 27–34.

[75] J. FENG, H. XU, S. MANNOR, AND S. YAN, *Robust logistic regression and classification*, in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1, NIPS'14, Cambridge, MA, USA, 2014, MIT Press, pp. 253–261.

[76] F. FERRUCCI, P. SALZA, AND F. SARRO, *Using hadoop mapreduce for parallel genetic algorithms: A comparison of the global, grid and island models*, Evolutionary Computation, 26 (2018), pp. 535–567.

[77] G. FIDEL, R. BITTON, AND A. SHABTAI, *When explainability meets adversarial learning: Detecting adversarial examples using SHAP signatures*, CoRR, abs/1909.03418 (2019).

[78] C. FINN, P. F. CHRISTIANO, P. ABBEEL, AND S. LEVINE, *A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models*, CoRR, abs/1611.03852 (2016).

[79] W. FLESHMAN, E. RAFF, R. ZAK, M. MCLEAN, AND C. NICHOLAS, *Static malware detection & subterfuge: Quantifying the robustness of machine learning and current anti-virus*, in 13th International Conference on Malicious and Unwanted Software, MALWARE 2018, Nantucket, MA, USA, October 22-24, 2018, 2018, pp. 3–12.

[80] D. B. FOGEL, *An introduction to simulated evolutionary optimization*, IEEE Transactions on Neural Networks, 5 (1994), pp. 3–14.

[81] N. FROSST AND G. E. HINTON, *Distilling a neural network into a soft decision tree*, in Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017, 2017.

[82] T.-C. FU, *A review on time series data mining*, Engineering Applications of Artificial Intelligence, 24 (2011), pp. 164–181.

[83] S. R. L. G. FREILING, G. JANK, *Existence and uniqueness of open-loop stackelberg equilibria in linear-quadratic differential games*, Journal of Optimization Theory and Applications, 110 (2001), pp. 515–544.

[84] Z. GAN, L. CHEN, W. WANG, Y. PU, Y. ZHANG, H. LIU, C. LI, AND L. CARIN, *Triangle generative adversarial networks*, in NIPS, 2017, pp. 5253–5262.

[85] J. GILMER, R. P. ADAMS, I. J. GOODFELLOW, D. ANDERSEN, AND G. E. DAHL, *Motivating the rules of the game for adversarial example research*, CoRR, abs/1807.06732 (2018).

[86] A. GLOBERSON AND S. ROWEIS, *Nightmare at test time: Robust learning by feature deletion*, in Proceedings of the 23rd International Conference on Machine Learning, ICML '06, New York, NY, USA, 2006, ACM, pp. 353–360.

[87] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st ed., 1989.

[88] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep Learning*, MIT Press, 2016.
`http://www.deeplearningbook.org`.

[89] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial nets*, in Advances in neural information processing systems (NIPS), 2014, pp. 2672–2680.

[90] ——, *Generative adversarial nets*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 2672–2680.

[91] I. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, in Proceedings of International Conference on Learning Representations, 2015.

[92] A. GOYAL, N. R. KE, A. LAMB, R. D. HJELM, C. PAL, J. PINEAU, AND Y. BENGIO, *Actual: Actor-critic under adversarial learning*, CoRR, abs/1711.04755 (2017).

[93] C. W. GRANGER, *Investigating causal relations by econometric models and cross-spectral methods*, Econometrica: Journal of the Econometric Society, (1969), pp. 424–438.

[94] K. GREGOR, I. DANIHELKA, A. GRAVES, D. J. REZENDE, AND D. WIERSTRA, *DRAW: A recurrent neural network for image generation*, in Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, 2015, pp. 1462–1471.

[95] K. GROSSE, N. PAPERNOT, P. MANOHARAN, M. BACKES, AND P. D. MCDANIEL, *Adversarial examples for malware detection*, in Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II, 2017, pp. 62–79.

[96] S. GU AND L. RIGAZIO, *Towards deep neural network architectures robust to adversarial examples*, (2015).

[97] R. GUIDOTTI, A. MONREALE, S. RUGGIERI, F. TURINI, F. GIANNOTTI, AND D. PEDRESCHI, *A survey of methods for explaining black box models*, ACM Comput. Surv., 51 (2018).

[98] D. N. GUJARATI AND D. C. PORTER, *Essentials of econometrics*, (1999).

[99] I. GULRAJANI, F. AHMED, M. ARJOVSKY, V. DUMOULIN, AND A. C. COURVILLE, *Improved training of wasserstein gans*, in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 2017, pp. 5767–5777.

