Adversarial Attacks, Coarse Robustness, and Dataless Neural Networks: Novel Techniques for Improved Classification and Combinatorial Optimization

Ismail Alkhouri
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/etd2020

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
https://stars.library.ucf.edu/etd2020/1500
ADVERSARIAL ATTACKS, COARSE ROBUSTNESS, AND DATALESS NEURAL NETWORKS: NOVEL TECHNIQUES FOR IMPROVED CLASSIFICATION AND COMBINATORIAL OPTIMIZATION

by

ISMAIL RASHID ALKHOURI
B.S. in Electronics and Communications Engineering, University of Baghdad, 2009
M.S. in Electrical Engineering, San Diego State University, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2023

Major Professor: George K. Atia
ABSTRACT

Neural networks (NN) have become a central component in most machine learning systems. However, studies have shown that these models are not robust against adversarial attacks. As such, in this dissertation, we explore four directions. In the first direction, we investigate adversarial attacks on two hierarchical classification (HC) models: the Flat HC (FHC), and the Top-Down HC (TDHC). In particular, we formulate attacks against these models by using convex programming. Through experimental results, it is shown that FHCs are more robust than TDHCs. Second, we formalize a new notion of coarse robustness that is defined with respect to a specified grouping of the class labels. We propose a training mechanism that incorporates the coarse label information in addition to the finer ones, and empirically and theoretically show that this mechanism improves the proposed notion of coarse robustness. The third direction is the Bidirectional One-Shot Synthesis (BOSS) problem for synthesizing adversarial examples using structures similar to generative adversarial networks. However, BOSS does not require the use of any training data. In particular, we explore solutions where the generated data must simultaneously satisfy input/output user-defined constraints. We prove that the BOSS problem is \textbf{NP-complete}, and experimentally verify that the method either outperforms or performs on par with the state-of-the-art methods. Subsequently, for the fourth direction, we extend the synthesis problem of adversarial attacks to solving the Maximum Independent Set (MIS) problem. This is accomplished by presenting NN structures derived with respect to finding MISs in the graph, where no data is required for training the neural networks that produce the solution. Experimental results on various graphs demonstrate that our proposed method performs on par or outperforms state-of-the-art learning-based methods without requiring any training data.
Dedicated to Rashid, Manal, Rand, Fayrouz, and Farah
ACKNOWLEDGMENTS

I would like to thank my advisors Dr. George Atia, Dr. Wasfy Mikhael, and Dr. Alvaro Velasquez for providing the best possible forms of motivation, knowledge, kindness, and guidance.

I would like to thank my family without which this work would not have been possible. My wife, Rand, for her unlimited support and belief in this journey. She moved with me across the country to make this possible. My first and best teacher, my father, for his wisdom and motivation. I am grateful for my mother whose smile and positive attitude have given me strength to keep going. Thanks to Farah, my daughter, the source of my happiness, joy, and biggest motivation. I am thankful for my sister, Dr. Fayrouz Alkhouri, and brothers, Sinan Musa, and Dr. Rami Alattar, for their support and presence, and for being the best listeners to my jokes. I am grateful to my uncle, Dr. Rafid Alkhoury, for being an inspiration of scientific research.

I would like to thank my bigger family, my uncles, aunts, grandparents, and in-laws. Many thanks to my friends, Zaid Matloub, Hashim Talib, Fahmi Alsabie, Anmar Enja, Karam Alkhoury, Husam Alkhoury, Dr. Noori BniLam, and Dr. Andre Beckus for their support and shared laughs.

As the main sponsors of my research, I would like to acknowledge the support of the Air Force Research Laboratory Contract FA8750-20-3-1004, the Air Force Office of Scientific Research Award 20RICOR012, NSF CAREER Award CCF1552497, and NSF grant CCF2106339.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. x

LIST OF TABLES ................................................................. xiv

CHAPTER 1: INTRODUCTION .................................................. 1

Notation and Preliminaries .................................................. 3

Classification Models ......................................................... 4

Alternating Direction Method of Multipliers ............................. 5

The Maximum Independent Set Problem ................................. 6

CHAPTER 2: ADVERSARIAL ATTACKS ON HIERARCHICAL CLASSIFIERS VIA CONVEX PROGRAMMING ......................... 7

Introduction ........................................................................ 7

Related Work ..................................................................... 8

Hierarchical Classification Models ...................................... 9

Nested Dichotomies ............................................................. 12

Attack Formulations ........................................................... 13

ADMM-based Solution ......................................................... 20
Experimental Results ................................................................. 21
Discussion and Conclusion ....................................................... 28

CHAPTER 3: ON THE COARSE ROBUSTNESS OF CLASSIFIERS .......... 29
Introduction .............................................................................. 29
The Coarse Robustness Measure .............................................. 30
Finding Best Mappings ............................................................. 32
Improving the Coarse Robustness ............................................ 34
Numerical Results ................................................................. 38
Discussion and Conclusion ..................................................... 41

CHAPTER 4: BIDIRECTIONAL ONE-SHOT SYNTHESIS FOR ADVERSARIAL EX-
AMPLES .......................................................... 42
Introduction .............................................................................. 42
Related Work ........................................................................... 43
Problem Formulation & Characterization ............................... 45
Generative Approach ............................................................... 49
Attack Formulation against TDHC Two-stage models using BOSS 51
Experimental Results ............................................................. 52
CHAPTER 5: AN APPROACH TO THE MAXIMUM INDEPENDENT SET PROBLEM USING NEURAL NETWORK STRUCTURES

Introduction

Related Work

Methodology

Dataless Neural Network Construction

On the Duality of MIS and MC

Scaling up: Community Detection Approach

Solution Improvement by dNNs

Experimental Evaluation

Setup, Benchmarks, and Baselines

Results on SNAP and Citation Network Benchmarks

Results on Synthetic Graphs

Advantage of Degree-based Initialization

Impact of the choice of $\alpha$

Discussion and Conclusion
# LIST OF FIGURES

2.1 Flat Hierarchical Classifier (FHC) block diagram. ........................................... 10

2.2 Top Down Hierarchical Classifier (TDHC) block diagram. .............................. 11

2.3 For the FHC (TDHC) model, the classification direction is shown by the bottom-up (top-down) arrow on the left (right). .................................................... 11

2.4 An Example of a Nested Dichotomies structure with $M = 5$ leaf nodes. ....... 12

2.5 Proposed NDC structure for MNIST fashion dataset with $M = 10$ leaf nodes (underlined labels). Sets $R(m)$ and $V(m), \forall m \in [10]$ are presented. The classification accuracy for every binary classifier $C_k, \forall k \in [9]$, is given at the bottom. The overall classification accuracy is 89.2%. Grouping of the leaf node classes is based on the location and the shape of the item. .......................... 23

2.6 Samples of altering the super (coarser) label according to a target super (coarser) class for the FHC-A (TDHC-C) attack with perceptibility measure $\rho_2$. .......... 24

2.7 Perturbations needed to alter the super/coarser classification for the first 100 samples of the MNIST fashion dataset represented by SSIM (left) and the CDF of the SSIM (right). .......................................................... 24

2.8 Samples of the TDHC-F attack with the required perturbations represented by the perceptibility measure $\rho_2$. ................................................................. 25
2.9 Samples of each class of the MNIST fashion dataset (top), the corresponding perturbed versions from MIFGSM (middle) where $\eta_M$ represents the generated perturbations from MIFGSM, and the corresponding perturbed versions from Algorithm 1 (bottom). .......................... 27

2.10 SSIM ($I$) of the first 100 examples with MIFGSM and our proposed Algorithm (left) and CDF of $I$ (right). The lines in the left plot are the smoothed average for each case with moving average window of length 25. ............ 27

3.1 An example of the additional parameters required for the AHCT method in a neural network classifier. ................................. 36

3.2 Results for the best mapping function $T$ obtained as a solution to (BM) using the CMQ matrix $\Pi_2$ for CIFAR-10 (first and second) and FMNIST (third and fourth). ................................. 38

4.1 Problem demonstration (left) and examples (right). The true labels are placed on the left of each desired features (image) with bold font. First row of images represent the desired features where the desired PMFs are placed on the left of each sample. The second row presents the synthesized examples, by BOSS, with their corresponding predictions w.r.t their trained classifiers. From left to right, samples are picked from the MNIST digits [1], MNIST fashion [2], CIFAR-10 [3], GTSRB [4], and ImageNet [5], respectively. .............................. 43

4.2 Example reduction from a graph $G$ (left) to a classifier $p(\cdot; \theta)$ (right). ......... 48
4.3 Samples from each class of the CIFAR-10 dataset (columns). The first row shows the original examples. Rows 2-4 represent the synthesized images for BOSS-C, BOSS-B, and BOSS-T, respectively. The rounded percentage values of the confidence level, \( c \), of the BOSS-C samples are placed at the bottom of each image along with the predicted label. For BOSS-B, the first pair at the bottom of every image represents the highest two predicted labels along with their rounded classification scores (second pair). Predicted labels are placed at the bottom of each BOSS-T example. The percentage of the rounded similarity measure \( I \) is placed on top of each generated example.

4.4 Clean samples from some classes of the ImageNet dataset on VGG16. The first row shows the original examples. Rows 2 and 3 show the synthesized images for BOSS-T and BOSS-B, respectively. The labels are given at the bottom of each example along with the classification confidence. The SSIM and JS measures are reported on the right of each BOSS image.

5.1 Block diagram of the proposed dNN, \( f(\theta) \).

5.2 An example of graph \( G = (V = \{v_1, v_2, v_3, v_4, v_5\}, E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\}) \) and its dNN construction \( f \) for the MIS problem.

5.3 Network \( f \) used in the proof of Theorem 4.1.

5.4 An example of graph \( G = (V = \{v_1, v_2, v_3, v_4, v_5\}, E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\}) \) and its dNN construction \( h \) for the MIS problem by leveraging the duality between MIS and MC.
5.5 Comparison between the proposed degree-based initialization and the initialization from a uniform random distribution in terms of the average size of the found MIS (top) and the average execution time (bottom) 76

5.6 The impact of the probability threshold $\alpha$ w.r.t. the average number of tuning steps (first), average execution time (second), average normalized output value with re-scaled zoomed values (third), and average size $k$ of the MIS found (fourth) 78
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>CNN Architecture w.r.t the output size $L_c$.</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>FHC attacks averaged over 1000 trials.</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>eADMM comparison with state-of-the-art solver CVXPY [6] w.r.t the attacks</td>
<td>26</td>
</tr>
<tr>
<td>2.4</td>
<td>Overall Performance of Algorithm 2 and MIFGSM.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Performance of Algorithm 3 in obtaining the best overall mapping. The results are based on the average obtained optimal values of $\alpha_2$, and the average required number of iterations over 5 different initial mappings, $\mathbb{E}(Q)$.</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td>Results for enhancing the coarse robustness evaluated using the GCR and GSR measures for models trained using NT and AHCT.</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>BOSS-C and NewtonFool with $c_d = 0.6$, $\delta_c = 0.2$, and $\delta_s = 0.85$ for the MNIST dataset.</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>BOSS-T attack overall comparison with state-of-the-art attack methods using the CIFAR-10 dataset. For targets, all labels other than the predicted ones are considered.</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison to state-of-the-art baselines using real-world benchmarks in terms of the size of the identified MIS.</td>
<td>74</td>
</tr>
</tbody>
</table>
5.2 Comparison to state-of-the-art baselines using synthetic graphs in terms of the average size of the MIS.
CHAPTER 1: INTRODUCTION

There has been enormous progress in the development of powerful classifiers in numerous applications of machine learning and artificial intelligence, including modern techniques that make use of deep learning architectures and neural networks [7, 8, 9, 10]. However, recent literature has revealed the fragility of One-Stage Classifiers (OSCs) given their susceptibility to imperceptible, crafted perturbation attacks [11, 12, 13, 14]. Understanding the impact of adversarial attacks is both critical and momentous considering the envisioned mass adoption of such classifiers in safety-critical systems, such as in autonomous driving, surveillance, and military applications [15].

There are several taxonomies one could use to categorize adversarial attacks based on attacker’s side information, goal of the attack, attack scenario, and scope of the attack. In the side information-based taxonomy, adversarial attacks can be characterized as black (white) box attacks when the attacker has no (full) access to the classifier’s function [16] and semi-black-box attacks when the attacker has partial access [17]. The strongest adversary is the white-box attacker given its full knowledge of the target model. As such, defense methods that succeed against black-box/semi-black-box attacks could be vulnerable to an efficient white-box attack [18]. Depending on the goal of the attack, attacks can be classified into non-targeted, targeted, and confidence reduction attacks. The goal of non-targeted attacks is to modify the input in such a way that it is misclassified, whereas targeted attacks seek to alter the output prediction to a predefined target class [19]. Confidence reduction attacks aim to reduce the confidence in the label estimation of the target model to introduce ambiguity [20]. In the context of the attack scenario, evasion attacks refer to scenarios where the adversary attempts to evade the detection system during the system operation when samples are modified at test time, while a poisoning attack (also known as contamination attack) is when the adversary attempts to poison the data during the training phase [21]. Attacks can also be assorted into individual or universal attacks based on their scope. Individual attacks generate
perturbations against every input feature vector, while universal perturbations are designed against the entire dataset [20]. In terms of the nature of the perturbations, they can be additive when perturbations are added to the example, or non-additive where techniques such as rotation, inversion, and other transformations are applied to the original sample [22, 23]. In this thesis, we assume a white-box, evasion, and individual attack scenario.

In Chapter 2, instead of One-Stage Classifiers (OSCs), we steer our focus to generating adversarial perturbations against Hierarchical Classifiers. In particular, we consider (i) the Flat Hierarchical Classifier (FHC), where a testing example is first classified according to the finer class, then the upper label is retrieved from the fine classification, and (ii) the Top-Down Hierarchical Classifier (TDHC) where the observation is first classified according to the coarser genres, and then finer classifiers are used based on the result of the initial upper level classification. We formulate convex programs to achieve our goal. We derive an efficient iterative algorithm based on the Alternating Direction Method of Multipliers (ADMM).

Chapter 3 focuses on OSCs where we introduce a new notion of global coarse robustness of a classifier which captures the hardness of confusing the coarse predictions induced by a certain grouping of the class labels. Furthermore, we propose measures of goodness of class mappings and formulate an optimization problem to optimize such measures. An algorithmic procedure is proposed and shown to perform on par with combinatorial brute force search, while requiring a considerably smaller number of iterations. By utilizing the coarse information, we develop the coarse training approach for obtaining improved coarse robustness models.

In Chapter 4, we present our generative attack method against OSCs. First, we present the Bidirectional One-Shot Synthesis (BOSS) problem to synthesize feature vectors that follow some desired input and output specifications. Second, we prove that BOSS is \textbf{NP-complete}. Third, we propose our algorithmic procedure that is based on generative networks and the back-propagation algo-
rithm [24] to produce (from scratch) these examples. Fourth, we present methods to select the input/output specifications to generate targeted adversarial attacks, confidence reduction attacks, and decision boundary samples.

Inspired by the BOSS problem in Chapter 4, in Chapter 5, we extend the synthesis problem to solving the Maximum Independent Set (MIS) Problem in graphs. The MIS problem is a known **NP-complete** problem, hence, falling under the category of combinatorial optimization problems. This is accomplished by developing Neural Network (NN) structures that are derived from the graph of interest and with respect to the MIS problem. Instead of generating adversarial examples, we represent the MIS problem in a single differentiable NN function. As such, we generate inputs that correspond to solutions in the graph using the backpropagation algorithm.

**Notation and Preliminaries**

Throughout this dissertation, we use boldface uppercase letters (e.g., $X$) to denote matrices, boldface lowercase letters (e.g., $x$) to denote vectors, and Roman lowercase letters (e.g., $x$) to represent scalars. The operator $|.|$ is used for the cardinality of a set, as well as the absolute value, which will be clear from the context. Given a set $S$, the set $S'$ denotes its complement. The $p$-norm of a vector $x = (x_1, \ldots, x_n)$ is defined as $\|x\|_p := (\sum_{i=1}^{n}|x_i|^p)^{1/p}$ for $p \in [1, \infty)$. For any positive integer $M$, the index set $[M] := \{1, \ldots, M\}$. The set difference of sets $A$ and $B$ is the set of elements in $A$ that are not in $B$ which is denoted as $A \setminus B$. Given vector $x$, vector $y$, and matrix $Z$, the notation $Z = \begin{bmatrix} x & y \end{bmatrix}$ refers to the matrix obtained by concatenating columns $x$ and $y$. The matrices $Z^T$, $Z^{-1}$, and $Z^\dagger$ represent the transpose, inverse, and psuedo-inverse of matrix $Z$, respectively. We use sign(.) to denote the signum function and $I_N$ denotes the identity matrix of size $N \times N$.

An undirected graph is denoted by $G = (V,E)$, where $V$ is its vertex set and $E \subseteq V \times V$ is its
edge set. The number of nodes is $|V| = n$ and the number of edges is $|E| = m$. We also use the notation $V(H)$ and $E(H)$ to refer to the vertex and edge sets of some graph $H$, respectively. The degree of a node $v \in V$ is denoted by $d(v)$, and the maximum degree of the graph by $\Delta(G)$. The neighborhood of node $v \in V$ is $N(v) = \{ u \in V \mid (u, v) \in E \}$. For a subset of nodes $U \subseteq V$, $G[U] = (U, E[U])$ is used to represent the subgraph induced by $U$, i.e., the graph on $U$ whose edge set $E[U] = \{ (u, v) \in E \mid u, v \in U \}$ consists of all edges of $G$ with both ends in $U$. The complement of graph $G$ is the graph $G' = (V, E')$ on $V$, where $E' = V \times V \setminus E$, i.e., $E'$ consists of all the edges between nodes that are not adjacent in $G$, with $|E'| = m'$. Hence, $m + m' = n(n - 1)/2$ is the number of edges in the complete graph on $V$.

Next, we present the preliminary materials and concepts required in this dissertation.

**Classification Models**

For the classification of $M$ classes, let $k(.)$ be the classifier function that maps the input vector $x \in \mathbb{R}^N$ to its estimated label, i.e., $k(x) \in [M]$. While the function $k$ is not differentiable, there exist $M$ discriminant functionals $f_i : \mathbb{R}^N \to \mathbb{R}, i \in [M]$ such that classifier $k$ is given as a vector valued discriminant function $f(x)$, which returns the evaluation of $M$ scalar functions $f_i, i \in [M]$ representing the class belonging. Classifier $k$ is then given as,

$$k(x) = \arg\max_{i \in [M]} f_i(x). \quad (1.1)$$

For the example of a 2-hidden layers Neural Network based classifier, [25], the vector valued discriminant function $f(x)$ is given as

$$f(x) = W^{(3)}\sigma\left(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}) + b^{(3)}, \quad (1.2)$$

4
where $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ denote the weight matrix and bias vector of layer $i$, and $\sigma$ is utilized to denote the non-linear activation function commonly selected as the Rectified Linear Unit (ReLU), which is applied point-wise and returns $y(x) = \max(0, x)$.

