12-2020

Formal Verification of the Adversarial Robustness Property of Deep Neural Networks Through Dimension Reduction Heuristics, Refutation-based Abstraction, and Partitioning

Joshua Smith
Utah State University

FORMAL VERIFICATION OF THE ADVERSARIAL ROBUSTNESS PROPERTY OF DEEP NEURAL NETWORKS THROUGH DIMENSION REDUCTION HEURISTICS, REFUTATION-BASED ABSTRACTION, AND PARTITIONING

by

Joshua James Smith

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

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Zhen Zhang, Ph.D. Major Professor	Jacob Gunther, Ph.D. Committee Member
Koushik Chakraborty, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

Formal Verification of the Adversarial Robustness Property of Deep Neural Networks Through Dimension Reduction Heuristics, Refutation-based Abstraction, and Partitioning

by

Joshua James Smith, Master of Science

Utah State University, 2020

Major Professor: Zhen Zhang, Ph.D.
Department: Electrical and Computer Engineering

As neural networks find increasing use in safety-critical systems, the need to formally guarantee safety properties grows as well. Neural networks are susceptible to adversarial inputs which cause networks to behave unexpectedly. Adversarial inputs can be very prevalent and can even be generated through small, potentially random, perturbations to known and stable inputs. Local adversarial robustness is a formally defined and desirable safety property of neural networks that, when proven, ensures that a region in the input space of the network is free of adversarial inputs. Modern neural networks can contain millions of parameters and often operate on very high dimensional inputs. These characteristics pose unique challenges to the formal verification of properties such as local adversarial robustness. This thesis presents and evaluates three novel techniques for optimizing formal verification methods to better cope with these challenges: (1) specification-guided dimension reduction, (2) refutation-based abstraction through adversarial example generation, and (3) a framework that augments existing formal methods with refutation-based abstraction, input partitioning, and parallelism. Specification-guided dimension reduction is a method for ranking input dimensions based on their approximate contribution to the classification output of a neural network. This novel technique is compared with a state-of-the-art

method and shows improvements in efficiency and reports adversarial inputs to which the neural network is more susceptible on average. Leveraging the apparent prevalence of adversarial inputs, refutation-based abstraction through adversarial example generation is a novel optimization, performed before more exhaustive formal verification methods, that can rapidly disprove the local adversarial robustness property. The *randomized fast gradient sign method* (RFGSM) is a refutation-based abstraction algorithm, presented in this thesis, that better represents the region it abstracts by significantly increasing output variance while maintaining the success rate of its predecessor. Finally, a novel framework for augmenting existing formal methods with the proposed optimizations is presented. This framework uses refutation-based abstraction and input partitioning to provide a large number of adversarial inputs that are well-distributed throughout the region under test and to enable parallel verification of independent regions.

(62 pages)

PUBLIC ABSTRACT

Formal Verification of the Adversarial Robustness Property of Deep Neural Networks Through Dimension Reduction Heuristics, Refutation-based Abstraction, and Partitioning

Joshua James Smith

Neural networks are tools that are often used to perform functions such as object recognition in images, speech-to-text, and general data classification. Because neural networks have been successful at approximating these functions that are difficult to explicitly write, they are seeing increased usage in fields such as autonomous driving, airplane collision avoidance systems, and other safety-critical applications. Due to the risks involved with safety-critical systems, it is important to provide guarantees about the networks performance under certain conditions. As an example, it is critically important that self driving cars with neural network based vision systems correctly identify pedestrians 100% of the time. The ability to identify pedestrians correctly is considered a *safety property* of the neural network and this property must be rigorously verified to produce a guarantee of safe functionality. This thesis focuses on a safety property of neural networks called local adversarial robustness. Often, small changes or noise on the input of the network can cause it to behave unexpectedly. Water droplets on the lens of a camera that feeds images to a network for classification may render the classification output useless. When a network is locally robust to adversarial inputs it means that small changes to a known input do not cause the network to behave erratically. Due to some characteristics of neural networks, safety properties like local adversarial robustness are extremely difficult to verify. For example, changing the color of the pedestrians shirt to blue should not effect the network's classification. What about if the shirt is red? What about all the other colors? What about all the possible color combinations of shirts and pants? The complexity of verifying these safety properties grows very quickly.