[100] S. GUO, C. LADROUE, AND J. FENG, *Granger Causality: Theory and Applications*, vol. 15, Springer-Verlag London Limited, 2010, p. 83.

[101] M. R. GUPTA AND Y. CHEN, *Theory and use of the em algorithm*, Found. Trends Signal Process., 4 (2011), pp. 223–296.

[102] S. GURUMURTHY, R. K. SARVADEVABHATLA, AND R. V. BABU, *Deligan: Generative adversarial networks for diverse and limited data*, in CVPR, IEEE Computer Society, 2017, pp. 4941–4949.

[103] D. HA AND D. ECK, *A neural representation of sketch drawings*, in ICLR 2018.

[104] T. HASTIE, R. TIBSHIRANI, AND J. FRIEDMAN, *The elements of statistical learning – data mining, inference, and prediction*.

[105] M. HAUSCHILD AND M. PELIKAN, *An introduction and survey of estimation of distribution algorithms*, Swarm and Evolutionary Computation, 1 (2011), pp. 111 – 128.

[106] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2016.

[107] D. R. HJELM, A. P. JACOB, T. CHE, K. CHO, AND Y. BENGIO, *Boundary-seeking generative adversarial networks*.

[108] J. HO AND S. ERMON, *Generative adversarial imitation learning*, in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, 2016, pp. 4565–4573.

[109] X. HOU, L. SHEN, K. SUN, AND G. QIU, *Deep feature consistent variational autoencoder*, in 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017, 2017.

[110] L. HUANG, A. D. JOSEPH, B. NELSON, B. I. RUBINSTEIN, AND J. TYGAR, *Adversarial machine learning*, in Proceedings of the 4th ACM workshop on Security and artificial intelligence, ACM, 2011, pp. 43–58.

[111] C. L. L. J. H. G. HUANG CH., LEE TH., *Adversarial attacks on sdn-based deep learning ids system*, in In: Kim K., Kim H. (eds) Mobile and Wireless Technology 2018. ICMWT 2018. Lecture Notes in Electrical Engineering, vol. 513, Springer, Singapore.

[112] A. IGNATIEV, N. NARODYTSKA, AND J. MARQUES-SILVA, *Abduction-based explanations for machine learning models*, 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, 2019, pp. 1511–1519.

[113] A. IGNATIEV, N. NARODYTSKA, AND J. MARQUES-SILVA, *On relating explanations and adversarial examples*, in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., 2019, pp. 15857–15867.

[114] A. ILYAS, L. ENGSTROM, AND A. MADRY, *Prior convictions: Black-box adversarial attacks with bandits and priors*, in 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019.

[115] A. ILYAS, S. SANTURKAR, D. TSIPRAS, L. ENGSTROM, B. TRAN, AND A. MADRY, *Adversarial examples are not bugs, they are features*, in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., 2019, pp. 125–136.

[116] R. JIA AND P. LIANG, *Adversarial examples for evaluating reading comprehension systems*, in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, 2017, pp. 2021–2031.

[117] K. JUN, L. LI, Y. MA, AND X. J. ZHU, *Adversarial attacks on stochastic bandits*, in Advances in Neural Information Processing Systems 31: Annual Conference

on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada., 2018, pp. 3644–3653.

[118] M. KANTARCIOĞLU, B. XI, AND C. CLIFTON, *Classifier evaluation and attribute selection against active adversaries*, Data Mining and Knowledge Discovery, 22 (2011), pp. 291–335.

[119] M. KANTARCIOGLU, B. XI, AND C. CLIFTON, *Classifier evaluation and attribute selection against active adversaries*, Data Min. Knowl. Discov., 22 (2011), pp. 291–335.

[120] H. KAZEMIAN AND S. AHMED, *Comparisons of machine learning techniques for detecting malicious webpages*, Expert Syst. Appl., 42 (2015), pp. 1166–1177.

[121] E. KEOGH, S. CHU, D. HART, AND M. PAZZANI, *Segmenting time series: A survey and novel approach*, Data mining in time series databases, 57 (2004), pp. 1–22.

[122] R. O. KEOHANE, *Counterfactuals and Causal Inference: Methods and Principles for Social Research By Stephen E. Morgan and Christopher Winship Cambridge University Press. 2007. 319 pages.*
83.99*cloth*,
28.99 paper, Social Forces, 88 (2009), pp. 466–467.

[123] A. N. KERCHEVAL AND Y. ZHANG, *Modelling high-frequency limit order book dynamics with support vector machines*, Quantitative Finance, 15 (2015), pp. 1315–1329.

[124] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.