**Alternating Direction Method of Multipliers**

In this subsection, we present the use of the Alternating Direction Method of Multipliers (ADMM) algorithm, proposed in [26], to solve convex optimization problems with linear constraints.

Given the optimization program

$$\min_x D(x) \quad \text{subject to} \quad \mathbf{G}^T \mathbf{x} - \mathbf{b} \geq \mathbf{0},$$

that consists of a convex objective, given by $D(x)$, and linear inequality constraints, given by matrix $\mathbf{G}$ and vector $\mathbf{b}$. This convex problem can be written in the ADMM standard form as [27],

$$\min_{x,z} D(x) + E(z) \quad \text{subject to} \quad \mathbf{G}^T \mathbf{x} - \mathbf{b} - \mathbf{z} = \mathbf{0}.$$  

Vector $\mathbf{z}$ is added as a slack variable together with $E(z) = 0$ if $z \geq 0$, and $+\infty$ otherwise, to implement ADMM with equality constraints. Let $D(x) = \|x\|_2^2$. The augmented Lagrangian is

$$\mathcal{L}_\lambda(x, z, \mu) = \|x\|_2^2 + E(z) + \frac{\lambda}{2} \left( \| \mathbf{G}^T \mathbf{x} - \mathbf{b} - \mathbf{z} - \mu \|_2^2 - \| \mu \|_2^2 \right),$$

where $\lambda$ is a penalty factor and $\mu$ is the vector of Lagrangian multipliers. Then, the ADMM algorithm solves for $x$ by iteratively updating the following assignments.

$$x \leftarrow -\lambda (2\mathbf{I}_N + \lambda \mathbf{G}\mathbf{G}^T)^{-1} \mathbf{G}(\mathbf{b} + \mathbf{z} + \mu),$$

(1.6a)
\[ z \leftarrow \max \left( 0, G^T x - b + \mu \right), \]  
\[ \mu \leftarrow \mu + G^T x - b - z. \]  

The Maximum Independent Set Problem

In this dissertation, we consider the **NP-hard** problem of finding maximum independent sets (MIS). We define the MIS problem and the related maximum clique (MC) and minimum vertex cover (MVC) problems, then briefly describe how MC and MVC can be represented as instances of MIS.

**MIS Problem:** Given an undirected graph \( G = (V, E) \), MIS is the problem of finding a subset of vertices \( \mathcal{I} \subseteq V \) such that \( E(G[\mathcal{I}]) = \emptyset \), and \( |\mathcal{I}| \) is maximized.

**MC Problem:** Given an undirected graph \( G = (V, E) \), MC is the problem of finding a subset of vertices \( C \subseteq V \) such that \( G[C] \) is a complete graph, and \( |C| \) is maximized.

**MVC Problem:** Given an undirected graph \( G = (V, E) \), MVC is the problem of finding a subset of vertices \( R \subseteq V \) such that, for every \( (u, v) \in E \), either \( u \in R \) or \( v \in R \), and \( |R| \) is minimized.

For the MC problem, the MIS of a graph is an MC of the complement graph [28]. MVC and MIS are complementary, i.e., a vertex set is independent if and only if its complement is a vertex cover [29]. We exploit these properties in the development of our dNNs.
CHAPTER 2: ADVERSARIAL ATTACKS ON HIERARCHICAL CLASSIFIERS VIA CONVEX PROGRAMMING

Introduction

Adversarial attacks were shown to drastically damage the performance of one-stage classifiers while being undetectable. In this chapter, we examine the susceptibility of both flat and top-down hierarchical classifiers, abbreviated FHCs and TDHCs respectively, to non-targeted and targeted adversarial attacks. Furthermore, as a special case of multi-stage TDHC, we consider Nested Dichotomies Classifiers (NDCs), which decompose a multi-class problem into a collection of binary ones. Convex programs are formulated to generate input perturbations geared at fooling their output predictions. A competitive solver based on the Alternating Direction Method of Multipliers (ADMM) is developed and is shown to outperform state-of-the-art commercial solvers. The attacks developed for FHCs, TDHCs, and NDCs are evaluated based on their success rate and imperceptibility. It is shown that FHCs are inherently more robust than TDHCs to said attacks in the sense that fooling their coarse classification generally requires higher levels of perturbations.

Recent studies have primarily focused on OSCs. In this chapter, we depart from OSCs by considering Hierarchical Classifiers (HCs) where the samples to be classified are organized into a class hierarchy [33]. HCs, in which inputs are classified according to coarser and finer levels, are highly pertinent in various applications, including speech classification and computer vision [33, 34]. We consider the two main types of HCs. The first type is the Flat Hierarchical Classifier (FHC), where an observation is first classified according to the finer label, then the upper label (super-class) is obtained from the fine classification. Another common terminology for FHC models is the direct
approach, global, and bottom-up classifiers [33]. The second type is the Top-Down Hierarchical Classifier (TDHC) – also known as the coarse-to-fine classifier – where the feature vectors are classified in line with the coarser genres, followed by finer prediction as a result of the initial upper level classification. HCs have direct bearing on numerous important applications. Examples include text categorization, protein function prediction, musical genre classification, speech classification, computer vision, COVID-19 identification, marine benthic biota, satellite spectral images, and forensics [34],[33], [35], [36], [37], [38]. We refer the reader to the recent survey [39] and references therein for more details.

As a special case of TDHC, we investigate the impact of adversarial attacks on NDCs. Multi-class classifiers constructed from multiple stages of binary predictors include One-Vs.-One (OVO), One-Vs.-All (OVA), Error Correcting Output Codes (ECOC), and Nested Dichotomies Classifiers (NDCs) [40]. The importance of studying attacks on such classification systems is due to their high interpretability and their pervasiveness in Automated Machine Learning (AutoML) [41]. Compared to other decomposition techniques, NDCs afford flexible classifier structures with high predictive accuracy [42].

Related Work

The literature abounds with approaches for generating individual perturbations against OSCs in white-box settings. Optimization-based techniques, such as the Carlini and Wagner attack [43], the box-constrained Limited-Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) attack [44], Deepfool [45], and saliency map attack [19] generate adversarial examples by optimizing a cost function expressed in terms of the perturbation norm and/or the model’s loss subject to misclassifications of the adversarial examples. Other methods, such as the Fast Gradient Sign Method (FGSM) [46], compute the gradient of the loss function with respect to (w.r.t.) the input vector – which can be
computed efficiently using backpropagation – to generate perturbations. An iterative version (I-FGSM) is proposed in [47] to ensure the perturbations result in mistaking the input examples for less likely classes by taking iterative steps in the direction of the negative gradient of the loss function. An approach that integrates a momentum term (which accumulates previous gradients) into the iterative procedure to escape local maxima is presented in [11], thereby boosting the adversarial attacks. The approach proposed in [48] decouples the norm and the direction of the perturbation to avert the expensive iterations of optimization-based techniques. In order to constrain the norm of the adversarial perturbation while also ensuring it induces a mis-classification, the algorithm projects the generated perturbation on a sphere centered at the original example of varying radius. The elastic-net attack generates perturbations that achieve the twin objective of low $L_1$ distortion and good visual quality using regularization with a mixture of $L_1$ and $L_2$ penalty functions [49]. In sharp contrast, in this chapter, we take a principled approach in which we formulate constrained programs to craft adversarial perturbations capable of fooling the fine and/or coarse predictions in hierarchical settings, and develop solutions of various relaxations of said programs.

Hierarchical Classification Models

In this section, we formulate the FHC and TDHC models used in this dissertation. For the FHC whose block diagram is illustrated in Figure 2.1, we define the classifier function $k : \mathbb{R}^N \rightarrow [M]$ that maps the input feature vector $x \in \mathbb{R}^N$ to its predicted label out of $M$ candidate classes. We assume there exist $M$ discriminant functionals $J_l : \mathbb{R}^N \rightarrow \mathbb{R}, l \in [M]$, such that

$$k(x) = \arg\max_{l \in [M]} J_l(x).$$ (2.1)

The classification function in (2.1) represents the first stage of the FHC. In the second stage, a function $T : [M] \rightarrow [M_c]$ maps the predicted fine label from the first stage to a super-class $i \in [M_c]$. 

9
where $M_c$ is the total number of super-classes. For example, a fine prediction ‘cat’ is mapped to the upper-class ‘pet’. To characterize the inverse mapping, we define the super-class sets

$$S_i = \{ l \in [M] : T(l) = i \}, \quad i \in [M_c],$$

(2.2)

which group all the fine labels that belong to the same super-class $i$. For the TDHC model, we define the function $r : \mathbb{R}^N \to [M_c]$, which obtains the coarse prediction of the input $x$. To this end, we define $M_c$ discriminant functionals, $h_i(x) : \mathbb{R}^N \to \mathbb{R}, i \in [M_c]$, such that

$$r(x) = \arg\max_{i \in [M_c]} h_i(x).$$

(2.3)

Further, we define the mapping $f : \mathbb{R}^N \times [M_c] \to [M]$ as

$$f(x, i) = \arg\max_{l \in [M_i]} g_l^i(x),$$

(2.4)

which obtains the fine label $l \in [M]$ of the input sample $x$ given the coarse prediction $i \in [M_c]$ by maximizing a set of discriminant functionals $g_l^i : \mathbb{R}^N \to \mathbb{R}, l \in [M_i]$, where $M_i$ is the cardinality
of the super-class set $S_i$ defined in (2.2). Equations (2.3) and (2.4) represent the first and second stages of the TDHC, respectively. For example, an input feature vector is classified through $r$ as belonging to the super-class ‘pet’, which is subsequently classified through $f$ to a finer label ‘cat’ or ‘dog’. Figure 2.2 illustrates a block diagram of the TDHC model.

The direction of the arrows in Figure 2.3 reflects the direction of the classification for both the FHC and TDHC models.
Nested Dichotomies

In this subsection, we formulate the NDC model. For the multi-class problem of \( M > 2 \) candidate classes, a nested dichotomies classification system consists of \( M - 1 \) binary predictors arranged in a dendrogram structure [50]. Denote each binary estimator as \( C_i \) for node \( i \in [M - 1] \). For every class \( m \in [M] \), we use \( V(m) \) to denote the set consisting of all nodes from the root to the leaf node \( m \). We also define \( R(m) := \{C_k, l\}, \forall k \in V(m) \) and \( l \in \{0, 1\} \) to be the set of binary classifiers along with the binary decisions to reach leaf node \( m \). An example of an NDC with \( M = 5 \) is shown in Figure 2.4. For an instance \( x \in \mathbb{R}^N \), a probability mass function (PMF), \( q_k(x) \in [0, 1]^2, \forall k \in V(m) \), defined as

\[
q_k(x) := [\Pr(C_{k,0} \mid x) \quad \Pr(C_{k,1} \mid x)]^T,
\]

where \( \Pr(C_{k,0} \mid x) = 1 - \Pr(C_{k,1} \mid x) \), denotes the output of binary classifier of node \( k \). The entries of \( q_k(x) \) are denoted by \( q_k^{(j)}(x) \). We use \( q_k(x) \) and \( q_k \) interchangeably for simplicity of notation. Hence, the classification for every node \( k \in V(m) \) is given by

\[
g_k(x) = \arg\max_{j \in \{C_{k,0}, C_{k,1}\}} q_k^{(j)}(x).
\]
Based on the chain rule of probabilities [51], we use $Q \in [0,1]^M$ to express the overall probability assignment function for the multi-class problem, which also represents the overall PMF. The entries of $Q$ are

$$Q_m(x) = \Pr(m \mid x) = \prod_{j \in R(m)} \Pr(j \mid x), \ m \in [M],$$

such that the predicted label $w(x)$ for the multi-class problem is obtained by

$$w(x) = \arg\max_{m \in [M]} Q_m(x).$$

**Attack Formulations**

In terms of constraining the minimum allowable perturbations, attacks are categorized as restricted (perturbations of a fixed size), and unrestricted attacks. Restricted methods can be used to solve the unrestricted case by attempting a set of bounds with some step size. This makes it a function of the steps and bounds. Thus, the unrestricted case (the one considered in our formulations) is more general. In this section, first, we formulate the optimization problems for generating unrestricted adversarial attacks against the models considered in the previous section.

For the FHC model, attacks are formulated to induce misclassifications of the super-classes – for example, enforcing the classification of a ‘dog’ to a super-label ‘vehicle’. For the TDHC, we formulate attacks targeting both the coarser or the finer levels.

In designing perturbations $\eta \in \mathbb{R}^N$ to be added to the input samples, imperceptibility, which measures the distance between the perturbed and the original observations, plays a key role in determining the quality and efficiency of the attack. The attacker seeks to ensure that the perturbed input is sufficiently similar to the original unperturbed example, with respect to (w.r.t.) some distance.
function $D(x, x + \eta)$. In our formulations, we utilize

$$D(x, x + \eta) = D(\eta) := \|x + \eta - x\|^2 = \|\eta\|^2.$$  \hfill (2.9)

To change the classification to a specified target, we encode constraints in our convex programs on the discriminant functionals $J_i(x + \eta)$, $h_i(x + \eta)$, and $g_i(x + \eta)$ with perturbations. To maintain convexity, we make use of the first order Taylor series expansion \cite{52},

$$J(x + \eta) \approx J(x) + \eta^T \nabla_x J(x),$$ \hfill (2.10)

which results in linear constraints.

For the FHC, the goal of the attacker is to change the super classification to a target super-class set $S_t$, i.e.,

$$k(x + \eta) \in S_t.$$ \hfill (2.11)

Given the definition in (2.2), this translates into the requirement that

$$\exists j \in S_t : J_j(x + \eta) > J_l(x + \eta), \forall l \in S'_t,$$ \hfill (2.12)

where $S'_t$ is the complement of set $S_t$. In other words, the attacker seeks to ensure that the discriminant functional of some label in $S_t$ dominates all those in the complement set. A key challenge in satisfying the requirement (2.12) lies in the choice of the class $j \in S_t$. Next, we propose two approaches to address this challenge.

The first is an algorithmic approach which we call the targeted set FHC-Algorithmic approximation (FHC-A). The idea is to iterate over all labels inside the targeted super-class set $S_t$ to obtain the
Algorithm 1 FHS-A Attack

Input: $x, k, T, D$.
Output: $\eta^*$

1: for $j \in S_t$
2: find $\eta_j$ as the solution of (2.13)
3: if $k(x + \eta_j) \in S_t$
4: find $D(\eta_j)$
5: $\eta^* = \arg\min_{j \in S_t'} D(\eta_j)$

minimum perturbation ensuring the condition in (2.11) is met. Hence, for every $j \in S_t$, we solve

$$\min_{\eta} D(\eta) \quad \text{subject to} \quad \eta^T (\nabla_x J_j(x) - \nabla_x J_l(x)) \geq J_l(x) - J_j(x) + \epsilon_b, \forall l \in S_t'.\quad (2.13)$$

The arbitrarily small constant $\epsilon_b > 0$ is used to transform the strict inequalities (2.12) to bounded ones, thereby avoiding an open set of feasible solutions [53]. The procedure is presented in Algorithm 2, which solves $|S_t|$ convex programs. At each iteration, the minimization includes $|S_t'|$ linear constraints.

The second approach uses the label in the target super-class set $S_t$ associated with the largest discriminant functional prior to adding perturbations. Specifically, since the functionals $J_i, i \in [M]$, represent the class membership, we define $j^*$ as

$$j^* = \arg\max_{j \in S_t} J_j(x),\quad (2.14)$$

which is then used to encode constraints in the convex program (2.15) to generate perturbations.

$$\min_{\eta} D(\eta) \quad \text{subject to} \quad \eta^T (\nabla_x J_{j^*}(x) - \nabla_x J_l(x)) \geq J_l(x) - J_{j^*}(x) + \epsilon_b, \forall l \in S_t'.\quad (2.15)$$
We call the generated attack the targeted set FHC-Nearest label approximation (FHC-N) attack. This attack requires solving only one convex program with $|S_t|$ linear constraints.

**Proposition 1.** Let $\eta_A^*$ and $\eta_N^*$ be the optimal solutions of Algorithm 1 and the minimization problem (2.15), respectively. Then $D(\eta_A^*) \leq D(\eta_N^*)$.

The proof of Proposition 1 follows directly from the fact that Algorithm 1 solves (2.13) for every $j \in S_t$, including $j^*$ in (2.14), and picks the minimizing perturbation in its last step.

For the TDHC, given the class hierarchy of the TDHC model, inducing a misclassification of the coarser level will clearly result in fooling the finer classification. However, the converse is not necessarily true. As such, we consider the following two scenarios.

In the first scenario, the attack is geared at altering the coarser classification to a target label, irrespective of the finer level classification. Define $t_c \in [M_c] \{r(x)\}$ as the target coarser class. Hence, the perturbation is generated to enforce the requirement

$$r(x + \eta) = t_c.$$  \hspace{1cm} (2.16)

From (2.3), this reduces to the requirement

$$h_{t_c}(x + \eta) > h_l(x + \eta), \forall l \in [M_c] \{t_c\}.$$  \hspace{1cm} (2.17)

Hence, we formulate the convex program

$$\min_{\eta} D(\eta) \quad \text{subject to}$$

$$\eta^T (\nabla_x h_{t_c}(x) - \nabla_x h_l(x)) \geq h_l(x) - h_{t_c}(x) + \varepsilon_c, \forall l \in [M_c] \{t_c\}.$$  \hspace{1cm} (2.18)

to generate the targeted TDHC-Coarser level (TDHC-C) attack. The program (2.18) consists of
$M_c - 1$ linear constraints, where the constant $\varepsilon_c > 0$ is used to relax the strict inequalities to bounded inequalities as in (2.13).

In the second scenario, the attacker’s goal is to generate a perturbation so as to alter the finer prediction to a target fine label $t_f$ while also preserving the coarse level prediction. Hence, $\eta$ is generated such that,

$$r(x + \eta) = r(x), \quad (2.19a)$$

$$f(x + \eta, r) = t_f. \quad (2.19b)$$

The requirement (2.19a) to preserve the coarse prediction is equivalent to

$$h_r(x + \eta) > h_k(x + \eta), \forall k \in [M] \setminus \{r(x)\}, \quad (2.20)$$

where $r$ is used as short for $r(x)$. Confusing the finer label to target $t_f$ as in (2.19b) amounts to the requirement

$$g'_{t_f}(x + \eta) > g'_l(x + \eta), \forall l \in [M_r] \setminus \{t_f\}. \quad (2.21)$$

Based on (2.20) and (2.21), the targeted TDHC-Finer level attack (TDHC-F) is formulated through the minimization,

$$\min_{\eta} D(\eta) \quad \text{subject to}$$

$$\eta^T(\nabla_{x}h_r(x) - \nabla_{x}h_k(x)) \geq h_k(x) - h_r(x) + \varepsilon_c, \forall k \in [M_c] \setminus \{r(x)\}, \quad (2.22)$$

$$\eta^T(\nabla_{x}g'_{t_f}(x) - \nabla_{x}g'_l(x)) \geq g'_l(x) - g'_{t_f}(x) + \varepsilon_f \forall l \in [M_r] \setminus \{t_f\}.$$  

For every feature vector, it is required to solve a convex program with $M_c + M_r - 2$ linear constraints. The constant $\varepsilon_f$ is used for a similar purpose as $\varepsilon_b$ and $\varepsilon_c$.

Next, we formulate an optimization problem to obtain imperceptible additive perturbations, $\eta \in $
In terms of the induced PMF, condition (2.23) can be expressed as

$$\exists m \in [M] \setminus \{w\} : Q_m(x + \eta) > Q_w(x + \eta).$$

(2.24)

Using the definition of $Q$ in (2.7), we formulate the following program.

$$
\min_{\eta} D(\eta) \quad \text{subject to} \\
\exists m \in [M] \setminus \{w\} : \Pr(j \mid x + \eta) > 0.5, \; \forall j \in R(m).
$$

(2.25)

The constraints in (2.25) are to ensure that there exists some label $m$, other than the predicted one, such that all nodes of the binary predictors $i \in V(m)$, make decisions favoring the path $R(m)$.

Next, we propose an algorithmic solution to (2.25). The algorithm iterates over all labels, $m \in [M] \setminus \{w\}$, and obtains the smallest perturbation for which condition (2.23) is met. Hence, for each $m \in [M] \setminus \{w\}$, we solve the program,

$$
\min_{\eta} D(\eta) \quad \text{subject to} \quad \Pr(j \mid x + \eta) > 0.5, \; \forall j \in R(m),
$$

(2.26)

to generate perturbations for each possible class, and then choose the minimum with respect to (w.r.t.) $D$. Algorithm 2 presents the procedure and Theorem 1 establishes the correctness of Algorithm 2.

**Theorem 1.** Let $\eta_{\text{alg}}^*$ be the output of Algorithm 2. If (2.25) is feasible, then $\eta_{\text{alg}}^*$ is an optimal solution of (2.25).
**Algorithm 2** Algorithmic Solution of (2.25)

**Input:** $\mathbf{x}, w, D$

**Output:** $\eta^*$

1. for $m \in [M] \setminus \{w\}$
2. find $\eta_m$ from (2.26) (for approximation, we solve (2.27)).
3. if $w(x) \neq w(x + \eta_m)$
4. find $D(\eta_m)$
5. $\eta^* = \arg\min_{m \in [M] \setminus \{w\}} D(\eta_m)$

**Proof.** Let $\eta^*$ be an optimal solution of (2.25), and for the sake of contradiction, assume that $\eta_{\text{alg}}^*$ is not optimal. Then, $D(\eta^*) < D(\eta_{\text{alg}}^*)$. Since $\eta^*$ is a feasible solution of (2.25), it satisfies (2.23), i.e., $\exists m_0 \in [M] \setminus \{w\} : \Pr(j \mid x + \eta^*) > 0.5, \forall j \in R(m_0)$. Thus, the feasible set of (2.25) is a subset of the feasible region of (2.26) with $m_0$. It follows that $D(\eta_{m_0}^*) \leq D(\eta^*)$, where $\eta_{m_0}^*$ is the optimal solution to (DA$_{m_0}$). Algorithm 2 solves (2.26) for all $m \in [M] \setminus \{w\}$ and chooses the one corresponding to the smallest value of the objective (step 5 of the Algorithm), thus $D(\eta_{\text{alg}}^*) \leq D(\eta_{m_0}^*)$, $\forall m \in [M] \setminus \{w\}$. Hence, $D(\eta_{\text{alg}}^*) \leq D(\eta_{m_0}^*) \leq D(\eta^*)$, yielding a contradiction. Therefore, $\eta_{\text{alg}}^*$ is an optimal solution of (2.25).

Since the constraints in (2.26) are generally non-convex, similar to formulations of the FHC and TDHC methods, we make use of the first order Taylor series expansion [52], $\Pr(j \mid x + \eta) \approx \Pr(j \mid x) + \eta^T \nabla_x \Pr(j \mid x)$, so (2.26) becomes

$$
\min_{\eta} D(\eta) \quad \text{subject to} \quad \Pr(j \mid x) + \eta^T \nabla_x \Pr(j \mid x) \geq 0.5 + \epsilon, \forall j \in R(m). \tag{2.27}
$$

The problem in (2.27) is a convex problem since the $|R(m)|$ constraints are linear in $\eta$. The constant $\epsilon > 0$ is added to avoid the strict inequality that leads to an open set of feasible solutions [53].
ADMM-based Solution

In this subsection, we propose the use of ADMM to solve the convex problems presented in this chapter. Following the formulation given in (1.4) and the preceeding iterative approach to solve for the primal and dual, here, we only provide the encoding of matrix $G$ and vector $b$.

For the FHC and TDHC, the encoding is given in $G \in \mathbb{R}^{N \times D_1}$ and $b \in \mathbb{R}^{D_1}$ as,

$$G = \begin{cases} 
\left[ \nabla_x J_j(x) - \nabla_x J_l(x) \right], & \forall l \in S'_f, \text{FHC-A iteration } j \\
\left[ \nabla_x J_j'(x) - \nabla_x J_l'(x) \right], & \forall l \in S'_f, \text{FHC-N} \\
\left[ \nabla_x h_t(x) - \nabla_x h_l(x) \right], & \forall l \in [M_c] \setminus \{t_c\}, \text{TDHC-C} \\
\left[ \nabla_x h_r(x) - \nabla_x h_k(x), \nabla_x g_r'(x) - \nabla_x g_l'(x) \right], & \forall k \in [M_c] \setminus \{r\}, \forall l \in [M_r] \setminus \{t_f\}, \text{TDHC-F}. 
\end{cases} \quad (2.28)$$

$$b = \begin{cases} 
\left[ J_l(x) - J_j(x) + \epsilon_b \right]^T, & \forall l \in S'_f, \text{FHC-A iteration } j \\
\left[ J_l(x) - J_j'(x) + \epsilon_b \right]^T, & \forall l \in S'_f, \text{FHC-N} \\
\left[ h_l(x) - h_t(x) + \epsilon_c \right]^T, & \forall l \in [M_c] \setminus \{t_c\}, \text{TDHC-C} \\
\left[ h_k(x) - h_r(x) + \epsilon_c, g_r'(x) - g_l'(x) + \epsilon_f \right]^T, & \forall k \in [M_c] \setminus \{r\}, \forall l \in [M_r] \setminus \{t_f\}, \text{TDHC-F}. 
\end{cases} \quad (2.29)$$

The dimension $D_1$ is given by

$$D_1 = \begin{cases} 
|S'_f|, & \text{for the FHC attacks,} \\
M_c - 1, & \text{for the TDHC-C attack,} \\
M_c + M_r - 2, & \text{for the TDHC-F attack.} 
\end{cases} \quad (2.30)$$

In the case of the NDC formulation, the columns of $G \in \mathbb{R}^{N \times |R(m)|}$ and the entries of $b \in \mathbb{R}^{|R(m)|}$.
are combined for every $j \in R(m)$ as follows,

$$G = \left[ \nabla_x \Pr(j \mid x) \right], \forall j \in R(m), \quad (2.31)$$

$$b = \left[ 0.5 + \varepsilon - \Pr(j \mid x) \right]^T, \forall j \in R(m). \quad (2.32)$$

Experimental Results

In this chapter, we use the following evaluation metrics. For perceptibility of the added disturbances, we use the relative perturbation $\rho_p = \frac{\| \eta \|_p}{\| x \|_p}$. Since we implement an image classification system, we also utilize the Structural Similarity Index (SSIM) $I \in [0, 1]$ as a measure of similarity, where $I = 1$ indicates two identical images, which captures luminance, contrast, and structure in the measurements [54]. For the overall performance, we first use the attack success rate $\alpha = \frac{N_s}{|\mathcal{X}|}$, defined as the fraction of the number of feature vectors that are misclassified when perturbations are added to the total number of observations of interest. Second, we use $\sigma_p$ and $\sigma_s$ to denote the average relative perturbation and similarity over the set of observations $\mathcal{X}$, respectively. In targeted attacks, the performance is highly dependent on the selection of the target label/set for each input. Therefore, the selection mechanism used needs to be specified as we examine the efficiency of a given attack. There are three cases for selecting the target class/superclass. The first is the average case in which the targets are selected uniformly at random then fixed. The second (third) case is the best (worst) case scenario in which for every example, all target labels/sets are tried and the one yielding the minimum (maximum) perturbation is selected [43]. In this chapter, we use the average case scenario.

We remark that, while we apply our proposed methods on a hierarchical image classification model, our formulations can be applied to any classification model in which the attacker has access to the
discriminant functionals and their gradients w.r.t. the input feature vectors.

The MNIST fashion dataset [2] is used, where each observation vector is a grayscale $28 \times 28$ pixels image. The labels 0 to 9 correspond to: ‘T-shirt’, ‘Trouser’, ‘Pullover’, ‘Dress’, ‘Coat’, ‘Sandal’, ‘Shirt’, ‘Sneaker’, ‘Bag’, and ‘Ankle boot’, respectively. The super-class sets are ‘top’: $\{0, 2, 6\}$, ‘bottom’: $\{1, 5, 7, 9\}$, and ‘other’: $\{3, 4, 8\}$. We train Convolutional Neural Networks (CNNs) consisting of 8 layers with $L_c$ outputs representing the number of the discriminant functionals as shown in Table 2.1. The same CNN architecture is used for all five classifiers: FHC $k(\cdot)$ with $L_c = 10$, TDHC coarse $r(\cdot)$ with $L_c = 3$, TDHC ‘top’ fine $f(\cdot, 1)$ with $L_c = 3$, TDHC ‘bottom’ fine $f(\cdot, 2)$ with $L_c = 4$, and TDHC ‘other’ fine $f(\cdot, 3)$ with $L_c = 3$. For the TDHC, the classification accuracy of the four classifiers are 95.29%, 86.2%, 97.4%, and 97.33%, respectively. The trained CNN for the FHC scores a classification accuracy of $CA = 90.09\%$ and a super classification accuracy $CA^{sup} = 95.51\%$ for the test dataset $X$ with $|X| = 10,000$. For the Nested Dichotomies example, the details are provided in Figure 2.5. For the FHC model, we proposed two attacks.

Table 2.1: CNN Architecture w.r.t the output size $L_c$.

<table>
<thead>
<tr>
<th>Layer</th>
<th>kernel Size</th>
<th>Layer Output Shape</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reshape</td>
<td>N/A</td>
<td>$28 \times 28 \times 1$</td>
<td>0</td>
</tr>
<tr>
<td>Conv2D+ReLU</td>
<td>$5 \times 5 \times 1 \times 32$</td>
<td>$24 \times 24 \times 32$</td>
<td>832</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td>N/A</td>
<td>$12 \times 12 \times 32$</td>
<td>0</td>
</tr>
<tr>
<td>Conv2D+ReLU</td>
<td>$5 \times 5 \times 32 \times 64$</td>
<td>$8 \times 8 \times 64$</td>
<td>51264</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td>N/A</td>
<td>$4 \times 4 \times 64$</td>
<td>0</td>
</tr>
<tr>
<td>Flatten</td>
<td>N/A</td>
<td>1024</td>
<td>0</td>
</tr>
<tr>
<td>Dense+Softmax</td>
<td>$1024 \times L_c$</td>
<td>$L_c$</td>
<td>$1024L_c$</td>
</tr>
</tbody>
</table>

Table 2.2 presents the performance of FHC-A and FHC-N in terms of success rate $\alpha$, perceptibility $\sigma_2$, and the execution time over the batch of feature vectors of interest. FHC-A outperforms FHC-N with regard to success rate and detectability. The observed difference in the average perceptibility verifies the result of Proposition 1. FHC-A has a longer execution time (approximately three times slower for 1000 trials) since Algorithm 1 iterates over all $j \in S_t$ to obtain the minimum perturbations required to alter the prediction to the target set.
Figure 2.5: Proposed NDC structure for MNIST fashion dataset with $M = 10$ leaf nodes (underlined labels). Sets $R(m)$ and $V(m), \forall m \in [10]$ are presented. The classification accuracy for every binary classifier $C_k, \forall k \in [9]$, is given at the bottom. The overall classification accuracy is 89.2%. Grouping of the leaf node classes is based on the location and the shape of the item.

Table 2.2: FHC attacks averaged over 1000 trials.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$\alpha$ (%)</th>
<th>$\sigma_2$ (%)</th>
<th>run-time over trials (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHC-A (Algorithm 1)</td>
<td>87.9</td>
<td>17.89</td>
<td>456.9</td>
</tr>
<tr>
<td>FHC-N (2.15)</td>
<td>81.71</td>
<td>20.29</td>
<td>146.7</td>
</tr>
</tbody>
</table>

Figure 2.6a and Figure 2.6b show samples for each super/coarser class using FHC-A and TDHC-C attacks, respectively. The images along the diagonal represent the original unperturbed images. The rows and columns represent the ground truth and targeted super-labels, respectively. Each figure shows the images of the perturbed samples that are classified according to the predefined target, along with the corresponding perceptibility measure. For instance, altering ‘top’ to ‘bottom’ using FHC-A has perceptibility $\rho_2 = 17.3\%$, while changing ‘top’ to ‘other’ only required $\rho_2 = 8.82\%$ as shown in the first row of Figure 2.6a. Also, it is visually evident that larger perturbation is used to alter ‘top’ to ‘bottom’ for this particular image. Similar results are observed in the remaining rows and for the TDHC-C attack in Figure 2.6b. Noting that the targeted classes for both attacks are identical for each sample from the MNIST testing dataset under the average case scenario in selecting target labels, we see that it is generally harder to fool the FHC than the TDHC for the selected samples.
<table>
<thead>
<tr>
<th>Ground-Truth Super Labels</th>
<th>Targeted Super Labels</th>
<th>Targeted Coarser Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>“top”</td>
<td>“top”</td>
<td>“top”</td>
</tr>
<tr>
<td>Original</td>
<td>Original</td>
<td>Original</td>
</tr>
<tr>
<td>( \rho_2 = 17.3%)</td>
<td>( \rho_2 = 14.54%)</td>
<td>( \rho_2 = 12.6%)</td>
</tr>
<tr>
<td>“bottom”</td>
<td>“bottom”</td>
<td>“bottom”</td>
</tr>
<tr>
<td>Original</td>
<td>Original</td>
<td>Original</td>
</tr>
<tr>
<td>( \rho_2 = 8.82%)</td>
<td>( \rho_2 = 7.21%)</td>
<td>( \rho_2 = 5.9%)</td>
</tr>
<tr>
<td>“other”</td>
<td>“other”</td>
<td>“other”</td>
</tr>
<tr>
<td>( \rho_2 = 21.11%)</td>
<td>( \rho_2 = 14.34%)</td>
<td>( \rho_2 = 14.34%)</td>
</tr>
</tbody>
</table>

(a) Samples of the FHC-A attack.

(b) Samples of the TDHC-C attack.

Figure 2.6: Samples of altering the super (coarser) label according to a target super (coarser) class for the FHC-A (TDHC-C) attack with perceptibility measure \( \rho_2 \).

![Figure 2.6](image_url)

Figure 2.7: Perturbations needed to alter the super/coarser classification for the first 100 samples of the MNIST fashion dataset represented by SSIM (left) and the CDF of the SSIM (right).

![Figure 2.7](image_url)

We also show the similarity index between the original and perturbed examples. Figure 2.7 (left) presents the SSIM for the first 100 samples of the test MNIST fashion dataset, and Figure 2.7 (right) shows the Cumulative Distribution Function (CDF) of the SSIM. As shown, the perturbations required to alter the coarse level prediction are higher in the case of the FHC model in comparison with the TDHC model for the same target labels. Hence, FHC-A is inherently more robust to the...
attacks derived in this chapter.

Figure 2.8 presents samples from each coarse class for the TDHC-F attack along with the amount of perturbation represented by the perceptibility measure $\rho_2$. As observed, TDHC-F is successful in maintaining the coarse level prediction while altering the finer classification to the pre-defined finer target class. For example, in the first column, it is shown that TDHC-F alters the finer classification from ‘T-shirt’ to ‘pullover’ while preserving the coarse classification as ‘top’, yielding perceptibility of $\rho_2 = 11.76\%$. A similar behavior is observed in the second and third columns for samples from ‘bottom’ and ‘other’ with $\rho_2 = 13.22\%$ and $\rho_2 = 23.74\%$, respectively. Table 2.3 compares our proposed ADMM solver with the state-of-the-art CVXPY solver [6]. We use the term eADMM to note that we use an enhanced version where we check for the attack success criteria at each iteration. eADMM consistently yields gains in the success rate and imperceptibility tradeoff. For FHC-A (TDHC-C), we observe that, with approximately similar success rate
Table 2.3: eADMM comparison with state-of-the-art solver CVXPY [6] w.r.t the attacks scenarios over 1000 trials.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Solver</th>
<th>α(%)</th>
<th>σ²(%)</th>
<th>σₛ(%)</th>
<th>run-time(sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHC-A</td>
<td>eADMM</td>
<td>87.9</td>
<td>17.89</td>
<td>77.64</td>
<td>465.9</td>
</tr>
<tr>
<td>FHC-A</td>
<td>CVXPY[6]</td>
<td>87.4</td>
<td>21.07</td>
<td>71.49</td>
<td>140.7</td>
</tr>
<tr>
<td>TDHC-C</td>
<td>eADMM</td>
<td>95.8</td>
<td>16.36</td>
<td>81.99</td>
<td>126.8</td>
</tr>
<tr>
<td>TDHC-C</td>
<td>CVXPY[6]</td>
<td>95.3</td>
<td>22.43</td>
<td>73.86</td>
<td>38.94</td>
</tr>
<tr>
<td>TDHC-F</td>
<td>eADMM</td>
<td>87.3</td>
<td>19.45</td>
<td>79.77</td>
<td>314.5</td>
</tr>
</tbody>
</table>

Table 2.4: Overall Performance of Algorithm 2 and MIFGSM.

<table>
<thead>
<tr>
<th>Model</th>
<th>Attack</th>
<th>CA(%)</th>
<th>α(%)</th>
<th>σ₂</th>
<th>σₛ</th>
<th>time (min.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDC</td>
<td>Algorithm 2</td>
<td>89.2</td>
<td>88.2</td>
<td>0.15</td>
<td>0.87</td>
<td>35</td>
</tr>
<tr>
<td>MOSC</td>
<td>MIFGSM [11]</td>
<td>90.09</td>
<td>100</td>
<td>0.06</td>
<td>0.92</td>
<td>61</td>
</tr>
</tbody>
</table>

α ≈ 88% (α ≈ 95%), eADMM yields smaller preceptibility σ₂ ≈ 18% (σ₂ ≈ 16%) and σₛ ≈ 78% (σₛ ≈ 82%). For the TDHC-F, the eADMM solver outperforms CVXPY in both perceptibility and attack success rate. CVXPY incurs a shorter execution time since eADMM searches for a suitable solution at each ADMM iteration. Irrespective of the used solver, comparing the execution time of TDHC-C (126.8 and 38.94 seconds) with that of TDHC-F (314.5 and 112.42 seconds), we see that altering the fine level prediction while preserving the coarse classification incurs longer execution time. This is due to the additional linear constraints in program (2.22) in comparison with (2.18).