[125] S. KLEINBERG, *Causality, probability, and time*, Cambridge University Press, 2012.

[126] S. KLEINBERG, *Causal inference with rare events in large-scale time-series data*, in Proceedings of the 2013 International Joint Conference on Artificial Intelligence, 2013.

[127] M. KLOFT AND P. LASKOV, *Online anomaly detection under adversarial impact*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Y. W. Teh and M. Titterington, eds., vol. 9 of Proceedings

of Machine Learning Research, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010, PMLR, pp. 405–412.

[128] M. KOCAOGLU, C. SNYDER, A. G. DIMAKIS, AND S. VISHWANATH, *CausalGAN: Learning causal implicit generative models with adversarial training*, in International Conference on Learning Representations, 2018.

[129] P. W. KOH AND P. LIANG, *Understanding black-box predictions via influence functions*, in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML,Äô17, JMLR.org, 2017, p. 1885,Äì1894.

[130] A. KOŁCZ AND C. H. TEO, *Feature Weighting for Improved Classifier Robustness*, in Proc. 6th Conf. on Email and Anti-Spam, July 2009.

[131] J. KOS, I. FISCHER, AND D. SONG, *Adversarial examples for generative models*, in Proceedings of 2018 IEEE Security and Privacy Workshops (SPW), 2018.

[132] J. KOS AND D. SONG, *Delving into adversarial attacks on deep policies*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, 2017.

[133] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[134] N. KUMARI, M. SINGH, A. SINHA, H. MACHIRAJU, B. KRISHNAMURTHY, AND V. N. BALASUBRAMANIAN, *Harnessing the vulnerability of latent layers in adversarially trained models*, in Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, 2019, pp. 2779–2785.

[135] A. KURAKIN, I. J. GOODFELLOW, AND S. BENGIO, *Adversarial examples in the physical world*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, 2017.

[136] ——, *Adversarial machine learning at scale*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[137] H. LAKKARAJU, S. H. BACH, AND J. LESKOVEC, *Interpretable decision sets: A joint framework for description and prediction*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ‚Äô16, New York, NY, USA, 2016, Association for Computing Machinery, p. 1675‚Äì1684.

[138] A. B. L. LARSEN, S. K. SØNDERBY, H. LAROCHELLE, AND O. WINTHER, *Autoencoding beyond pixels using a learned similarity metric*, in Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, pp. 1558–1566.

[139] S. LAXMAN AND P. S. SASTRY, *A survey of temporal data mining*, Sadhana, 31 (2006), pp. 173–198.

[140] Y. LECUN, S. CHOPRA, R. HADSELL, F. J. HUANG, AND ET AL., *A tutorial on energy-based learning*, in PREDICTING STRUCTURED DATA, MIT Press, 2006.

[141] Y. LECUN, C. CORTES, AND C. J. BURGES, *The mnist database*, URL http://yann. lecun. com/exdb/mnist, (1998).

[142] Y. LECUN AND F. HUANG, *Loss functions for discriminative training of energy-based models*, in AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005, pp. 206–213.

[143] S. LEDESMA, G. AVINA, AND R. SANCHEZ, *Practical considerations for simulated annealing implementation*, in Simulated Annealing, C. M. Tan, ed., IntechOpen, Rijeka, 2008, ch. 20.

[144] B. LI AND Y. VOROBEYCHIK, *Feature cross-substitution in adversarial classification*, in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 2087–2095.

[145] B. LI AND Y. VOROBEYCHIK, *Scalable Optimization of Randomized Operational Decisions in Adversarial Classification Settings*, in Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, G. Lebanon and S. V. N. Vishwanathan, eds., vol. 38 of Proceedings of Machine Learning Research, San Diego, California, USA, 09–12 May 2015, PMLR, pp. 599–607.

[146] Z. LI, G. ZHENG, A. AGARWAL, L. XUE, AND T. LAUVAUX, *Discovery of causal time intervals*, in Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, 2013, pp. 804–812.

[147] Y.-C. LIN, Z.-W. HONG, Y.-H. LIAO, M.-L. SHIH, M.-Y. LIU, AND M. SUN, *Tactics of adversarial attack on deep reinforcement learning agents*, in Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, AAAI Press, 2017, pp. 3756–3762.

[148] C. LIU, B. LI, Y. VOROBEYCHIK, AND A. OPREA, *Robust linear regression against training data poisoning*, in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec '17, New York, NY, USA, 2017, ACM, pp. 91–102.

[149] N. LIU, H. YANG, AND X. HU, *Adversarial detection with model interpretation*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ,Äô18, New York, NY, USA, 2018, Association for Computing Machinery, p. 1803,Äì1811.