Next, we present the attack results against NDCs. Figure 2.9 (top) presents a sample feature vector from each class of the MNIST fashion dataset along with the true label $w^*(x)$. The corresponding perturbed images using Algorithm 2 are shown in Figure 2.9 (bottom), along with the SSIM value (at the top of every image), and the predicted label after perturbation ($w(x + \eta)$). For example, for the first image labeled as ‘T-shirt’, the perturbation by Algorithm 2 induces a misclassification as an ‘Ankle-Boot’ with $SSIM = 79\%$. Similar observations are seen for all the other samples.

Table 2.4 presents the performance of our algorithm in terms of the success of the attack (represented by α), perceptibility measures (represented by σₛ and σ₂), and execution time. Results for the state-of-the-art MIFGSM to fool the MOSC are also presented. We observe that our attack is
Table 2.9: Samples of each class of the MNIST fashion dataset (top), the corresponding perturbed versions from MIFGSM (middle) where $\eta_M$ represents the generated perturbations from MIFGSM, and the corresponding perturbed versions from Algorithm 1 (bottom).

![Table Image](image)

Figure 2.10: SSIM ($I$) of the first 100 examples with MIFGSM and our proposed Algorithm (left) and CDF of $I$ (right). The lines in the left plot are the smoothed average for each case with moving average window of length 25.

successful at changing the classification of the NDC with $\alpha = 88.2\%$ while being as imperceptible as possible according to the last step of the algorithm. This is evidenced by the reported values $\sigma_2 = 0.15$ and $\sigma_s = 0.87$. Furthermore, we observe that MOSC attack is more successful at fooling the classification task of the trained model. This is noted in both the success ratio and the perceptibility measures. This means that, while NDCs are a harder to build and train, they are more robust to adversarial attacks in comparison to MOSCs. Indeed, disturbances fooling NDCs are more perceptible than the ones fooling MOSCs. In addition, our attack requires less computational resources to implement in comparison to MIFGSM and MOSC as shown by the reported execution
The requirement of higher perturbations to fool NDCs is seen in Figure 2.10 where the SSIM values are reported together with the Conditional Density Function (CDF) of SSIM. This is observed in the high values of MIFGSM vs. MOSC case (less imperceptibility) and the lower values of SSIM in the Algorithm 2 vs. NDC case (more imperceptibility).

Discussion and Conclusion

One-Stage Classifiers (OSCs) have been the main focus of previous studies on adversarial attacks. In this chapter, we developed methods to generate attacks on two types of Hierarchical Classifiers, FHC and TDHC (with considering the special case of Nested Dichotomies). We formulated convex programs to generate perturbations using the discriminant functionals and developed a competitive solver based on the ADMM. We have shown that perturbations required to alter the coarse predictions to specific labels in FHC models are generally larger than those needed in TDHC models. In other words, a FHC is inherently more robust to such attacks. Also, it is easier to induce coarse misclassifications than to alter the fine prediction to a pre-specified target class while preserving the coarse level prediction. It was also shown that fooling NDCs requires more perceptible disturbances than those needed to alter the prediction of Multi-class One-Stage Classifiers.
CHAPTER 3: ON THE COARSE ROBUSTNESS OF CLASSIFIERS

Introduction

A common approach to gauge the robustness of a certain classifier to adversarial attacks revolves around determining the least amount of perturbation capable of inducing a misclassification – e.g., see the CLEVER metric [55] and the work in [43, 19, 11, 30, 56] for some of the state-of-the-art attack and robustness evaluation methods.

Given a trained classifier, most previous works have focused on metrics based on the standard notion of robustness that make use of state-of-the-art adversarial attack methods such as [43, 46, 19, 11]. However, since this approach generally ignores the semantic relations among the classes, it falls short of capturing a measure of severity of the induced misclassifications, and in turn of the safety and brittleness of the classifier design. For example, misclassifying ‘Can’ as ‘Cup’ is less drastic than classifying ‘Truck’ as ‘Cat’ (all distinct categories in CIFAR-10 [3]).

In this chapter, we introduce a new notion of coarse robustness which captures the susceptibility of a classifier to perturbations inducing misclassifications of the coarse labels with respect to (w.r.t.) a specified grouping of the class labels. A clear advantage is that the derived measures could be used to gauge the ability of a classifier to recognize the coarse genres w.r.t. the grouping.

In addition to the proposed measures, a training approach that integrates the coarse information is introduced. When compared to conventional training, the introduced coarse training approach is proven to obtain stronger classifiers as leveraged by the introduced measures that evaluate the coarse robustness w.r.t. consistent groupings of the class labels.

We make the following contributions. First, this work introduces a new notion of global coarse ro-
bustness of a classifier which captures the hardness of confusing the coarse predictions induced by a certain grouping of the class labels. Second, to find suboptimal groupings, we propose measures of goodness of class groupings and formulate a mathematical program to optimize such measures. An algorithmic procedure is developed based on proposed measures, shown to perform on par with combinatorial brute force search, while requiring a considerably smaller number of iterations. The utility of the introduced measures and the performance of the search algorithm are demonstrated using image classification on known benchmark datasets. Third is the coarse training approach for obtaining improved coarse robustness models. In comparison to Natural Training (NT), we prove that our approach obtains stronger models in terms of the introduced coarse robustness measure. While requiring the same training time and small relative additional parameters, experiments show that our approach produces improved models when compared to models trained using NT\(^1\).

The Coarse Robustness Measure

In this section, we first define the classification model and the conventional way of training a neural network based classifier. Define the classifier \(h : \mathbb{R}^N \rightarrow [M]\), which maps an observation \(x \in \mathbb{R}^N\) to one of \(M\) possible (fine) labels. The predicted label is obtained as the index of the maximizing discriminant probabilistic functional in vector \(f : \mathbb{R}^N \rightarrow \Delta^M\) where \(\Delta^M\) is the probability simplex, with entries \(f_m, m \in [M]\), as \(h(x) = \arg\max_{m \in [M]} f_m(x)\).

**Definition 1.** Given classifier \(h\), parameterized by \(\theta\), Natural Training (NT) on dataset \(\mathcal{D}\) with entries in the form \((x, 1_y)\), where \(y \in [M]\) is the true label of \(x\) and \(1_y\) is one hot encoding vector representation of \(y\), can be defined as the task of minimizing the loss in

\[
\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x, 1_y) \in \mathcal{D}} \mathcal{L}(f(x; \theta), 1_y). \tag{3.1}
\]

\(^1\)The material in this chapter is, in part, based on our published paper at the 2022 Asilomar Conference [57].
Let \( T : [M] \to [M_c] \) (same as the one defined in the previous chapter) be a grouping function that maps the classifier’s output to a coarse label \( i \in [M_c] \), where \( M_c < M \) is the number of coarse classes. The function \( T \) induces the coarse class sets \( S_i \) as defined in (2.2). A standard measure of robustness of classifier \( h \) w.r.t. a feature vector \( x \) is the least amount of perturbation (relative to some norm) \( \eta \in \mathbb{R}^N \), required to produce a false prediction. It is defined as

\[
\eta^*_m(x) := \arg\min_\eta \{ \| \eta \|_p : h(x + \eta) \neq h(x) \}.
\] (3.2)

**Definition 2.** We define a Global Standard Robustness (GSR) measure in vector \( g_p \in \mathbb{R}^M \) whose entries represent the average of the \( l_p \) distances between examples \( x \in \mathbb{R}^N \) and their misclassified perturbed versions \( \hat{x} = x + \eta \), i.e.,

\[
g_p(m) := \frac{1}{|D_m|} \sum_{x \in D_m} \| \eta^*_m(x) \|_p ,
\] (3.3)

where \( D_m := \{x \in \mathbb{R}^N : h(x) = m\} \).

Given that \( h(x) = m \) with the predicted coarser set \( S_{T(m)} \), one could define the minimum perturbation required to induce misclassification of the coarse label (coarse misclassification) as

\[
\eta_m(x; T) := \arg\min_\eta \{ \| \eta \|_p : h(x + \eta) \notin S_{T(m)} \} .
\] (3.4)

For a global representation, we introduce the next definition to describe a Global measure of Coarse Robustness (GCR).

**Definition 3.** Let vector \( c_p(T) \in \mathbb{R}^M \), whose entries

\[
c_p(m; T) := \frac{1}{|D_m|} \sum_{x \in D_m} \| \eta_m(x; T) \|_p ,
\] (3.5)
reflect the average minimum perturbations required to cause coarse misclassification from predicted label \( m \), used to define our global measure of coarse robustness, dubbed GCR.

Finding Best Mappings

In this section, we use targeted perturbations from label \( m \) to target \( n \),

\[
\eta_{mn}(x) := \arg\min_{\eta} \|\eta\|_p : h(x) = m, h(x + \eta) = n \, ,
\]

(3.6)

to find best mappings w.r.t. the notion of coarse robustness. As such, we first define the matrix

\[
G_p \in \mathbb{R}_+^{M \times M}
\]

with zero diagonal, whose entries represent the average of the \( l_p \) distances between examples \( x \in \mathbb{R}^N \) and their misclassified perturbed versions \( \hat{x} = x + \eta \), i.e.,

\[
G_p(m, n) = \frac{1}{|D_m|} \sum_{x \in D_m} \|\eta_{mn}(x)\|_p \, , \ m \neq n \, .
\]

(3.7)

Given \( T \), we define matrix \( C_p \in \mathbb{R}_+^{M_c \times M_c} \) whose entries,

\[
C_p(i, j; T) = \frac{1}{|S_i||S_j|} \sum_{m \in S_i, n \in S_j} G_p(m, n) \, ,
\]

(3.8)

represent the average of the minimum perturbations to induce a misclassification of the coarse label from set \( S_i \) to set \( S_j \). The diagonal of \( C_p \) is a measure of the average perturbations required to induce misclassifications within the same coarse set \( S_i, i \in [M_c] \). In order to obtain the best grouping function \( T \), for every coarse label \( i \in [M_c] \), it requires on average a higher level of perturbation to misclassify the coarse labels (inter-set misclassification) than to misclassify the original prediction within the same coarse set (intra-set misclassification). Further, it is desirable that the mapping holds a semantic-based grouping of the labels. Therefore, for each row \( i \in [M_c] \), we require that
min_{j \in [M_c]} C_p(i, j) = C_p(i, i). In order to quantify the quality of the grouping of labels, we seek a measure that captures the hardness of confusing the coarse predictor induced by \( T \) relative to classifier \( h \), but also accounts for the variability between labels within each coarse class which is captured in the diagonal of \( C_p \). To this end, we introduce the following.

**Definition 4.** The **Coarse Mapping Quality matrix (CMQ)** is the zero diagonal matrix \( \Pi_p(T) \in \mathbb{R}^{M_c \times M_c} \) associated with the grouping induced by the mapping \( T \), whose entries are derived from matrix \( C_p \) as

\[
\Pi_p(i, j; T) = C_p(i, j; T) - C_p(i, i; T), \quad i, j \in [M_c].
\]

This CMQ identifies features that best separate two or more classes. A larger value in \( \Pi_p \) captures the relative hardness of moving coarse label \( i \) to \( j \) under grouping function \( T \), measured by the difference of the mean minimum perturbations for altering classifications between the coarse classes and within the class. Phrased differently, a larger value indicates that it is harder to misclassify coarse label \( i \) as \( j \) relative to inducing misclassifications within set \( i \). Therefore, the entry \( \Pi_p(i, j; T) \) serves as a good measure of the separability of coarse classes \( S_i \) and \( S_j \).

To characterize the overall quality of a mapping \( T \) based on the CMQ, we use

\[
\alpha_p(T) = \sum_{i \in [M_c]} \sum_{j \in [M_c]} \Pi_p(i, j; T),
\]

\[
\beta_p(i; T) = \sum_{j \in [M_c]} \Pi_p(i, j; T).
\]

Favorable mappings – in the sense of inducing well-separated coarse classes – will yield larger values of \( \alpha_p \). Therefore, to obtain the overall best mapping, we formulate the optimization problem (BM) in (3.12), which maximizes the index \( \alpha_p \) among the class of mappings that satisfy the constraints.
\[
\max_T \{ \alpha_p(T) : \Pi_p(i, j; T) > 0, \forall i, j \in [M_c], i \neq j \}. \tag{3.12}
\]

In the experimental evaluation section, we will show that if \( T \) is a favorable, high values of \( \alpha_p \) always represent groupings of finer classes that express a semantic relationship.

The program in (BM) searches for the best placement of \( M \) distinct classes into \( M_c \) non-empty groups. Therefore, the brute force approach to finding the optimal mapping amounts to a search over \( \mathcal{S}(M, M_c) \) (the Stirling number of the second kind [58]) groupings. Thus, the optimal solutions to (BM) is
\[
T_{BM}^* = \arg\max_{T \in L} \{ \alpha_p(T) : \Pi_p(i, j; T) > 0, \forall i, j \in [M_c], i \neq j \}, \tag{3.13}
\]
where \( L \) is the set of all possible mappings with \( |L| = \mathcal{S}(M, M_c) \). We develop the procedure described in Algorithm 3 where we only consider \( Q \ll \mathcal{S}(M, M_c) \) possible groupings depending on the CMQ matrix. First, we initialize the mapping at random (\( T_{\text{ini}} \)). Then, based on the minimum value of the corresponding CMQ, we choose source and destination coarse sets with indices \( i_s \) and \( i_d \) in step 3. From the source set \( S_{i_s} \), we find label \( m_s \) that minimizes \( G_p \) over the source and destination pair and re-locate it to the destination set \( S_{i_d} \) in step 5. These steps are repeated \( Q \) times. Steps 6 and 8 ensure that we do not repeat a previous grouping. Based on the constraints of (BM), the suboptimal mapping is then returned based on \( \alpha_p \).

Improving the Coarse Robustness

Given the trained classifier \( h \) and some grouping function \( T \), in this section, we propose enhancing the proposed global coarse robustness of \( h \) by introducing our coarse training method. We consider additional \( M_c \) heads at the output of \( h \) representing the coarse class belonging for enhancing the proposed global coarse robustness of \( h \). We augment the two sets of outputs into one set similar
Algorithm 3 Finding suboptimal mapping for (BM)

**Input:** $G_p, M_c, Q, T_{ini}$

**Output:** $T^*$

1: Initialize $T = T_{ini}$, $P = \{\emptyset\}$, iteration = 0
2: While iteration $\leq Q$
3: 
4: 
5: re-locate $m_s$ from $S_{is}$ to $S_{id}$. update $T$, iteration $\leftarrow$ iteration + 1
6: if $T \notin P$, update $P \leftarrow P \cup \{T\}$
7: else, restart $T$ uniformly at random
8: obtain $V = \{T \in P : \text{Constrains. of (BM)}\}$
9: return $T^* = \arg\max_{T \in V} \alpha_p(T)$

to NT but with considering $M + M_c$ outputs and a re-labeled dataset according to $T$. Formally, We consider $k : \mathbb{R}^N \rightarrow \Delta^{M+M_c}$ with entries $k_i, i \in [M + M_c]$, such that the fine and coarse predictions are given as $h(x) = \arg\max_{i \in [M]} k_i(x)$, and

$$T(x) = \arg\max_{i \in \{M+1, \ldots, M+M_c\}} k_i(x) - M. \quad (3.14)$$

We use $\theta_a$ to denote the set of the augmented adjustable parameters along with $\theta$. Moreover, we introduce training dataset $D_a$ with entries $(x, p)$ where $x$ is from $\mathcal{D}$, and vector $p \in \Delta^{M+M_c}$ is obtained from $y$ as follows. The true finer and coarser labels are presented in vector $p$ by stacking vectors $0.5 \mathbf{1}_y$ and $0.5 \mathbf{1}_c$, i.e., $p = 0.5[\mathbf{1}_y \quad \mathbf{1}_c]^T$. See Figure 3.1 for an example.

**Definition 5.** Given classifier $h'$, parameterized by $\theta_a$, we define the Augmented Heads Coarse Training (AHCT) on dataset $D_a$ as the task of minimizing the loss function in (3.15) in order to obtain high classification accuracy w.r.t. to the fine and coarse information.

$$\min_{\theta_a} \frac{1}{|D_a|} \sum_{(x, p) \in D_a} \mathcal{L}(k(x; \theta_a), p). \quad (3.15)$$
Next, we present the following proposition to prove the results theoretically using the proposed GCR measure.

**Theorem 2.** Given mapping $T$, NT classifier $h$ with $c_p^{NT}(m; T)$, and AHCT classifier $h'$ with $c_p^{AHCT}(m; T)$, then

$$
\mathbb{E}_{m \in [M]} c_p^{NT}(m; T) \leq \mathbb{E}_{m \in [M]} c_p^{AHCT}(m; T).
$$

(3.16)

**Proof.** Given an observation vector $\mathbf{x}$, we use $\eta^{NT}(\mathbf{x})$ and $\eta^{AHCT}(\mathbf{x})$ to denote the minimum perturbations needed to induce coarse mis-classification for models trained by the NT and AHCT methods, respectively. This means,

$$
\eta^{NT}(\mathbf{x}) = \arg\min_{\eta} \{ \| \eta \|_p : \arg\max_{m \in [M]} f_m(\mathbf{x} + \eta) \notin S_{T(y)} \},
$$

(3.17)
\[ \eta^{\text{AHCT}}(x) = \arg\min_{\eta} \{ \| \eta \|_p : \text{argmax}_{i \in [M]} k_i(x + \eta) \notin S_{T(y)} \} \cap \text{argmax}_{i \in \{M+1,...,M+M_c\}} k_i(x + \eta) \neq T(y) \}. \]

(3.18)

Given the definition of the GCR, if we prove that \( \| \eta^{\text{NT}}(x) \|_p \leq \| \eta^{\text{AHCT}}(x) \|_p \), then it follows that (3.16) is satisfied. The values of \( \eta^{\text{NT}}(x) \) and \( \eta^{\text{AHCT}}(x) \) are obtained using targeted attacks as follows. All target labels in the set outside the true coarser set, \( t \in \bar{S}_{T(y)} \), are tried to get \( \eta^{\text{NT}}_t(x) \) and \( \eta^{\text{AHCT}}_t(x) \), then the minimum is selected. Given a target \( t \), the targeted perturbations are obtained using the standard unrestricted targeted attack formulation for the NT and AHCT models as given in (3.19) and (3.20), respectively.

\[
\begin{align*}
\min_{\eta} & \| \eta \|_p \quad \text{s.t.} \\
& f_t(x + \eta) > f_m(x + \eta), \forall m \in [M] \setminus \{t\}.
\end{align*}
\]

(3.19)

\[
\begin{align*}
\min_{\eta} & \| \eta \|_p \quad \text{s.t.} \\
& k_t(x + \eta) > k_i(x + \eta), \forall i \in [M] \setminus \{t\}, \\
& k_{M+T(t)}(x + \eta) > k_{M+i}(x + \eta), \forall i \in [M_c] \setminus \{M + T(t)\}.
\end{align*}
\]

(3.20)

Given the additional constraints in (3.20), the feasible set of (3.20) is a subset of the feasible region of (3.19). Hence, \( \| \eta^{\text{NT}}_t(x) \|_p \leq \| \eta^{\text{AHCT}}_t(x) \|_p \) which yields to \( \| \eta^{\text{NT}}(x) \|_p \leq \| \eta^{\text{AHCT}}(x) \|_p \), and that concludes the proof. \( \square \)
In this chapter, we consider the CIFAR10 [3] (FMNIST [2]) dataset by which class numbers 0-9 represent airplane (T-shirt), car (trouser), bird (dress), cat (coat), deer (sandal), dog (shirt), frog (sneaker), horse (bag), ship (boot), and truck. We use standard convolutional neural network classifiers. We use the cross-entropy loss for $\mathcal{L}$ in both NT and AHCT, and the state-of-the-art targeted version of the Projected Gradient Descent attack [59] method with $p = 2$ to generate the perturbations. We use Intel(R) Core(TM) i9-9940 CPU @ 3.30GHz machine.

For our first experiment, where the outcomes are given in Figure 3.2, we present results for the best mapping function $T$ obtained as a solution to (BM) using the CMQ matrix $\Pi_2$ for CIFAR-10 (first and second) and FMNIST (third and fourth).

**Numerical Results**

Figure 3.2: Results for the best mapping function $T$ obtained as a solution to (BM) using the CMQ matrix $\Pi_2$ for CIFAR-10 (first and second) and FMNIST (third and fourth).
Table 3.1: Performance of Algorithm 3 in obtaining the best overall mapping. The results are based on the average obtained optimal values of $\alpha_2$, and the average required number of iterations over 5 different initial mappings, $\mathbb{E}(Q)$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$S(10,3)$</th>
<th>$\alpha_2^*$</th>
<th>$\mathbb{E}(\alpha_2)$</th>
<th>$\mathbb{E}(Q)$</th>
<th>$S(10,4)$</th>
<th>$\alpha_2^*$</th>
<th>$\mathbb{E}(\alpha_2)$</th>
<th>$\mathbb{E}(Q)$</th>
<th>$S(10,5)$</th>
<th>$\alpha_2^*$</th>
<th>$\mathbb{E}(\alpha_2)$</th>
<th>$\mathbb{E}(Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>9330</td>
<td>0.59</td>
<td>0.59</td>
<td>2139</td>
<td>34105</td>
<td>1.17</td>
<td>1.17</td>
<td>2017</td>
<td>42525</td>
<td>1.645</td>
<td>1.645</td>
<td>2369.8</td>
</tr>
<tr>
<td>FMNIST</td>
<td>9330</td>
<td>2.58</td>
<td>2.58</td>
<td>1308.4</td>
<td>34105</td>
<td>4.44</td>
<td>4.44</td>
<td>1828.4</td>
<td>42525</td>
<td>7.11</td>
<td>7.11</td>
<td>2477.6</td>
</tr>
</tbody>
</table>

$M_c = 4$. Furthermore, higher values in $\Pi_2$ are associated with groupings that are more semantically consistent. For example, in the first CMQ, the grouping in the first row, which places instances of ‘Ship’ and ‘Cat’ in one set, returns a low $\beta_2 = 0.032$, while the semantic-based grouping of ‘Auto’ and ‘Truck’ returns $\beta_2 = 0.3817$. Interestingly, class separation is not always in one-to-one correspondence with natural semantic-based groupings as for the former example. Therefore, our analysis also sheds some light on a source of vulnerability of classifiers to imperceptible adversarial attacks in that they may not capture semantic similarities of classes. Specifically, the long-observed vulnerability is not very surprising considering the fact that certain labels could be close with regard to the amount of perturbation causing their misclassification while not being close semantically.

In Table 3.1, the performance of Algorithm 3 in obtaining the best overall mapping in comparison with the brute force method is demonstrated. We report the index of the optimal mapping $\alpha_2^*$ (brute force), the index of the mapping obtained by Algorithm 3 averaged over 5 random initial mappings $T_{ini}$, and the required number of iterations for both approaches. As shown, Algorithm 3 succeeds in obtaining the exact optimal mapping in all scenarios. The optimal values are obtained by our algorithm are confirmed by the brute force method for which all the $S(10,M_c)$ possibilities are tried. For all cases, in general, Algorithm 3 requires a considerably smaller number of iterations to converge than brute force search.

For our third experiment in Table 5.1, we present results comparing the global standard and coarse robustness measures using natural training and the proposed augmented heads coarse training. Table 5.1 presents CIFAR10 and FMNIST results using semantic based and random mappings. For each dataset, the mapping that returns the highest CMQ is also considered. The first column is
used to reference the experimental setting of each case. The last two columns represent the average
global standard and coarse robustness measures which will be used as our evaluation of robustness
when we present the following observations.

In all the considered scenarios (trained models and groupings), the GCR is higher than GSR. This
indicates that inducing coarse miss-classification requires, on average, larger amount of pertur-
bations when compared to those needed to induce any miss-classification. Furthermore, when
we compare NT and AHCT models for the amount of average perturbations needed to cause any
miss-classification using the GSR results, we observe that it is not necessary the AHCT scores
higher as seen in run ID 10 vs. run ID 11 for an example. This reflects that inducing any miss-
classification in the AHCT models may became easier as we are not using one-hot encoding for
the finer label as in NT, and use a value of 0.5 to represent the true fine label as presented in the
AHCT method. Lastly, the observed improvements, in the AHCT training, is evidenced to take
place while requiring approximately 100 (400) additional trainable parameters for the FMNIST
(CIFAR-10) dataset, which is a relatively small number when compared to the total number of parameters, 26605 (1211786).

Discussion and Conclusion

In this chapter, we first introduced global measures of the notion of coarse robustness. An efficient algorithm to identify robust groupings relative to the introduced measures was developed. Furthermore, we presented a method to improve the coarse robustness using a modified structure that incorporates the coarse information. Further, we proved the enhancement theoretically and empirically. An intriguing observation of our experiments on benchmark datasets for image classification is that semantically plausible groupings of the class labels are often consistent with large values of the measures introduced.
CHAPTER 4: BIDIRECTIONAL ONE-SHOT SYNTHESIS FOR ADVERSARIAL EXAMPLES

Introduction

The problem of robustness is being assessed in adversarial machine learning via additive perturbations to data and the synthesis of adversarial examples, which are often used to test the robustness of a given model. In this chapter, we reconcile the notion of one-shot learning [60] and the synthesis of adversarial examples for the first time in what we call one-shot synthesis. In particular, given a datum $x_d$ and a pre-trained model $p(.; \theta)$ parameterized by $\theta$, we propose a synthesis procedure that generates a new datum $x$ to be used as an input to $p(.; \theta)$ such that constraints are satisfied on both the input structure and the output inference. In terms of the input, we ensure that $x$ is similar to the given reference datum $x_d$ by enforcing a small distance $d(x, x_d) \leq \delta_s$. In terms of the output, we generate $x$ such that it approximately induces a user-defined output distribution $p_d$ as the inference result $p(.; \theta)$ of the pre-trained model by enforcing a small distance $D(p(x; \theta), p_d) \leq \delta_c$. In this sense, the underlying Bidirectional One-Shot Synthesis (BOSS) problem is concerned with generating data satisfying constraints on both the input and output directions of the given classifier $p(.; \theta)$. By controlling the induced output distribution, our approach generalizes traditional notions of targeted and non-targeted attacks [61]. Confidence reduction attacks can also be implemented in our approach, where the goal is to lower the confidence level of the true label to cause ambiguity [19], specifically against systems for which a confidence threshold is introduced and the classification is only regarded if the prediction confidence score is above that threshold [62].

We propose a solution to the BOSS problem by leveraging generative models whose parameters

---

1The material in this chapter is published in MLSP 2022 [63].
Bidirectional Input/Output Constrains:

Given Trained Classification Model:
\( p(x; \theta) \),

Synthesize such that:
\( x \approx x_d \)

Desired Specifications

- **Yield**: 0.50
- **Speed 70**: 0.50
- **Yield**: 0.45
- **Speed 70**: 0.55

\( p_d \)

\( x \)

Related Work

Our approach is related to generative modeling using GANs in that we utilize a similar structure and layer configuration to its generator side, but there are key differences. Underlying GANs is the training of two sub-models: a generator model trained to generate fake examples resembling ones from an original dataset, and a discriminator model trained to classify examples as either real (i.e., from the domain) or fake (generated) [64]. The process of training a GAN requires a large amount of training data. Here, the underlying task, and the training process and its requirements are altogether different: given a trained model, we seek to generate an example that induces a predefined Probability Mass Function (PMF) at its output and to simultaneously enforce similarity to a desired example, without access to any training dataset. This is accomplished in our implementation without the use of a discriminator. Additionally, the generator and the discriminator of a GAN are
trained together in a zero-sum game until the discriminator is fooled about 50% of the time [64]. In our case, we update the parameters of the generator using loss functions defined with respect to (w.r.t.) given specifications for every feature vector. Variants of GANs have also been developed in prior work to perform individual and universal attacks. Examples include [65], in which a GAN architecture is trained along with an attacker model in a three-way game for enhanced adversarial training and faster convergence, the Conditional GANs (CGANs) in [66], where the generator, discriminator, and the classifier are trained to generate additive perturbations, and the Auxiliary Classifier GAN (AC-GAN) [67], in which an auxiliary classifier is used instead of the discriminator. More closely related to this chapter is the work in [68] ([69]) which uses generative networks without discriminators to synthesize additive (non-additive) adversarial samples, however, there are two major distinctions. First, unlike these works, we do not require a labeled dataset to train our system to synthesize adversarial samples at inference time. Second, the methods in [68] and [69] are only implemented for targeted and non-targeted attacks. Our approach, on the other hand, is unifying in that it offers full control over the desired output PMF of the trained classifier (while maintaining similarity to specific input features), making it applicable to a wide range of scenarios, including targeted/non-targeted attacks, confidence reduction attacks, and synthesis of decision boundary examples.

In fact, our proposed framework is more similar to the additive attack methods where a large body of works are presented such as the CW attack [43], the L-BFGS attack [44], Deepfool [45], Fast Adaptive Boundary (FAB) attack [70], saliency map attack [19], and NewtonFool [71]. These methods, however, largely optimize a cost function expressed in terms of an additive perturbation norm and/or the model’s loss subject to misclassifications of the adversarial examples. In sharp contrast, we use a generative approach in which adversarial samples are synthesized from scratch (rather than generating imperceptible perturbations). In this chapter, for baselines, we will use the CW attacks (NewtonFool) as it is the state-of-the-art method for targeted (confidence reduction)
attacks.

Problem Formulation & Characterization

Suppose we have some trained model $p$ with parameters $\theta$ (e.g., a trained Neural Network) and a probability distribution $p(.; \theta) : \mathbb{R}^N \rightarrow \Delta^M$ over the output of the model with entries $p_m(x; \theta)$ for $m \in [M]$, where $M$ is the total number of outputs, and $\Delta^M$ is the probability simplex over $M$ dimensions.

Given a clean example (desired input features) $x_d$, the well-known formulation of the basic iterative extension of the Fast Gradient Sign Method (FGSM) method [47] generates an adversarial example $x$ by minimizing some differentiable loss function between $p(x; \theta)$ and $p_d$. The distance between $x$ and $x_d$, however, is restricted to the $l_p$ norm. A more general formulation is used in [43] where the loss functions on the input and output of the classifier of interest can be chosen more flexibly. Therefore, we will compare our approach to the attacks in [43] and an advanced version in [49].

Let $d : \mathbb{R}^N \times \mathbb{R}^N \rightarrow [0, 1]$ and $D : \Delta^M \times \Delta^M \rightarrow [0, 1]$ denote distance functions between two feature vectors and distributions, respectively, where a value 0 indicates identical arguments.

Definition

**Definition 6 (BOSS Problem).** Given a learning model $p(.; \theta) : \mathbb{R}^N \rightarrow \Delta^M$ parameterized by $\theta$, a tensor $x_d \in \mathbb{R}^N$, and a probability distribution $p_d \in \Delta^M$, find an input tensor $x \in \mathbb{R}^N$ such that $d(x, x_d) \leq \delta_x$ and $D(p(x; \theta), p_d) \leq \delta_c$, where upper bounds $\delta_x$ and $\delta_c$ and loss functions $d$ and $D$ are given.

**Definition 7 (CLIQUE Problem).** Given an undirected graph $G = (U, E)$ and an integer $k$, find a fully connected sub-graph induced by $U' \subseteq U$ such that $|U'| = k$. 

45
First, we prove that the BOSS problem is **NP-complete**. This establishes that, in general, there is no polynomial-time solution to the BOSS problem unless $P = NP$. In the next section, we develop a generative approach to obtain an approximate solution to the BOSS problem.

**Theorem 3.** The Bidirectional One-Shot Synthesis (BOSS) problem in Definition 6 is **NP-complete**.

**Proof.** It is easy to verify that the BOSS problem is in **NP** since, given a tensor $x$, one can check whether the input and output constraints $d(x, x_d) \leq \delta_s$ and $D(p(x; \theta), p_d) \leq \delta_c$ are satisfied in polynomial time. It remains to be shown whether the BOSS problem is **NP-hard**. We will establish this result via a reduction from the CLIQUE problem in Definition 7. Given a CLIQUE instance $\langle G = (U, E), k \rangle$ with $|U| = n$ and $|E| = m$, we construct its corresponding BOSS instance $\langle p(\cdot; \theta), x_d, p_d, \delta_s, \delta_c \rangle$ as follows. Let $x_d = 0$ denote the all-zeroes vector and let $p_d$ be defined as

$$p_d = \begin{pmatrix} e^{\varepsilon k} & e^{\varepsilon k(k-1)/2} \\ e^{\varepsilon k(k-1)/2} + e^{\varepsilon k} & e^{\varepsilon k(k-1)/2} + e^{\varepsilon k} \end{pmatrix},$$

where $\varepsilon \leq 1 - \sqrt{1 - (1/(k + 1))}$. Finally, let $\delta_s = k/n$ and $\delta_c = 0$. We choose the mean square error loss (MSE) function to compute $d(x, x_d) \leq \delta_s$. The choice of loss function for computing $D(p(x; \theta), p_d) \leq \delta_c$ is superfluous since we have chosen $\delta_c = 0$. For the given trained model $p(\cdot; \theta)$, we define its connectivity and parameters $\theta$ as follows. The input layer consists of $n$ entries given by the solution $x \in [0, 1]^n$. There is one hidden layer consisting of $n + m$ ReLU functions $\sigma_1, \ldots, \sigma_{n+m}$ such that the first $n$ ReLU functions have a bias term of $\varepsilon - 1$ and the next $m$ ReLU functions have a bias term of $\varepsilon - 2$. Finally, there is an output layer with two softmax output activation functions $p_1(x; \theta)$ and $p_2(x; \theta)$. Let $\theta_{ij}^h$ denote the weight of the connection between the $i^{th}$ input $x_i$ and the $j^{th}$ ReLU activation $\sigma_j$ in the hidden layer. For each $u_i \in U$ in the given CLIQUE instance, we have $\theta_{ii}^h = 1$. The outputs of these $n$ ReLU activation functions
are fully connected to the softmax output activation function $p_1(x; \theta)$, each with a corresponding weight of 1. For each edge $e_k = (u_i, u_j) \in E$, we have $\theta^h_{i,n+k} = \theta^h_{j,n+k} = 1$. This defines the input connectivity of ReLU functions $\sigma_{n+1}$ to $\sigma_{n+m}$. The outputs of these are then fully connected to the second softmax function $p_2(x; \theta)$, each with weight 1. See Figure 4.2 for an example. We now prove that there is a clique of size $k$ in $G$ if and only if there is a feasible solution $x$ to the reduced BOSS instance.

( $\implies$ ) Assume there is a clique of size $k$ in $G$. We can derive a feasible solution $x$ to the reduced BOSS instance as follows. For every vertex $u_i \in U$ in the clique, let $x_i = 1$ and let all other values of $x$ be 0. The corresponding MSE loss is $d(x, x_d) = k/n$, thereby satisfying the input constraint defined by $\delta_k$. The solution $x$ induces an output of $\sigma_i(x_i + (\varepsilon - 1)) = \varepsilon$ for each entry of $x$ corresponding to a vertex $u_i$ in the clique and an output of 0 for all other entries. Thus, we have $k$ inputs of value $\varepsilon$ into the first softmax output function. Now, let us consider the edges induced by this clique. For each edge $e_k = (u_i, u_j) \in E$ in the clique, we have $\sigma_{n+k}(x_i + x_j + (\varepsilon - 2)) = \varepsilon$ and an output of 0 for all other edges. Since there are $k(k - 1)/2$ edges in a clique of size $k$, this yields $k(k - 1)/2$ inputs of value $\varepsilon$ into the second softmax output function. Thus, we have $p(x; \theta) = p_d$ and the constraint $D(p(x; \theta), p_d) \leq 0 = \delta_c$ is satisfied. As a caveat, it is worth noting that, under this construction, the all-zeroes vector $0$ yields equal outputs $p_1(0; \theta) = p_2(0; \theta) = 1/2$. Per the preceding arguments, it is also the case that a feasible solution $x$ derived for a clique of size $k = 3$ will output $p_1(x; \theta) = p_2(x; \theta) = 1/2$. This is because $k = k(k - 1)/2$ for $k = 3$. Thus, for the remainder of the proof, we assume that cliques of interest are of size $k > 3$.

( $\Longleftarrow$ ) We prove the contrapositive. That is, if there is no clique of size $k$ in $G$, then the reduced BOSS instance is infeasible. We proceed by showing that there must be exactly $k$ non-zero entries in $x$ in order to satisfy constraints $d(x, x_d) \leq k/n$ and $D(p(x; \theta), p_d) \leq 0$ and that, if there is no clique of size $k$, then there is no choice of $k$ non-zero entries in $x$ that will satisfy $D(p(x; \theta), p_d) \leq 0$. Note that there must be at least $k$ entries in $x$ with value strictly greater than $(1 - \varepsilon)$ in order to yield
Figure 4.2: Example reduction from a graph $G$ (left) to a classifier $p(\cdot; \theta)$ (right).

an input of $\varepsilon k$ into the first softmax output function and satisfy the first entry in $p_d$. Let us consider the minimum MSE loss for a solution $x$ with more than $k$ non-zero entries. For $k+1$ entries of value strictly greater than $(1 - \varepsilon)$, we have $d(x, x_d) = (k + 1)(1 - \varepsilon)^2/n$. With some algebraic manipulation, we have that, for any value of $\varepsilon \leq 1 - \sqrt{1 - (1/(k+1))}$, $(k + 1)(1 - \varepsilon)^2/n > k/n$, thereby violating the constraint $d(x, x_d) \leq \delta_s$. Thus, there must be exactly $k$ non-zero entries in $x$.

Now, let us consider the second softmax output function, which requires an input of $\varepsilon k(k - 1)/2$. Since there is no clique of size $k$ in $G$, any choice of $k$ vertices in $G$ will induce a set of edges whose cardinality is strictly less than $k(k - 1)/2$. Therefore, the output of the second softmax function will be strictly less than the second entry in $p_d$. This violates the constraint $D(p(x; \theta), p_d) \leq 0$.