[150] W. LIU AND S. CHAWLA, *Mining adversarial patterns via regularized loss minimization*, Machine Learning, 81 (2010), pp. 69–83.

[151] ——, *Mining adversarial patterns via regularized loss minimization*, Mach. Learn., 81 (2010), pp. 69–83.

[152] W. LIU, S. CHAWLA, J. BAILEY, C. LECKIE, AND K. RAMAMOHANARAO, *AI 2012: Advances in Artificial Intelligence: 25th Australasian Joint Conference, Sydney, Australia, December 4-7, 2012. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, ch. An Efficient Adversarial Learning Strategy for Constructing Robust Classification Boundaries, pp. 649–660.

[153] Y. LIU, X. CHEN, C. LIU, AND D. SONG, *Delving into transferable adversarial examples and black-box attacks*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[154] Y. LOU, R. CARUANA, AND J. GEHRKE, *Intelligible models for classification and regression*, in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ,Äô12, New York, NY, USA, 2012, Association for Computing Machinery, p. 150,Äì158.

[155] D. LOWD AND C. MEEK, *Adversarial learning*, in Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, New York, NY, USA, 2005, ACM, pp. 641–647.

[156] S. M. LUNDBERG AND S.-I. LEE, *A unified approach to interpreting model predictions*, in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 2017, pp. 4765–4774.

[157] Y. MA, K. JUN, L. LI, AND X. ZHU, *Data poisoning attacks in contextual bandits*, in Decision and Game Theory for Security - 9th International Conference, GameSec 2018, Seattle, WA, USA, October 29-31, 2018, Proceedings, 2018, pp. 186–204.

[158] A. MAKHZANI, J. SHLENS, N. JAITLY, AND I. J. GOODFELLOW, *Adversarial autoencoders*, CoRR, abs/1511.05644 (2015).

[159] M. T. MAMOUN ALAZAB, *Deep Learning Applications for Cyber Security (Advanced Sciences and Technologies for Security Applications)*, Springer Nature, Switzerland AG, 2019.

[160] A. MANDLEKAR, Y. ZHU, A. GARG, L. FEI-FEI, AND S. SAVARESE, *Adversarially robust policy learning: Active construction of physically-plausible perturbations*, in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017, 2017, pp. 3932–3939.

[161] H. C. G. J. MANISH GUPTA, ABHISHEK SINGH, *Context-aware time series anomaly detection for complex systems*, in Proc. of the SDM Workshop on Data Mining for Service and Maintenance, January 2013.

[162] X. MAO, Q. LI, H. XIE, R. Y. K. LAU, Z. WANG, AND S. P. SMOLLEY, *Least squares generative adversarial networks*, in ICCV, IEEE Computer Society, 2017, pp. 2813–2821.

[163] D. L. MARINO, C. S. WICKRAMASINGHE, AND M. MANIC, *An adversarial approach for explainable AI in intrusion detection systems*, in IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, October 21-23, 2018, 2018, pp. 3237–3243.

[164] O. C. MARTIN AND S. W. OTTO, *Combining simulated annealing with local search heuristics*, tech. rep., 1993.

[165] H. MASNADI-SHIRAZI AND N. VASCONCELOS, *On the design of loss functions for classification: theory, robustness to outliers, and savageboost*, in Advances in Neural Information Processing Systems 21, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds., Curran Associates, Inc., 2009, pp. 1049–1056.

[166] F. MATERN, C. RIESS, AND M. STAMMINGER, *Exploiting visual artifacts to expose deepfakes and face manipulations*, in 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW), Jan 2019, pp. 83–92.

[167] S. MEI AND X. ZHU, *The Security of Latent Dirichlet Allocation*, in Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, G. Lebanon and S. V. N. Vishwanathan, eds., vol. 38 of Proceedings of Machine Learning Research, San Diego, California, USA, 09–12 May 2015, PMLR, pp. 681–689.

[168] M. MELIS, A. DEMONTIS, B. BIGGIO, G. BROWN, G. FUMERA, AND F. ROLI, *Is deep learning safe for robot vision? adversarial examples against the icub humanoid*, in 2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017, 2017, pp. 751–759.

[169] M. MELIS, D. MAIORCA, B. BIGGIO, G. GIACINTO, AND F. ROLI, *Explaining black-box android malware detection*, in 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018, 2018, pp. 524–528.

[170] Z. MICHALEWICZ, *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*, Springer-Verlag, Berlin, Heidelberg, 1996.