Note that, for a given CLIQUE instance in the proof of Theorem 3, the corresponding reduced BOSS instance is such that, if there exists a polynomial-time solution to the BOSS problem, then we could use this solution to solve the CLIQUE problem in polynomial time. This would imply that $P = NP$. We therefore conjecture that a polynomial-time solution to the BOSS problem is not likely to exist.
Generative Approach

To obtain a solution to the BOSS problem in Definition 6, we take a generative approach in which $x$ is obtained as the output of a generative network, $g(\cdot; \phi) : \mathbb{R}^Q \rightarrow \mathbb{R}^N$, with parameters $\phi$, i.e., $g(z; \phi) = x$, where $z \in \mathbb{R}^Q$ is a random input to the generative network. We utilize the adjustable parameters of network $g$ for the objectives of BOSS. Therefore, we define the combined network $h(\cdot; \psi) : \mathbb{R}^Q \rightarrow [M]$, whose layers are the concatenation of the layers of $g$ and $p$, where $\psi = \{\phi, \theta\}$. In other words, $h(z; \psi) = p(x; \theta) = p(g(z; \phi); \theta)$. We augment a repeated version of vector $z$ to create a small training dataset. Given the two objectives of BOSS, and the utilization of the adjustable parameters of network $h$, $\phi$, we introduce the surrogate losses $\mathcal{L}_h(p(g(z; \phi); \theta), p_d)$ and $\mathcal{L}_g(g(z; \phi), x_d)$, and use the back-propagation algorithm [24] to optimize $\phi$ based on the minimization

$$\min_{\phi} \left[ \mathcal{L}_g(g(z; \phi), x_d) + \lambda \mathcal{L}_h(p(g(z; \phi); \theta), p_d) \right], \quad (4.2)$$

where $\lambda$ is a loss weight. It is important to note that (4.2) is used to update parameters $\phi$ while the trained classifier parameters $\theta$ remain unchanged. Due to the use of network $h$, the surrogate loss functions $\mathcal{L}_g$ and $\mathcal{L}_h$ can be selected as the MSE and the categorical cross-entropy loss, respectively.

In the following, we present an algorithmic approach to solve BOSS by iteratively optimizing (4.2). At every iteration, the adjustable parameters $\phi$ of the generator model $g$ are updated to satisfy the two objectives of small PMF distance from $p_d$ and high similarity of the generated example to $x_d$. We define an exit criteria if either a maximum number of iterations/steps is reached, or if a feasible solution per Definition 6 is found given $x_d$, $p_d$, $\delta_s$, and $\delta_c$. The parameter $\lambda$ in (4.2) weighs the relative importance of each loss function to both avoid over-fitting and handle situations in which the solver converges for one loss function prior to the other [72]. We propose a dynamic update that depends on the distance $D$ between the desired specification $p_d$ and the actual output $p(g(z; \phi); \theta)$.
Algorithm 4 BOSS Algorithm

**Input:** \( z, p(.; \theta), g, x_d, p_d, \delta_c, \delta_s \)

**Output:** \( x \)

1: Initialize \( x, \phi, \lambda \)
2: while \( D(p(.; \theta), p_d) \geq \delta_c \) or \( d(x, x_d) \geq \delta_s \)
3: obtain \( \phi \) as the minimizer of (4.2) with \( \lambda \)
4: \( x = g(z; \phi) \)
5: update \( \lambda \) using (4.6)
6: return \( x \)

at every iteration. Specifically, we update \( \lambda \) as

\[
\lambda \leftarrow \sigma \left( \lambda - \lambda^0 \frac{\delta_c}{D} \left( \frac{\delta_c}{D} - 1 \right) \right). \tag{4.3}
\]

As such, it is required to have the distance function \( D \) returning values in the range of \([0, 1]\). Here, we utilize the Jensen-Shannon (JS) divergence distance [73], which returns 0 for two equivalent PMFs and is upper bounded by 1. The updates are also a function of the initial selection of \( \lambda \) denoted \( \lambda^0 \). In this chapter we focus on the task of image classification. Specifically, we set \( d = 1 - I \), where \( I \) is the Structural Similarity Index (SSIM) [54], which is equal to 1 for two identical images and captures luminance, contrast, and structure in the measurements.

The signum function \( . \) is used to determine whether to increase or decrease \( \lambda \), based on the ratio of the actual and desired specifications which regulates the amount of change. The ReLU function \( \sigma(.) \) prevents \( \lambda \) from becoming negative. This occurs when the desired \( p_d \) is easily attained in early steps of the algorithm. The procedure is presented in Algorithm 4.
In this subsection, we extend BOSS to generate targeted adversarial examples against the two-stage FHC and TDHC models. Given the formulation of these models in Chapter 2, the parameters of the generator networks are updated using (4.4) for the FHC and (4.5) for the TDHC.

\[
\min_{\phi} \left[ \mathcal{L}_g \left( g(z; \phi), x_d \right) + \lambda \mathcal{L}_h \left( J(g(z; \phi)), p_d \right) \right].
\] (4.4)

\[
\min_{\phi} \left[ \mathcal{L}_g \left( g(z; \phi), x_d \right) + \lambda_r \mathcal{L}_{h,r} \left( (h(g(z; \phi)), p_{r,d}^d) \right) + \lambda_f \mathcal{L}_{h,f} \left( (G(g(z; \phi)), p_d) \right) \right].
\] (4.5)

We note that, here, we use \( G \) instead of \( g \) to define the finer classifier in the TDHC. We choose \( \mathcal{L}_g \) as the mean square error, and \( \mathcal{L}_h, \mathcal{L}_{h,r}, \) and \( \mathcal{L}_{h,f} \) as the categorical cross-entropy loss. The parameters \( \lambda, \lambda_r, \) and \( \lambda_f \) are used to weigh the relative importance of each loss function [63]. We dynamically update these parameters using the ratio of the desired and induced distances as

\[
\lambda \leftarrow \sigma \left( \lambda - \lambda^0 \frac{\delta_c}{D} - 1 \right),
\] (4.6)

\[
\lambda_r \leftarrow \sigma \left( \lambda_r - \lambda^0_r \frac{\delta_{r,c}}{D_r} \text{sign} \left( \frac{\delta_{r,c}}{D_r} - 1 \right) \right),
\] (4.7)

\[
\lambda_f \leftarrow \sigma \left( \lambda_f - \lambda^0_f \frac{\delta_{f,c}}{D_f} \text{sign} \left( \frac{\delta_{f,c}}{D_f} - 1 \right) \right).
\]

where \( D_r = D(h(x), p_{r,d}^d) \) and \( D_f = D(G(x), p_{f,d}^d) \), and superscript 0 indicates the initial selection. We use the Jensen-Shannon (JS) divergence as our distance function \( D \), which returns values in \([0, 1]\) [73].

We seek to synthesize \( x \) such that \( k(x) \in S_t \), where \( S_t \subset [M_c] \) is a defined target super-class set. Equivalently, it is required that \( \exists j \in S_t : J_l(x) > J_k(x), \forall k \in [M_c] \setminus S_t \). Since \( J(x_d) \) captures the class membership of the benign example, we define the desired PMF as the indicator function

\[
51
\]
Algorithm 5 BOSS Algorithm for FHC (TDHC) model

Input: \( z, g, x_d, k \) (\( r \) and \( f \)), \( p_d \) (\( p_d^r \) and \( p_d^f \)), \( \delta_s, \delta_c, (\delta_c' \) and \( \delta_c'^f \))

Output: \( x \)

1: Initialize \( x, \phi, \lambda (\lambda_r \) and \( \lambda_f) \)

2: while \( d(x, x_d) \geq \delta_s \) or \( D(J(x), p_d^r) \geq \delta_c \) or \( D(h(x), p_d^f) \geq \delta_c'^f \)

3: obtain \( \phi \) as the minimizer of (4.4) \((4.5)\) with \( \lambda (\lambda_r \) and \( \lambda_f) \)

4: \( x = g(z, \phi) \)

5: update \( \lambda (\lambda_r \) and \( \lambda_f) \) using (4.6) \((4.7)\)

6: return \( x \)

\( p_d(l) = 1\{l = j^*\}, \) which takes the value 1 if \( l = j^* \) and 0 otherwise, where we obtain \( j^* \) as the maximizing class label in the target set \( S_T \), i.e., \( j^* = \arg \max_{j \in S_T} J_j(x_d) \). For the TDHC model, given a target coarse label \( t_c \) and a target fine label \( t_f \), it is required to generate \( x \) such that \( r(x) = t_c \) and \( f(x, r(x)) = t_f \). Hence, the desired PMFs are chosen as \( p_d^r(l) = 1\{l = t_c\}, \forall l \in [M_c] \) and \( p_d^f(l) = 1\{l = t_f\}, \forall l \in [M_c] \). The procedure of using BOSS against the FHC and TDHC models are illustrated in Algorithm 5.

We remark that the BOSS approach for the TDHC can be extended to include more than two classifiers, hence it is applicable to nested dichotomies [51] and error-correcting output codes [40], which use a series of binary classifier structures to perform multi-class classification.

Experimental Results

We show results for Targeted attacks which we call BOSS-T. The desired distribution is selected such that \( p_d(l) = 1 \) if \( l \) is the target entry and 0 otherwise. Second, Confidence reduction examples, which we dub BOSS-C. Let the true label of \( x_d \) be \( f^* \) and the desired confidence be \( c_d \), then \( p_d(l) = c_d \) if \( l = f^* \), and \( (1 - c_d)/(M - 1) \) otherwise. In addition to the samples from BOSS-T and BOSS-C, we also show instances of boundary adversarial examples. In this case, a value of 0.5 is assigned to the two class labels on both sides of the boundary.
Table 4.1: BOSS-C and NewtonFool with $c_d = 0.6, \delta_c = 0.2$, and $\delta_s = 0.85$ for the MNIST dataset.

<table>
<thead>
<tr>
<th>Environment</th>
<th>CA(%)</th>
<th>$\sigma_{con}$(%)</th>
<th>$\sigma_{c}$(%)</th>
<th>$\sigma_{JS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>98.1</td>
<td>99.1</td>
<td>100</td>
<td>0.03</td>
</tr>
<tr>
<td>Model+BOSS-C</td>
<td>98.1</td>
<td>66.73</td>
<td>87.89</td>
<td>0.19</td>
</tr>
<tr>
<td>Model+NewtonFool</td>
<td>75.5</td>
<td>67.85</td>
<td>97.02</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 4.2: BOSS-T attack overall comparison with state-of-the-art attack methods using the CIFAR-10 dataset. For targets, all labels other than the predicted ones are considered.

<table>
<thead>
<tr>
<th>Attack</th>
<th>$\alpha$(%)</th>
<th>$\sigma_2$</th>
<th>$\sigma_\infty$</th>
<th>$\sigma_s$</th>
<th>Average Adversarial Confidence</th>
<th>Average Run Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW-$l_2$ ($\kappa = 0$) [43]</td>
<td>99.55</td>
<td>0.4195</td>
<td>0.0519</td>
<td>0.9966</td>
<td>0.4371</td>
<td>97.0982</td>
</tr>
<tr>
<td>CW-$l_2$ ($\kappa = 10$) [43]</td>
<td>95.2381</td>
<td>0.7755</td>
<td>0.0907</td>
<td>0.9889</td>
<td>0.9964</td>
<td>95.5681</td>
</tr>
<tr>
<td>CW-$l_\infty$ ($\kappa = 0$) [43]</td>
<td>99.55</td>
<td>1.0761</td>
<td>0.1188</td>
<td>0.9780</td>
<td>0.7126</td>
<td>0.1744</td>
</tr>
<tr>
<td>CW-$l_\infty$ ($\kappa = 10$) [43]</td>
<td>96.82</td>
<td>1.91</td>
<td>0.1676</td>
<td>0.9999</td>
<td>0.9966</td>
<td>0.8912</td>
</tr>
<tr>
<td>EAD (EN decision) [49]</td>
<td>100</td>
<td>0.4419</td>
<td>0.0885</td>
<td>0.9961</td>
<td>0.3951</td>
<td>141.971</td>
</tr>
<tr>
<td>EAD ($l_1$ decision) [49]</td>
<td>100</td>
<td>0.5618</td>
<td>0.173</td>
<td>0.9938</td>
<td>0.3754</td>
<td>142.394</td>
</tr>
<tr>
<td>BOSS-T (MSE)</td>
<td>100</td>
<td>1.1589</td>
<td>0.1046</td>
<td>0.9818</td>
<td>0.9879</td>
<td>16.5554</td>
</tr>
<tr>
<td>BOSS-T (Huber)</td>
<td>100</td>
<td>1.1454</td>
<td>0.1075</td>
<td>0.9799</td>
<td>0.9806</td>
<td>21.6079</td>
</tr>
<tr>
<td>BOSS-T (log cosh)</td>
<td>100</td>
<td>1.1055</td>
<td>0.1042</td>
<td>0.9828</td>
<td>0.9785</td>
<td>21.1732</td>
</tr>
</tbody>
</table>

We use $D$ as the JS distance to compare PMFs (desired and actual) and the SSIM index $I$ as a measure of similarity between examples. We define $\sigma_{JS}$ and $\sigma_s$ as the average of $D$ and $I$, respectively, over the set of observations $\mathcal{X}$. In addition to the aforementioned metrics, for BOSS-T, we utilize the attack success rate $\alpha = N_s/|\mathcal{X}|$, where $N_s$ is the number of times a generated adversarial sample is classified as the predefined target label. Further, we use $\sigma_2$ and $\sigma_\infty$ to denote the average $l_2$ and $l_\infty$, respectively. For BOSS-C, we compute $\sigma_{con}$, defined as the average confidence level of prediction of the true label over the set of interest $\mathcal{X}$.

The random vector $z$ of dimension $Q = 100$ is generated from a uniform distribution over the interval $[0, 1]$, and 80 repeated samples are used for training. The initial loss weights are chosen as $\lambda^0 = \lambda_c^0 = 0.001$. The parameters are updated using the ADAM optimizer [74] with initial step size 0.025. The details of the pre-trained classifiers and the generative networks are available in [63].

For BOSS-C, Table 4.1 presents the results for $\sigma_{con}$, $\sigma_s$, and $\sigma_{JS}$. For NewtonFool, we use 50 iterations and set the small perturbations parameter as $\eta = 0.01$. For an average confidence of

53
Figure 4.3: Samples from each class of the CIFAR-10 dataset (columns). The first row shows the original examples. Rows 2-4 represent the synthesized images for BOSS-C, BOSS-B, and BOSS-T, respectively. The rounded percentage values of the confidence level, \( c \), of the BOSS-C samples are placed at the bottom of each image along with the predicted label. For BOSS-B, the first pair at the bottom of every image represents the highest two predicted labels along with their rounded classification scores (second pair). Predicted labels are placed at the bottom of each BOSS-T example. The percentage of the rounded similarity measure \( I \) is placed on top of each generated example.

Figure 4.4: Clean samples from some classes of the ImageNet dataset on VGG16. The first row shows the original examples. Rows 2 and 3 show the synthesized images for BOSS-T and BOSS-B, respectively. The labels are given at the bottom of each example along with the classification confidence. The SSIM and JS measures are reported on the right of each BOSS image.

\[ \sigma_{\text{con}} \approx 67\% \text{, BOSS-C (with } c_d = 0.6 \text{) and NewtonFool are successful in reducing the average confidence of the model from the original value } \sigma_{\text{con}} = 99.1\%. \text{ This is accomplished with very high level of similarity measure of } \sigma_s \approx 88\% \text{ and } \sigma_s \approx 97\% \text{ for BOSS-C and NewtonFool, respectively.}

While NewtonFool attack produces examples with higher \( \sigma_s \), it fails to maintain the classification
accuracy CA which drops from 98.1% to 75.5%, and yields a large distance $\sigma_{JS}$ from the desired PMF.

The results for BOSS-T are presented in Table 4.2 and compared to the state-of-the-art CW [43] and elastic nets attacks (EAD) [75]. We choose these baselines since their formulations admit any differentiable loss function, unlike the well-known Projected Gradient Descent method [76] where the distance between $x$ and $x_d$ is limited to the $l_p$ norm. For each testing example, all labels other than the predicted one are used as targets. Results of the average adversarial confidence and average run time are reported for each case in the last two columns. The parameters for CW and EAD on CIFAR10 are selected from the reported parameters in their respective papers. It is important to note that both of these methods apply their attacks based on the pre-softmax output (sometimes called logits), and hence, cannot specify an exact $p_d$. However, in the CW formulation, the parameter $\kappa$ was introduced to represent the desired logit value to achieve higher adversarial confidence. While setting $\kappa = 0$ in the CW attack returns the best result in terms of imperceptibility, it does not yield the best adversarial confidence. Therefore, we report results for $\kappa = 10$, which yields a better tradeoff between both measures. Furthermore, we implement BOSS-T with different surrogate loss functions in $x$ and $x_d$.

While some variant of EAD and CW achieve a relatively lower imperceptibility (as seen from $\sigma_2$ and $\sigma_\infty$), in terms of adversarial confidence, all variants of BOSS-T return the best results. CW, with $\kappa = 10$, reports similar adversarial confidence, but the attack success ratio does not achieve 100%, and requires 5 times the run time for $l_2$ and nearly 50% increase in imperceptibility (presented in $\sigma_2$ and $\sigma_\infty$) are observed.

Figures 4.1, 4.3, and 4.4 show BOSS-C, BOSS-T, and BOSS-B samples. As observed, BOSS is successful in generating adversarial examples given the desired input specification, represented the original images in the first row, and output specification as represented in the corresponding
application.