[171] P. MIROWSKI, M. RANZATO, AND Y. LECUN, *Dynamic auto-encoders for semantic indexing*, in Proceedings of the NIPS 2010 Workshop on Deep Learning, 2010, pp. 1–9.

[172] T. MIYATO, A. M. DAI, AND I. J. GOODFELLOW, *Adversarial training methods for semi-supervised text classification*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, 2017.

[173] S. MOOSAVI-DEZFOOLI, A. FAWZI, AND P. FROSSARD, *Deepfool: A simple and accurate method to fool deep neural networks*, in Proceedings of Conference on Computer Vision and Pattern Recognition CVPR, 2016.

[174] Y. MROUEH AND T. SERCU, *Fisher gan*, in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 2017, pp. 2513–2523.

[175] Y. MROUEH, T. SERCU, AND V. GOEL, *McGan: Mean and covariance feature matching GAN*, in Proceedings of the 34th International Conference on Machine Learning, D. Precup and Y. W. Teh, eds., vol. 70 of Proceedings of Machine Learning Research, International Convention Centre, Sydney, Australia, 06–11 Aug 2017, PMLR, pp. 2527–2535.

[176] N. NARODYTSKA, S. P. KASIVISWANATHAN, L. RYZHYK, M. SAGIV, AND T. WALSH, *Verifying properties of binarized deep neural networks*, in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 6615–6624.

[177] A. NEMIROVSKI, A. JUDITSKY, G. LAN, AND A. SHAPIRO, *Robust stochastic approximation approach to stochastic programming*, SIAM J. on Optimization, 19 (2009), pp. 1574–1609.

[178] A. NGUYEN, J. YOSINSKI, AND J. CLUNE, *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images*, (2015).

[179] T. NGUYEN, T. LE, H. VU, AND D. PHUNG, *Dual discriminator generative adversarial nets*, in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., 2017, pp. 2667–2677.

[180] N. NISAN, T. ROUGHGARDEN, E. TARDOS, AND V. V. VAZIRANI, *Algorithmic Game Theory*, Cambridge University Press, New York, NY, USA, 2007.

[181] J. OCENASEK, E. CANTÚ-PAZ, M. PELIKAN, AND J. SCHWARZ, *Design of parallel estimation of distribution algorithms*, in Scalable Optimization via Probabilistic Modeling, 2006, pp. 187–203.

[182] F. A. OLIEHOEK, R. SAVANI, J. GALLEGO-POSADA, E. VAN DER POL, E. D. DE JONG, AND R. GROSS, *Gangs: Generative adversarial network games*, CoRR, abs/1712.00679 (2017).

[183] N. PAPERNOT, P. MCDANIEL, I. GOODFELLOW, S. JHA, Z. BERKAY CELIK, AND A. SWAMI, *Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples*, ArXiv e-prints, (2016).

[184] N. PAPERNOT, P. MCDANIEL, I. GOODFELLOW, S. JHA, Z. B. CELIK, AND A. SWAMI, *Practical black-box attacks against machine learning*, in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, New York, NY, USA, 2017, ACM, pp. 506–519.

[185] N. PAPERNOT, P. MCDANIEL, A. SINHA, AND M. P. WELLMAN, *Sok: Security and privacy in machine learning*, in 2018 IEEE European Symposium on Security and Privacy (EuroS P), April 2018, pp. 399–414.

[186] N. PAPERNOT, P. D. MCDANIEL, A. SINHA, AND M. P. WELLMAN, *Towards the science of security and privacy in machine learning*, CoRR, abs/1611.03814 (2016).

[187] G. PEAKE AND J. WANG, *Explanation mining: Post hoc interpretability of latent factor models for recommendation systems*, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ,Äô18, New York, NY, USA, 2018, Association for Computing Machinery, p. 2060,Äì2069.

[188] M. PELIKAN, D. E. GOLDBERG, AND E. CANTU-PAZ, *Linkage problem, distribution estimation, and bayesian networks*, Evolutionary Computation, 8 (2000), pp. 311–340.

[189] D. PFAU AND O. VINYALS, *Connecting generative adversarial networks and actor-critic methods*, CoRR, abs/1610.01945 (2016).

[190] L. PINTO, J. DAVIDSON, R. SUKTHANKAR, AND A. GUPTA, *Robust adversarial reinforcement learning*, in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, 2017, pp. 2817–2826.

[191] M. PIRLOT, *General local search methods*, European Journal of Operational Research, 92 (1996), pp. 493–511.

[192] B. POOLE, A. ALEMI, J. SOHL-DICKSTEIN, AND A. ANGELOVA, *Improved generator objectives for gans*, 2016.

[193] Z. S. W. C. QIU S, LIU Q, *Review of artificial intelligence adversarial attack and defense technologies*, in MDPI Applied Sciences, 2019, p. 9(5):909.

[194] A. RADFORD, L. METZ, AND S. CHINTALA, *Unsupervised representation learning with deep convolutional generative adversarial networks*.

[195] S. D. RAMCHURN, P. VYTELINGUM, A. ROGERS, AND N. R. JENNINGS, *Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence*, Commun. ACM, 55 (2012), pp. 86–97.

[196] M. T. RIBEIRO, S. SINGH, AND C. GUESTRIN, *„Äúwhy should i trust you?„Äù: Explaining the predictions of any classifier*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD „Äô16, New York, NY, USA, 2016, Association for Computing Machinery, p. 1135„Äì1144.

[197] M. T. RIBEIRO, S. SINGH, AND C. GUESTRIN, *Anchors: High-precision model-agnostic explanations*, in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 1527–1535.

[198] L. ROSASCO, E. DE VITO, A. CAPONNETTO, M. PIANA, AND A. VERRI, *Are loss functions all the same?*, Neural Comput., 16 (2004).

[199] K. ROSE, *Deterministic annealing for clustering, compression, classification, regression, and related optimization problems*, Proceedings of the IEEE, 86 (1998), pp. 2210–2239.

[200] I. ROSENBERG, A. SHABTAI, L. ROKACH, AND Y. ELOVICI, *Generic black-box end-to-end attack against state of the art API call based malware classifiers*, in Research in Attacks, Intrusions, and Defenses - 21st International Symposium,

RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings, 2018, pp. 490–510.

[201] A. ROSSLER, D. COZZOLINO, L. VERDOLIVA, C. RIESS, J. THIES, AND M. NIESSNER, *Faceforensics++: Learning to detect manipulated facial images*, CoRR, abs/1901.08971 (2019).

[202] S. ROTA BULO, B. BIGGIO, I. PILLAI, M. PELILLO, AND F. ROLI, *Randomized prediction games for adversarial machine learning*, IEEE Transactions on Neural Networks and Learning Systems, 28 (2017), pp. 2466–2478.

[203] B. I. RUBINSTEIN, P. L. BARTLETT, L. HUANG, AND N. TAFT, *Learning in a large function space: Privacy-preserving mechanisms for svm learning*, Journal of Privacy and Confidentiality, Vol.4 : Iss.1, Article 4. (2009).

[204] B. I. RUBINSTEIN, B. NELSON, L. HUANG, A. D. JOSEPH, S.-H. LAU, S. RAO, N. TAFT, AND J. D. TYGAR, *Antidote: Understanding and defending against poisoning of anomaly detectors*, in Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, IMC '09, New York, NY, USA, 2009, ACM, pp. 1–14.

[205] C. RUDIN, *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*, Nature Machine Intelligence, 1 (2019), pp. 206–215.

[206] P. SAMANGOUEI, M. KABKAB, AND R. CHELLAPPA, *Defense-gan: Protecting classifiers against adversarial attacks using generative models*.

[207] S. SAMANTA AND S. MEHTA, *Generating adversarial text samples*, in Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings, 2018, pp. 744–749.

[208] T. SCHLEGL, P. SEEBÖCK, S. M. WALDSTEIN, U. SCHMIDT-ERFURTH, AND G. LANGS, *Unsupervised anomaly detection with generative adversarial networks to guide marker discovery*, (2017), pp. 146–157.

[209] M. SHARIF, S. BHAGAVATULA, L. BAUER, AND M. K. REITER, *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, CCS ,Äô16, New York, NY, USA, 2016, Association for Computing Machinery, p. 1528,Äì1540.

[210] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE Symposium on Security and Privacy (SP), May 2017, pp. 3–18.

[211] A. SHRIKUMAR, P. GREENSIDE, AND A. KUNDAJE, *Learning important features through propagating activation differences*, in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML,Äô17, JMLR.org, 2017, p. 3145,Äì3153.

[212] M. SIMAAN AND J. B. CRUZ, JR., *On the stackelberg strategy in nonzero-sum games*, J. Optim. Theory Appl., 11 (1973), pp. 533–555.

[213] A. SINHA, P. MALO, AND K. DEB, *A review on bilevel optimization: From classical to evolutionary approaches and applications*, IEEE Transactions on Evolutionary Computation, 22 (2018), pp. 276–295.