Discussion and Conclusion

We introduced BOSS, a framework for one-shot synthesis of adversarial samples that satisfy input and output specifications for pre-trained classifiers. We formulated the BOSS problem and proved that the problem is **NP-Complete**. We developed an approximate solution using generative networks and surrogate loss functions. The flexibility of BOSS is demonstrated through various applications, including synthesis of boundary examples, targeted attacks, and reduction of confidence samples. A set of experiments verify that BOSS, in general, performs on par with state-of-the-art methods and generates the highest adversarial confidence examples.
CHAPTER 5: AN APPROACH TO THE MAXIMUM INDEPENDENT SET PROBLEM USING NEURAL NETWORK STRUCTURES

Introduction

In his seminal work [28], Richard Karp demonstrated the reducibility among combinatorial problems that are complete for the complexity class \( \textbf{NP} \). Combinatorial optimization problems have since been frequently associated with the \textbf{NP-hard} complexity class for which no efficient solutions are likely to exist. Despite their apparent intractability, \textbf{NP-hard} problems have found ubiquitous application across many sectors [77]. While there is not a known polynomial-time solver with respect to (w.r.t.) the size of the input for any of these problems, there are many approximate and efficient solvers [78]. In general, these solvers are broadly categorized into heuristic algorithms [79], conventional branch-and-bound methods [80], and approximation algorithms [81]. One of the most well-studied \textbf{NP-hard} problems is the Maximum Independent Set (MIS) problem which consists of finding a set of maximum cardinality that contains vertices in a graph \( G = (V, E) \) such that no two vertices are connected by an edge [82]. There exist powerful heuristic solvers for the MIS problem. One of the most well-studied \textbf{NP-hard} problems is the Maximum Independent Set (MIS) problem which consists of finding a set of maximum cardinality that contains vertices in a graph \( G = (V, E) \) such that no two vertices are connected by an edge [82]. There exist powerful heuristic solvers for the MIS problem. The state-of-the-art solver is the ReduMIS evolutionary algorithm, which was developed in the pioneering work of [78]. Despite the notable success of ReduMIS and related heuristic solvers, developing alternative learning-based solutions to combinatorial optimization problems [83, 84] is premised on unceasing advancement in our understanding of the theoretical and computational foundations of machine learning. However, these methods typically require extensive training of neural networks (NNs) using large graph datasets for which solutions
are known. In this chapter, we develop an altogether different NN-based approach to the MIS problem that rests on a dataless NN (dNN) and a novel dataless training paradigm. In our approach, for each graph \( G \), we construct a NN with fixed and trainable parameters that is specific to \( G \) and require no training dataset, hence the appellation ‘dataless’.¹

Related Work

Exact algorithms for **NP-hard** problems are typically based on enumeration or branch-and-bound techniques. However, these techniques are not applicable to large problem spaces [85]. This motivated the development of efficient approximation algorithms and heuristics, such as the procedure implemented in the NetworkX library for solving MIS [81]. These polynomial-time algorithms and heuristics typically utilize a combination of various sub-procedures, including greedy algorithms, local search sub-routines, and genetic procedures [86]. Such heuristics cannot generally guarantee that the resulting solution is within some small factor of optimality. In fact, for the complementary problem of finding a Maximum Clique (MC), an algorithm that provably guarantees an approximate solution to MC within a factor of \( n^{1-\varepsilon} \), where \( n \) is the number of nodes in the underlying graph, for any \( \varepsilon > 0 \) is not possible unless \( P = NP \) [87]. Similar inapproximability results have been established for the MIS problem [88]. As such, heuristics without approximation guarantees have been adopted for practical purposes for these problems.

The ReduMIS method [78] is the state-of-the-art solver for the MIS problem. It consists of two main components: an iterative implementation of a series of graph reduction techniques, followed by the use of an evolutionary algorithm. The latter starts with a pool of independent sets, then evolves the pool over several rounds. In each round, the algorithm uses a selection procedure to select favorable nodes. This is achieved by executing graph partitioning that clusters the graph nodes

¹The material in this chapter is in part published in the 2022 Neural Networks Elsevier journal volume 155.
into disjoint clusters and separators for solution improvement. Our method, however, does not include solution combination operation. In contrast, we leverage a community detection method prior to obtaining the initial solution(s) for the purpose of scaling up to large graphs and not for solution improvement. Moreover, we do not enforce the partitions and separators to be disjoint (i.e., sharing no edges) (See Section 5).

Learning-based approaches which make use of RL algorithms and ML architectures have been recently introduced to solve NP-hard problems. RL-based methods train a deep Q-Network (DQN) such that the obtained policy operates as a meta-algorithm that incrementally yields a solution [77]. The recent state-of-the-art work of [89] combines a DQN with graph embeddings, allowing discrimination between vertices based on their impact on the solution, and enabling scalability to larger problem instances. By contrast, our proposed method does not require training of a DQN. The authors of [90] developed a method to the MIS problem, which, similar to our approach, does not require training data. However, different from our approach, it uses a graph neural network (GNN) (its output represents the probability of each node being in the solution) whose parameters are tuned based on a loss function inspired by quadratic binary unconstrained optimization that encodes the graph of interest. In contrast, we construct a dNN with a simple architecture that only has \( n \) trainable parameters, where \( n \) is the number of nodes in the graph. This is significantly smaller than the number of tunable parameters of the aforementioned GNN, which has \( n \) parameters just in its last layer. In addition, the approach in [90] was only tested using \( d \)-regular graphs [91] and compared against the approximate solver in [81]. The applicability of the former solver is limited to small size graphs and its performance was surpassed by other existing methods.

The supervised learning method in [84] nearly achieves the state-of-the-art performance for the MIS problem. It integrates several graph reductions [78], Graph Convolutional Networks (GCN) [92], guided tree search, and a solution improvement local search algorithm [93]. The GCN is trained using benchmark graphs and their solutions as the true labels to learn probability maps for
each vertex being in the optimal solution. The point of resemblance to our approach is the use of
an NN to derive solutions to combinatorial optimization problems. However, a major difference
is that our approach does not rely on supervised learning; it uses an entirely different dNN and
obtains a solution via dataless training. More specifically, the means by which we optimize the
dNN consists of applying backpropagation [24] to a loss function defined entirely in terms of the
given graph and the structure of the dNN without the need for a dataset as is standard in training
deep learning models.

Methodology

In this section, we describe the different components of our proposed approach. To preface the
discussion and distinguish our dataless solution from learning-based methods, consider first the
conventional supervised learning setting. In this setting, there is generally some set of data $D = \{(x_i, y_i)\}_i$ consisting of input tensors $x_i \in \mathbb{R}^n$ and their associated value, or label, $y_i \in Y$. The goal
of learning is then to train a learning model $f : \mathbb{R}^n \rightarrow Y$ parameterized by $\theta$ so that $f$ learns to
predict the input-output relationship of the underlying data distribution $D'$. This is given by the
objective below, in which $\mathcal{L}$ denotes the loss function.

$$\min_{\theta} \mathbb{E}_{(x,y) \sim D'} [\mathcal{L}(f(x; \theta), y)] \quad (5.1)$$

Since a dataset $D$ is used in lieu of the true underlying data distribution $D'$, the objective function
becomes

$$\min_{\theta} \frac{1}{|D|} \sum_{(x,y) \in D} \mathcal{L}(f(x; \theta), y) \quad (5.2)$$

The loss function is chosen to be a differentiable function, such as the minimum square error, in
order to leverage optimization using backpropagation. In our approach, we leverage dNNs, which
we define as neural networks whose loss function $\mathcal{L}$ does not depend on data. In this sense, what we present is a NN-based technique, but not a learning method and is therefore different from supervised, unsupervised, and reinforcement learning.

**Dataless Neural Network Construction**

Given a graph $G = (V, E)$, we construct a dNN $f$ with trainable parameters $\theta \in [0, 1]^n$ whose single output is $f(e_n; \theta) = f(\theta) \in \mathbb{R}$. Note that the input to $f$ is the all-ones vector $e_n$ and thus does not depend on any data. The network consists of an input layer, two hidden layers, and an
output layer. The trainable parameters \( \theta \in [0,1]^n \) connect the input layer \( e_n \) to the first hidden layer through an elementwise product. All other parameters are fixed during training and are presented next. The connectivity structure from the first hidden layer to the second is given by the binary matrix \( W \in \{0,1\}^{n \times (n+m)} \) and will depend on \( G \), the bias vector at the second hidden layer is given by \( b \in \{-1,-1/2\}^{n+m} \), and the fully-connected weight matrix from the second hidden layer to the output layer is given by \( w \in \{-1,1\}^{n+m} \). These parameters are defined as a function of \( G \).
The output of \( f \) is given by (5.3), where \( \odot \) is the element-wise Hadamard product that represents the operation of the first hidden layer of the constructed network. The second hidden layer is a fully-connected layer with fixed matrix \( W \) and bias vector \( b \) with a ReLU activation function \( \sigma(x) = \max(0,x) \), while the last layer is another fully-connected layer expressed in vector \( w \). See Figure 5.1 for a block diagram of the generalized proposed network.

\[
f(e_n; \theta) = f(\theta) = w^T \sigma(W^T(e_n \odot \theta) + b)
\]  

(5.3)

In the sequel, we prove that \( f(\theta) \) is an equivalent differentiable representation of the MIS problem \( G = (V,E) \) in that \( f \) achieves its minimum value when an MIS \( U \subseteq V \) is found in \( G \). Furthermore, this is a constructive representation since \( U \) can be obtained from \( \theta \) as follows. Let \( \theta^* = \arg\min_{\theta \in [0,1]^n} f(\theta) \) denote an optimal solution to \( f \) and let \( \mathcal{I} : [0,1]^n \to 2^V \) denote the corresponding independent set found by \( \theta \) such that

\[
\mathcal{I}(\theta) = \{ v \in V \mid \theta_v^* \geq \alpha \},
\]

(5.4)

for any \( \alpha > 0 \). We show that \( |\mathcal{I}(\theta^*)| = |U| \). Intuitively, this is tantamount to choosing the trained parameter indices in \( \theta \) whose values exceed some threshold and choosing the vertices in \( V \) corresponding to those indices to be the MIS. The fixed parameters of \( f \) are constructed from the given graph \( G = (V,E) \) as follows. The first \( n \times n \) submatrix of \( W \) represents the nodes \( V \) in
the graph and its weights are set equal to the identity matrix $I_n$. The following $m$ columns of $W$ correspond to the edges $E$ in the graph. In particular, for the column associated with a given edge, a value of 1 is assigned to the entries corresponding to both ends of that edge and 0 otherwise. For the bias vector $b$, we assign a value of $-1/2$ to the entries corresponding to the first $n$ nodes, and a value of $-1$ to the $m$ entries corresponding to the edges. Finally, these nodes are input to their corresponding ReLU activation functions. For the vector $w$ connecting the second hidden layer to the output layer, values of $-1$ and $n$ are assigned to entries corresponding to nodes and edges in the second hidden layer, respectively. Hence, the parameters $W$, $b$, and $w$ are defined as

\[
W(i, i) = 1, \ v_i \in V, \ i \in [n], \\
W(i, n + l) = W(j, n + l) = 1, \ \forall e_l = (v_i, v_j) \in E, l \in [m],
\]

\[
b(i) = -1/2, \ w(i) = -1, \ v_i \in V, \ i \in [n], \\
b(n + l) = -1, \ w(n + l) = n, \ l \in [m].
\]  

Therefore, we can rewrite (5.3) as follows

\[
f(\theta) = -\sum_{v \in V} \sigma(\theta_v - 1/2) + n \sum_{(u, v) \in E} \sigma(\theta_u + \theta_v - 1). 
\]  

(5.7)

Figure 5.2 presents an example. The following theorem establishes a relation between the minimum value of (5.7) and the size of the MIS.

**Theorem 4.** Given a graph $G = (V, E)$ and its corresponding dNN $f$, an MIS $U \subseteq V$ of $G$ is of size $|U| = k$ if and only if the minimum value of $f$ is $-k/2$.

**Proof.** ($\Rightarrow$) Assume that $|U| = k$ and let $\theta_{v_i} = 1$ for each $v_i \in U$ and $\theta_{v_i} = 0$ otherwise. For an arbitrary pair of nodes $v_i, v_j \in U$, consider the output of $f$ as visualized by Figure 5.3, where edge values denote the outputs of the preceding nodes in the network and nodes $\eta_i^1, \eta_i^2$ denote the $i^{th}$
neurons in the first and second hidden layers, respectively. We abuse notation to refer to both the output neuron and the output value as \( f(\theta) \).

Note that \( v_i \) and \( v_j \) each contribute an output of \(-1/2\) to \( f(\theta) \). This follows from the fact that, by definition of MIS, these vertices do not share an edge and so the output of \( \eta_{n+1}^2 \) is 0. Thus, for an MIS of size \(|U| = k\), we have \( f(\theta) = -k/2 \). This is the minimum value attainable by \( f \). Indeed, consider, for the sake of contradiction, that there exists \( \theta' \) such that \( f(\theta') < f(\theta) \). As with \( \theta \), this \( \theta' \) must be defined such that \( \theta_{v_i} = 1 \) for each \( v_i \in U \). Consider the addition of some other \( v_{k'} \notin U \). Then \( \eta_{k'}^2 \) will contribute \(-\sigma(\theta_{k'} - 1/2)\) to \( f(\theta') \) and \( \eta_{n+1}^2 \) will contribute at least \( n\theta_{k'} \) for every edge \( e_k = (v_{k'}, v) \in E \), where \( v \in U \). By definition of MIS, some such edge must exist. Therefore, we would have \( f(\theta') > f(\theta) \), yielding a contradiction.

( \( \Leftarrow \Rightarrow \)) Assume that the minimum value of \( f \) is \( f(\theta) = -k/2 \). Consider an arbitrary edge \( e_l = (v_i, v_j) \in E \). It follows from the construction of \( f \) that \( \theta_{v_i} + \theta_{v_j} \leq 1 \) must hold for \( f \) to achieve its minimum value. For the sake of contradiction, consider that \( \theta_{i} + \theta_{j} > 1 \). In this case, neuron \( \eta_{n+1}^2 \) contributes \( n(\theta_{i} + \theta_{j} - 1) > 0 \) to the output \( f(\theta) \). This yields a contradiction since we can simply choose \( \theta_{i} \) and \( \theta_{j} \) to be 0, thereby contributing a value of 0 to \( f(\theta) \). Given an arbitrary vertex \( v \) and its neighbors \( N(v) \), it must be the case that \( \theta_{i} = 1 \) for some \( v_i \in N(v) \) as this would contribute
a value of $-1/2$ to $f(\theta)$ through node $\eta_i^2$ and a value of 0 through nodes $\eta_{n+1}^2$, for all vertices $u \in N(v)$ connected to $v$ through edge $e_l$. It follows that there must be $k$ entries in $\theta$ with value 1, each contributing a value of $-1/2$ to the output $f(\theta)$ such that none of them share a neuron in the second hidden layer. These correspond to $k$ vertices in $V$ that do not share edges. Therefore, $|U| = k$. □

From Theorem 4, it follows that the minimum value of $f$ is achieved when the maximum number of entries in $\theta$ have value 1 such that their corresponding nodes in $G$ share no edges. This yields an independent set $\mathcal{I}(\theta) = \{v_i \in V \mid \theta_{v_i} = 1\}$ of maximum cardinality.

Due to the non-linearity introduced by the ReLUs in the dNN $f$, we obtain a minimizer for $f$ by leveraging the backpropagation algorithm along with the well-known ADAM optimizer [94]. This is applied only to optimize the adjustable parameters $\theta$. Hence, the ADAM optimizer updates $\theta$ depending on the computation of only one gradient, that is, $\nabla_\theta |f(\theta) - f_d|^2$. Given that $W$, $b$, and $w$ represent the graph structure, they do not need to be updated, thus computing gradients w.r.t. to these parameters is not required.

As such, we iteratively minimize the loss function $\mathcal{L}(f(\theta), f_d) = |f(\theta) - f_d|^2$, where $|\cdot|$ denotes the absolute value and $f_d$ is the minimum desired value used for parameter tuning. Per Theorem 4, the minimum achievable value of $f(\theta)$ is a function of the size of the MIS. Therefore, during training we select $f_d = -n/2$, a value that is only attained by $f(\theta)$ if $G$ is a null graph.

**On the Duality of MIS and MC**

Since the graph induced by the MIS is a null graph on $G$ and fully-connected on its complement $G'$, we propose to include the edges of $G'$ in the construction of $f$ to enhance the tuning of the parameters $\theta$. We term the resulting enhanced dataless neural network as $h$ with output value
$h(\theta)$. In this case, we extend the definition of the fixed parameters $W \in \{0, 1\}^{n \times (n+m+m')}$, $b \in \{-1, -1/2\}^{n+m+m'}$, and $w \in \{-1, n\}^{n+m+m'}$, by defining the mapping for the augmented portion of these parameters representing the $m'$ edges of $G'$ as

$$W(i,n+m+l) = W(j,n+m+l) = 1, \forall e_l = (v_i, v_j) \in E(G'), l \in [m'], \quad (5.8)$$

$$b(n+m+l) = w(n+m+l) = -1, l \in [m']. \quad (5.9)$$

The construction of the graph-specific dNN $h$ requires a total of $n^2 + n(m+m'+3) + 2m + 2m'$ parameters of which $n^2 + n(m+m'+2) + 2m + 2m'$ parameters are fixed and $n$ parameters are adjustable.

Given (5.7), (5.8), and (5.9), the output of $h$ is

$$h(\theta) = f(\theta) - \sum_{(u,v) \in E(G')} \sigma(\theta_u + \theta_v - 1). \quad (5.10)$$

Figure 5.4 presents an example of the proposed construction from a simple 5-node graph $G$ to its corresponding dNN $h$. Our next result is analogous to Theorem 4 and establishes a relation between the minimum value of (5.10) and the size of an MIS in $G$.

**Theorem 5.** Given a graph $G = (V,E)$ and its corresponding enhanced dNN $h$, an MIS $U \subseteq V$ of $G$ is of size $|U| = k$ if and only if the minimum value of $h$ is $-k^2/2$.

**Proof.** Per Theorem 4, the minimum value of the first term of (5.10) is $-k/2$. Therefore, we consider the minimum value of the remaining (second) term, which corresponds to the edges of $G'$. Similar to Theorem 4, assume that $\theta_v = 1$ for each $v \in U$ and $\theta_v = 0$ otherwise. The graph induced by the MIS w.r.t. $G'$ is a fully-connected graph, i.e., $|E(G'[U])| = k(k-1)/2$. Given the $-1$ bias, the outputs associated with the edges of $G'$ will be 1. Since the subgraph induced on $G'$ is complete,
Figure 5.4: An example of graph $G = (V = \{v_1, v_2, v_3, v_4, v_5\}, E = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5)\})$ and its dNN construction $h$ for the MIS problem by leveraging the duality between MIS and MC.

we get $-k(k-1)/2$ for the second term. The combined output is thus $-(k/2) - (k(k-1)/2) = -k^2/2$, which concludes the proof.

Since the minimum value of $h$ is $-k^2/2$, we use $h_d = -n^2/2$ for training the dNN by minimizing the loss

$$\mathcal{L}(h(\theta), h_d) = |h(\theta) - h_d|^2.$$  \hspace{2cm} (5.11)

In the case of (5.11), the backpropagation is used to optimize the adjustable parameters $\theta$ only. This is applied using the ADAM optimizer that updates $\theta$ depending on the computation of only $\nabla_\theta |h(\theta) - h_d|^2$.

In general, the vertices with high degrees are less likely to be part of an MIS than vertices with low degrees. Therefore, to speed up the training of the parameters of $h$, we initialize every element of $\theta$ with a probability that is decreasing in the node degree as

$$\theta_v = 1 - \frac{d(v)}{\Delta(G)} + s, \ \theta \leftarrow \frac{\theta}{\max_{v \in V} \theta_v},$$ \hspace{2cm} (5.12)
Algorithm 6 Finding MIS with dNN

Function: $\mathcal{I} = \text{dNN}(G, \alpha)$

1: construct $h$ from $G$ using (5.5), (5.6), (5.8), and (5.9)
2: initialize $\theta$ as in (5.12), $\mathcal{I}(\theta) = \emptyset$
3: while $\exists v \in V \setminus \mathcal{I}(\theta)$ s.t. $E(G[\mathcal{I}(\theta) \cup \{v\}] \neq \emptyset$
4: \hspace{1em} update $\theta \leftarrow \text{argmin}_{\theta \in [0,1]^n} |h(\theta) - h_d|^2$
5: \hspace{1em} obtain $\mathcal{I}(\theta) = \{v \in V \mid \theta_v \geq \alpha\}$

where we add a small value $s$ drawn from a uniform distribution over small positive bounds ($<< 0.1$) as part of the ADAM stochastic algorithm to improve performance when optimizing the loss function of $h$ [95].