[214] J. SIRIGNANO, *Deep learning for limit order books*, Trading and Market Microstructure, (2016).

[215] J. C. SPALL, *Introduction to Stochastic Search and Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 1 ed., 2003.

[216] P. SPIRTES, C. N. GLYMOUR, AND R. SCHEINES, *Causation, prediction, and search*, MIT press, 2000.

[217] A. SPURR, E. AKSAN, AND O. HILLIGES, *Guiding infogan with semi-supervision*, in ECML/PKDD (1), vol. 10534 of Lecture Notes in Computer Science, Springer, 2017, pp. 119–134.

[218] E. STRUMBELJ AND I. KONONENKO, *An efficient explanation of individual classifications using game theory*, J. Mach. Learn. Res., 11 (2010), p. 1,Äì18.

[219] L. SUN, M. TAN, AND Z. ZHOU, *A survey of practical adversarial example attacks*, Cybersecurity, 1 (2018), p. 9.

[220] V. SURYAN, A. SINHA, P. MALO, AND K. DEB, *Handling inverse optimal control problems using evolutionary bilevel optimization*, in 2016 IEEE Congress on Evolutionary Computation (CEC), July 2016, pp. 1893–1900.

[221] C. SZEGEDY, W. ZAREMBA, I. SUTSKEVER, J. BRUNA, D. ERHAN, I. GOODFELLOW, AND R. FERGUS, *Intriguing properties of neural networks*, in International Conference on Learning Representations, 2014.

[222] G. TAO, S. MA, Y. LIU, AND X. ZHANG, *Attacks meet interpretability: Attribute-steered detection of adversarial samples*, in Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada, 2018, pp. 7728–7739.

[223] R. TOMSETT, A. WIDDICOMBE, T. XING, S. CHAKRABORTY, S. JULIER, P. GURRAM, R. RAO, AND M. SRIVASTAVA, *Why the failure? how adversarial examples can provide insights for interpretable machine learning*, in 2018 21st International Conference on Information Fusion (FUSION), July 2018, pp. 838–845.

[224] L. TONG, B. LI, C. HAJAJ, C. XIAO, N. ZHANG, AND Y. VOROBEYCHIK, *Improving robustness of ML classifiers against realizable evasion attacks using conserved features*, in 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019, 2019, pp. 285–302.

[225] N. TRAN, T. BUI, AND N. CHEUNG, *Dist-gan: An improved GAN using distance constraints*, in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV, 2018, pp. 387–401.

[226] N.-T. TRAN, T.-A. BUI, AND N.-M. CHEUNG, *Dist-gan: An improved gan using distance constraints*, in Proceedings of European Conference on Computer Vision (ECCV), 2018.

[227] C. TSALLIS AND D. A. STARIOLO, *Generalized simulated annealing*, Physica A: Statistical Mechanics and its Applications, 233 (1996), pp. 395 – 406.

[228] M. UMMELS, *Stochastic multiplayer games: theory and algorithms*, PhD thesis, RWTH Aachen University, 2011.

[229] A. VERMA, X. LLORA, D. E. GOLDBERG, AND R. H. CAMPBELL, *Scaling genetic algorithms using mapreduce*, in 2009 Ninth International Conference on Intelligent Systems Design and Applications, Nov 2009, pp. 13–18.

[230] M. VIDYADHARI, K. KIRANMAI, K. R. KRISHNIAH, AND D. S. BABU, *Security evaluation of pattern classifiers under attack*, International Journal of Research, 3 (2016), pp. 1043–1048.

[231] F. WANG, W. LIU, AND S. CHAWLA, *On sparse feature attacks in adversarial learning*, in 2014 IEEE International Conference on Data Mining, Dec 2014, pp. 1013–1018.

[232] K. WANG, C. GOU, Y. DUAN, Y. LIN, X. ZHENG, AND F. WANG, *Generative adversarial networks: introduction and outlook*, IEEE/CAA Journal of Automatica Sinica, 4 (2017), pp. 588–598.

[233] T. WANG AND Q. LIN, *Hybrid predictive model: When an interpretable model collaborates with a black-box model*, CoRR, abs/1905.04241 (2019).

[234] T. WEISE, *Global Optimization Algorithms - Theory and Application*, self-published, Germany, 2009.

[235] L. D. WHITLEY, S. B. RANA, J. DZUBERA, AND K. E. MATHIAS, *Evaluating evolutionary algorithms*, Artif. Intell., 85 (1996), pp. 245–276.

[236] D. H. WOLPERT AND W. G. MACREADY, *No free lunch theorems for optimization*, IEEE Transactions on Evolutionary Computation, 1 (1997), pp. 67–82.

[237] Y. XIAN, T. LORENZ, B. SCHIELE, AND Z. AKATA, *Feature generating networks for zero-shot learning*, in 31st IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), Salt Lake City, UT, USA, 2018.

[238] C. XIAO, B. LI, J.-Y. ZHU, W. HE, M. LIU, AND D. SONG, *Generating adversarial examples with adversarial networks*, (2018).

[239] C. XIAO, J. ZHU, B. LI, W. HE, M. LIU, AND D. SONG, *Spatially transformed adversarial examples*, in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

[240] H. XIAO, B. BIGGIO, G. BROWN, G. FUMERA, C. ECKERT, AND F. ROLI, *Is feature selection secure against training data poisoning?*, in Proceedings of the 32nd International Conference on Machine Learning, F. Bach and D. Blei, eds., vol. 37 of Proceedings of Machine Learning Research, Lille, France, 07–09 Jul 2015, PMLR, pp. 1689–1698.

[241] H. XIAO, B. BIGGIO, B. NELSON, H. XIAO, C. ECKERT, AND F. ROLI, *Support vector machines under adversarial label contamination*, Neurocomput., 160 (2015), pp. 53–62.

[242] C. XIE, J. WANG, Z. ZHANG, Y. ZHOU, L. XIE, AND A. L. YUILLE, *Adversarial examples for semantic segmentation and object detection*, in IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, 2017, pp. 1378–1387.

[243] H. XU, M. FARAJTABAR, AND H. ZHA, *Learning granger causality for hawkes processes*, in International Conference on Machine Learning, 2016, pp. 1717–1726.

[244] C. H. YANG, Y. LIU, P. CHEN, X. MA, AND Y. J. TSAI, *When causal intervention meets adversarial examples and image masking for deep neural networks*, in 2019 IEEE International Conference on Image Processing (ICIP), Sep. 2019, pp. 3811–3815.

[245] L. YANG, P. LI, Y. ZHANG, X. YANG, Y. XIANG, AND W. ZHOU, *Effective repair strategy against advanced persistent threat: A differential game approach*, IEEE Transactions on Information Forensics and Security, 14 (2019), pp. 1713–1728.

[246] Z. YIN, F. WANG, W. LIU, AND S. CHAWLA, *Sparse feature attacks in adversarial learning*, IEEE Transactions on Knowledge and Data Engineering, PP (2018).

[247] ——, *Sparse feature attacks in adversarial learning*, IEEE Transactions on Knowledge and Data Engineering, 30 (2018), pp. 1164–1177.

[248] F. ZHANG, P. P. K. CHAN, B. BIGGIO, D. S. YEUNG, AND F. ROLI, *Adversarial feature selection against evasion attacks*, IEEE Trans. Cybernetics, 46 (2016), pp. 766–777.

[249] J. ZHANG, Z. HUI ZHAN, Y. LIN, N. CHEN, Y. JIAO GONG, J.-H. ZHONG, H. S.-H. CHUNG, Y. LI, AND Y. HUI SHI, *Evolutionary computation meets machine learning: A survey.*, IEEE Comp. Int. Mag., 6 (2011), pp. 68–75.

[250] J. ZHANG, Z. ZHAN, Y. LIN, N. CHEN, Y. GONG, J. ZHONG, H. S. H. CHUNG, Y. LI, AND Y. SHI, *Evolutionary computation meets machine learning: A survey*, IEEE Computational Intelligence Magazine, 6 (2011), pp. 68–75.

[251] J. J. ZHAO, M. MATHIEU, AND Y. LECUN, *Energy-based generative adversarial networks*.

[252] M. ZHAO, B. AN, Y. YU, S. LIU, AND S. J. PAN, *Data poisoning attacks on multi-task relationship learning*, in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 2628–2635.

[253] Y. ZHOU AND M. KANTARCIOGLU, *Modeling adversarial learning as nested stackelberg games*, in Advances in Knowledge Discovery and Data Mining, J. Bailey, L. Khan, T. Washio, G. Dobbie, J. Z. Huang, and R. Wang, eds., Cham, 2016, Springer International Publishing, pp. 350–362.

[254] Y. ZHOU, M. KANTARCIOGLU, AND B. XI, *A survey of game theoretic approach for adversarial machine learning*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, (2019).

[255] Z. ZHOU, H. CAI, S. RONG, Y. SONG, K. REN, W. ZHANG, J. WANG, AND Y. YU, *Activation maximization generative adversarial nets*, in International Conference on Learning Representations, 2018.