The complexity of optimizing over the proposed dNN depends on the ADAM algorithm that solves a minimization problem with a number of box-constrained variables equal to the number of nodes in the graph. The ADAM optimizer is a stochastic gradient-based method by which, in every iteration (update of parameters), the complexity depends on the computation of the gradient $\nabla_\theta \mathcal{L}(h(\theta), f_h)$. This is of the same computational complexity as evaluating the objective in (5.11) [94].

Scaling up: Community Detection Approach

To handle large-scale graphs for the MIS problem, many techniques have been introduced in prior work, including the linear programming (LP) reduction, removal of pendant vertices, and other heuristics as presented in [79] and adopted in some of the most recent state-of-the-art methods. However, these techniques are only applicable to sparse graphs as pointed out in [78].

This motivates our work here on developing a reduction technique that is independent of the graph type and density. The proposed procedure described next utilizes multiple dNNs to handle large graphs, which is particularly useful when optimizing the parameters of one dNN representing the
entire graph requires a large amount of storage and computation.

We perform community detection [96] to partition the graph into communities, which are groups of nodes with dense connections internally and sparser connections between groups. Then, using Algorithm 6, we obtain a MIS for the subgraph induced by each community separately. Subsequently, a MIS is obtained for the full graph by processing the identified sets. More specifically, let \( C_i, i \in [r] \) denotes the set of nodes in community \( i \), where \( r \) is the total number of communities found by a community detection algorithm. The inter-cluster edge set

\[
R = \{(u, v) \in E \mid u \in C_i, v \in C_j, i \neq j\}, \tag{5.13}
\]

is the set of edges between nodes in different communities. For every \( C_i \), we construct a dNN \( h_i \) and obtain an MIS \( \mathcal{I}_i = \text{dNN}(G_i, \alpha) \), where \( G_i = G[C_i] \). The union of these sets is the set \( B = \bigcup_{i \in [r]} \mathcal{I}_i \).

Note that \( B \) is generally not an IS w.r.t. graph \( G \) since there could exist edges between nodes in the solution sets of two different communities. We call these the forbidden edges and define the set

\[
F = \{(u, v) \in R \mid u \in \mathcal{I}_i, v \in \mathcal{I}_j, i \neq j\}. \tag{5.14}
\]

In order to obtain a MIS w.r.t. \( G \), we need to handle all nodes with edges in the set \( F \). To this end, we develop the following procedure which processes every pair in \( F \) until an IS is obtained w.r.t. \( G \). First, we select a pair \((u, v) \in F \), then for every node \( q \in \{u, v\} \), we check if it can be replaced by a node from its neighborhood. A candidate replacement, \( w \in N(q) \), must be 1-tight, that is \(|B \cap N(w)| = 1\). If no replacements are found for either \( u \) or \( v \), we remove the node with the larger number of repetitions in \( F \). This is repeated until the set \( F \) is empty. The entire procedure is presented in Algorithm 7. In the case that the resulting set \( \mathcal{I} \) is only an IS w.r.t. \( G \), we obtain a MIS by executing Algorithm 6 on the subgraph induced by nodes that are neither in the solution
**Algorithm 7 Handling Forbidden edges.**

**Input:** Graph $G = (V, E), B, F$

**Output:** IS $\mathcal{I}$ on $G$

1: initialize $\mathcal{I} = B$

2: while $F \neq \emptyset$

3: select a pair $(u, v) \in F$, initialize ReplacementFlag = 0

4: for all $q \in \{u, v\}$

5: if $\exists w \in N(q)$ s.t. $|\mathcal{I} \cap N(w)| = 1$

6: replace $q$ by $w$, that is $\mathcal{I} \leftarrow \mathcal{I} \setminus \{q\}, \mathcal{I} \leftarrow \mathcal{I} \cup \{w\}$

7: update $F$, ReplacementFlag = 1

8: break for

9: if ReplacementFlag = 0 (no replacement is found)

10: remove either $u$ or $v$ depending on their repetitions in $F$

11: update $\mathcal{I}$ and $F$

nor in its neighborhood. In particular, $\mathcal{I}$ is updated as

$$\mathcal{I} \leftarrow \mathcal{I} \cup d\text{NN}(G[V \setminus (\mathcal{I} \cup N(\mathcal{I}))], \alpha).$$  \hspace{1cm} (5.15)

The complexity of the proposed CD approach depends on the CD algorithm utilized, the post-processing step presented in Algorithm 7, and Equation (5.15). The complexity of the Algorithm depends on the size of set $F$, and the number of removed and replaced nodes in steps 6 and 10, respectively. If replacements and/or removals result in a MIS, then (5.15) is not needed. In the case where the output of the algorithm is an IS, the number of removed and replaced nodes determine the size of graph $G[V \setminus (\mathcal{I} \cup N(\mathcal{I}))]$ whose nodes and edges determine the size of the dNN needed to obtain a MIS from IS in (5.15).

*Solution Improvement by dNNs*


**Algorithm 8** Solution improvement by dNNs  
**Input:** Graph $G = (V,E)$, Solution $\mathcal{I}$, $\lambda$, IncreaseStep  
**Output:** $\mathcal{I}^*$

1: initialize $\mathcal{I}^* = \mathcal{I}$
2: while stopping criteria is not satisfied
3:    obtain $\mathcal{U} \subset \mathcal{I}$ : $|\mathcal{U}| = \lambda, \forall u \in \mathcal{U}, v \in \mathcal{I} \setminus \mathcal{U}, d(u) \leq d(v)$
4:    obtain $\mathcal{I} \leftarrow \mathcal{I} \cup \text{dNN}(G[V \setminus (\mathcal{U} \cup N(\mathcal{U}))], \alpha)$
5:    if $|\mathcal{I}| > |\mathcal{I}^*|$ (update the optimal if $\mathcal{I}$ is of higher cardinality)
6:        update $\mathcal{I}^* = \mathcal{I}$
7:    if $|\mathcal{I}| \leq |\mathcal{I}^*|$ (restart from the current optimal)
8:        update $\mathcal{I} = \mathcal{I}^*$
9:        update $\lambda \leftarrow \lambda + \text{IncreaseStep}$

After applying the community detection algorithm and using Algorithm 6 for every resulting subgraph, Algorithm 7 along with a possible use of (5.15) are used to obtain MIS $\mathcal{I}$. Since high-degree nodes are less likely to be in a large IS, given graph $G$ and solution $\mathcal{I}$, we propose a solution improvement procedure that removes a set of low-degree nodes $\mathcal{U} \subset \mathcal{I}$, such that $|\mathcal{U}| = \lambda$, along with their neighbours $N(\mathcal{U})$ from the graph. We then apply our dNN on the reduced graph $G[V \setminus (\mathcal{U} \cup N(\mathcal{U}))]$ with different initial seed for $s$ as a form of simulated annealing [97]. This procedure is iteratively applied where we increase $\lambda$ at every iteration. The best solution is maintained until some stopping criteria is met. The procedure is given in Algorithm 8. While a similar criteria is used to select $\mathcal{U}$ in [78], their method recursively tries all reduction techniques on the reduced graph where $\lambda$ is fixed.

The number of computations needed for Algorithm 8 depend on $\lambda$ and the rate by which it increases as this determines the size of the subgraph on which the dNN procedure is applied at each iteration.
Experimental Evaluation

In this section, we evaluate the performance of our proposed method and present comparisons to state-of-the-art methods using synthetic graphs and real-world benchmarks.

Setup, Benchmarks, and Baselines

We process graphs using the NetworkX library [98] and use Tensorflow [99] to construct the dNN $h$. The initial learning rate for the ADAM optimizer is set to 0.1. To tune the parameters $\theta$ of $h$, a set of repeated samples of the pair $(e_n, -n^2/2)$ is used as the dataset. We set the probability threshold $\alpha = 0.5$ and use degree-based initialization. Experiments justifying our choice are presented in later subsections. For community detection, we use the Louvain algorithm [100] with a resolution factor of 1.3 for large-scale low-density graphs and 0.8 for high-density instances. For Algorithm 8, we choose $\lambda = 5$ and increase it by 1 in every iteration. The algorithm stops when the number of nodes in the reduced graph is below 20. The experiments are run using Python 3 and Intel(R) Core(TM) i9-9940 CPU @ 3.30GHz machine.

For low-density graphs, we incorporate the inexpensive and non-recursive LP graph reduction presented in [101] prior to performing community detection and constructing the enhanced dNN $h$. A half-integral solution (using values 0, 1/2, and 1), $x^* = \arg\max\{\sum_{v \in V} x_v \text{ s.t. } x_v \geq 0, \forall v \in V, x_v + x_u \leq 1, \forall (u, v) \in E\}$ is obtained using bipartite matching. The vertices that are members of set $T = \{v \in V \mid x^*_v = 1\}$ must be in the MIS and can thus be removed from $G$ together with their neighbors in $N(T)$. The solution obtained from training $h$ on $G[V \setminus (T \cup N(T))]$ is joined with nodes in $T$ to obtain the MIS for $G$. Furthermore, we implement the 2-improvement basic local search algorithm [93]. The foregoing techniques are also used in most of the state-of-the-art
solvers presented in [78, 84, 102].

As a benchmark, we use the social network graphs from the Stanford Large Network Dataset Collection given in SNAP [103]. In these graphs, the vertices represent people and the edges reflect their interactions. We also use the citation network graphs [104] for data collected from academic search engines. In these graphs, nodes represent documents and edges reflect their citations. Using the aforementioned benchmarks, we compare the performance of our proposed framework to multiple MIS solvers, including the GCN method [84], which is an ML-based approach, and the RL-based method S2V-DQN [89]. We also report results from the state-of-the-art MIS solver ReduMIS [78]. We use the size of the identified independent set to measure the quality of the solution for every baseline considered. Furthermore, results from solving the MIS Integer Linear Program (ILP) in (5.16) using CPLEX are also reported.

\[
\max_{x} \sum_{v \in V(G)} x_v \text{ subject to } x_v \in \{0, 1\}, \forall v \in V, x_v + x_u \leq 1, \forall (v, u) \in E. \tag{5.16}
\]

The aforementioned benchmarks are considered sparse graphs. Therefore, we also test our proposed method on higher-density graphs randomly generated from the Erdos-Renyi (ER) [105], BarboSI-Alber (BA) [106], Holme and Kim (HK) [107], and the Stochastic Block (SBM) [108] models.

**Results on SNAP and Citation Network Benchmarks**

In this subsection, we present the overall comparison results with the GCN, ReduMIS, and S2V-DQN methods along with the results from solving the MIS ILP in (5.16) using CPLEX. Columns 4 to 8 of Table 5.1 present the size of the found MIS. The results, other than the CPLEX ILP, reported for the baselines are obtained from Table 5 of [84].
Table 5.1: Comparison to state-of-the-art baselines using real-world benchmarks in terms of the size of the identified MIS.

| Dataset       | |V| | |E| | | | |GCN| |ReduMIS| |S2V-DQN| |CPLEX| |dNNs| |
|---------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bitcoin-alpha | 3783 | 14124 | 2718 | 2718 | 2705 | 2718 | 2718 |
| bitcoin-otc   | 5881 | 21492 | 4346 | 4346 | 4334 | 4346 | 4347 |
| wiki-Vote     | 7115 | 100762 | 4866 | 4866 | 4779 | 4866 | 4866 |
| soc-slashdot0811 | 73399 | 497274 | 53314 | 53314 | 52719 | 53314 | 53314 |
| soc-slashdot0922 | 82168 | 582533 | 56398 | 56398 | 55506 | 56398 | 56395 |
| soc-Epinions1  | 75579 | 405740 | 53599 | 53599 | 53089 | 53599 | 53598 |
| Citeseer      | 3327 | 4536 | 1867 | 1867 | 1705 | 1808 | 1866 |
| Cora          | 2708 | 5429 | 1451 | 1451 | 1381 | 1451 | 1451 |
| PubMed        | 19717 | 44327 | 15912 | 15912 | 15709 | 15912 | 15912 |

Table 5.2: Comparison to state-of-the-art baselines using synthetic graphs in terms of the average size of the MIS.

| Graph Type | |V| | |E| | | | | \( \mathbb{E}(|E|) \) | ReduMIS | GCN | CPLEX | dNNs |
|------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ER         | 100 (|p| = 0.1) | 496 | 30.5 | 30.5 | 30.5 | 30.5 | 30.5 |
| ER         | 100 (|p| = 0.2) | 975 | 20 | 19.5 | 20 | 20 | 20 |
| ER         | 200 (|p| = 0.1) | 1991.5 | 41 | 40 | 41 | 41 | 41 |
| ER         | 200 (|p| = 0.2) | 3983.5 | 25.5 | 24 | 25.5 | 25.5 | 25.5 |
| SBM        | 250 (|p| = 0.1) | 1857.5 | 61 | 60.5 | 60.5 | 61 |
| SBM        | 250 (|p| = 0.2) | 2431 | 51 | 50.5 | 51 | 51 | 51 |
| SBM        | 350 (|p| = 0.1) | 3614.5 | 68 | 68 | 66.5 | 68 |
| SBM        | 350 (|p| = 0.2) | 4826 | 55.5 | 55 | 53.5 | 55.5 | 55.5 |
| BA         | 100 | 2450 | 45 | 45 | 45 | 45 | 45 |
| BA         | 200 | 9950 | 95 | 95 | 95 | 95 | 95 |
| HK         | 100 | 2500 | 30 | 30 | 30 | 30 | 30 |
| HK         | 200 | 9900 | 60 | 60 | 60 | 60 | 60 |

We briefly describe the reduction techniques utilized by these baselines as they contribute significantly to their final results. All three methods remove pendent, unconfined, and twin vertices and utilize vertex folding for degree-2 nodes. Additional reductions are also considered in ReduMIS, including finding node alternatives, using packing constraints, and adopting the same LP MIS relaxation reduction we adopt in this chapter. We refer the interested reader to [79] for a thorough discussion of these reductions.
In all datasets considered, our method outperforms S2V-DQN. Our method performs mostly on par with ReduMIS and GCN, both of which yield identical results. This is observed as exact MIS sizes are obtained for bitcoin-alpha, Wiki-Vote, soc-slashdot0811, Cora, and PubMed. While scoring higher for bitcoin-otc, our method scores lower for soc-slashdot0922, soc-Epinions1, and Citeseer. When compared to the ILP solver, our method outperforms CPLEX on bitcoin-otc and Citeseer while performing on par for all the other instances other than soc-slashdot0922 and soc-Epinions.

**Results on Synthetic Graphs**

In this subsection, we compare our proposed method to ReduMIS, GCN, and CPLEX using random graphs generated from the ER, BA, and HK models. Every size reported in Table 5.2 represents the average of two random graphs from the model given in the first column. For ER, $p$ represents the probability of an edge being present. For BA, we use $\lfloor 0.5n \rfloor$ and $\lfloor 0.45n \rfloor$ edges to attach a new node to existing nodes. For HK, we add $\lfloor 0.3n \rfloor$ random edges to each new node and set the probability of adding a triangle after adding a random edge to 0.5. For SBM, we generate graphs with 5 clusters, where two nodes from the same cluster share an edge with probability $p$ (intra-cluster density) and two nodes from different clusters share an edge with probability $q = 0.05$ (inter-cluster density). We set a 10-minute time limit for ReduMIS, GCN, and CPLEX and present the size of the largest independent set found within that time. The random graph parameters are selected to yield high-density graphs relative to the earlier benchmarks. As shown, for all the considered random high-density graphs, on average we attain the same MIS sizes of ReduMIS, while performing on par or outperforming the state-of-the-art learning-based method GCN or CPLEX. This is achieved without the use of labeled graphs for training.
**Advantage of Degree-based Initialization**

This subsection highlights the positive impact of utilizing the degree-based initialization, measured by the size of the MIS found and the execution time required to obtain a solution. The results, presented in Figure 5.5, are compared to initializing $\theta$ from a uniform distribution over the interval $[0, 1]$. Every point in Figure 5.5 (top) (bottom) represents the average size of the found MIS (execution time) of 5 uniformly generated random graphs with $n$ nodes and $m = 2n$ edges. As observed, on average, our proposed method in (12) outperforms random initialization in terms of the average MIS size and also incurs shorter execution time.

![Figure 5.5](image-url)

Figure 5.5: Comparison between the proposed degree-based initialization and the initialization from a uniform random distribution in terms of the average size of the found MIS (top) and the average execution time (bottom).

**Impact of the choice of $\alpha$**

In this experiment, we examine the impact of the probability threshold $\alpha$ on the number of training steps required, the execution time, and the output value of the network. Figure 5.6 shows the results for graphs generated uniformly at random with $n = 100$ and $m = 500$. From top to bottom, the
results show the average number of tuning steps required (first), average execution time (second), average normalized output values given by $-2h(\theta)/k^2$ (third), and the average obtained MIS size $k$ (fourth). As observed, when $\alpha$ increases, the output $h(\theta)$ of the dNN also increases along with the number of tuning steps and the required run-time. However, beyond $\alpha \approx 0.55$, the average size $k$ of the obtained MIS does not change. Therefore, to increase the size of the MIS while reducing the run-time and the number of steps, we set $\alpha = 0.5$ in step 5 of Algorithm 1.

The re-scaled zoomed curves in the inset of Figure 5.6 (third) shows that the optimal normalized output value approaches 1 as $\alpha \to 1$. This verifies that the minimum value of $h(\theta)$ occurs when $\theta \in \{0, 1\}^n$. This result was established in the proof of Theorem 4 and Theorem 5.

**Discussion and Conclusion**

We presented a dataless, differentiable methodology for solving the Maximum Independent Set (MIS) problem that is radically different from existing techniques. The underpinning of our approach is a reduction from the MIS problem to an equivalent dataless Neural Network (dNN) constructed from the given graph. The parameters of this dNN are trained without requiring data, thereby setting our approach apart from learning-based methods like supervised, unsupervised, and reinforcement learning. In particular, training is conducted by applying backpropagation to a loss function defined entirely based on the structure of the given graph. We also presented an enhanced version of the dNN by incorporating the edges from the complement graph and exploiting the duality of the Maximum Clique (MC) and MIS problems. Additionally, we developed a reduction procedure that leverages a community detection algorithm to scale our approach to larger and higher-density graphs. Unlike previous reductions, the procedure is independent of the type of the graph and its density. Experimental results on both real and synthetic benchmarks demonstrate that our proposed method performs on par with state-of-the-art learning-based methods in terms of
Figure 5.6: The impact of the probability threshold $\alpha$ w.r.t. the average number of tuning steps (*first*), average execution time (*second*), average normalized output value with re-scaled zoomed values (*third*), and average size $k$ of the MIS found (*fourth*).

the size of the found MIS without requiring any training data.
LIST OF REFERENCES