This thesis proposes three novel methods for tackling some of the challenges related to verifying safety properties of neural networks. The first is a method to strategically select which dimensions in the input will be searched first. These dimensions are chosen by approximating how much each dimension contributes to the classification output. This helps to manage the issue of high dimensionality. This proposed method is compared with a state-of-the-art technique and shows improvements in efficiency and quality. The second contribution of this work is an abstraction technique that models regions in the input space by a set of potential adversarial inputs. This set of potential adversarial inputs can be generated and verified much quicker than the entire region. If an adversarial input is found in this set then more expensive verification techniques can be skipped because the result is already known. This thesis introduces the *randomized fast gradient sign method* (RFGSM) that better models regions than its predecessor through increased output variance and maintains its high success rate of adversarial input generation. The final contribution of this work is a framework that adds these previously mentioned optimizations to existing verification techniques. The framework also splits the region being tested up into smaller regions that can be verified simultaneously. The framework focuses on finding as many adversarial inputs as possible so that the network can be retrained to be more robust to them.

To the reader...

ACKNOWLEDGMENTS

I am truly appreciative of the great support and encouragement that I have received at the hands of so many people. First and foremost, I would like to express my deep gratitude to Professor Zhen Zhang. He has been a supportive tutor, an exacting leader, and a great friend. This work is the result of many hours of discussions, trials, and exploration and he was there for it all. I look forward with pleasure to our continuing research in this field.

Graduate school certainly demands great effort from its participants but it also requires great sacrifices from those who choose to love them. My wife, Mariah Smith, has been so supportive and has worked tirelessly to help our family reach this goal. We have two young children who are anything but easy to manage and she has been an incredible wife and mother throughout this effort. My parents, David and Marie Smith, along with my parents-in-law-but-more-so-by-love, Wayne and Karen Campbell, have encouraged and enabled this work and now celebrate its completion with us. Thank you for your unfeigned love.

I also wish to thank the professors and staff at Utah State University. They have provided me with a strong intellectual foundation that has proven to be valuable in industry and empowering in research. The professors who sit on my committee, Dr Koushik Chakraborty and Dr Jacob Gunther, have been especially influential in my education and I greatly admire their expertise. Tricia Brandenburg has also been such a great help in organizing these efforts and I appreciate her going above and beyond to help me feel on track and prepared.

This work has been supported by generous grants from Adobe. Viswanathan Swaminathan, a graduate of Utah State and employee of Adobe, graciously has collaborated in this research and paved the way for our partnership with Adobe. I thank them for their patronage.

Joshua James Smith

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

CIFAR	Canadian Institute For Advanced Research
DLV	Deep Learning Verifier [2]
DNN	deep neural network
FGSM	fast gradient sign method
GTSRB	German Traffic Sign Recognition Benchmark
JSMA	Jacobian-based saliency map algorithm
MILP	mixed integer linear programming
MNIST	Modified National Institute of Standards and Technology
ReLU	rectified linear unit
RFGSM	randomized fast gradient sign method
R+FGSM	random step followed by fast gradient sign method
SAT	Boolean satisfiability problems
SIFT	scale-invariant feature transform
SMT	satisfiability modulo theories

CHAPTER 1

INTRODUCTION

Advances in *deep neural networks* (DNNs) have increased their deployment in safety-critical systems such as vision perception modules for autonomous vehicles and airborne collision avoidance system controllers for unmanned aircraft [3]. Providing formal guarantees for safety properties of these neural networks is extremely important to ensure safe functionality and increase the utility of DNNs. Adversarial robustness is a desirable safety property that speaks to a systems ability to behave according to its specification when presented with adversarial input. Adversarial inputs may be crafted specifically to approximate a "worst case" or may naturally occur as a result of noise inherent to most systems. When a system is not robust to adversarial inputs, it may behave erratically and produce unexpected output which, in the case of safety-critical systems, poses a significant or even potentially catastrophic hazard.

This thesis focuses on formally verifying the local adversarial robustness safety property of neural networks. It has been demonstrated that this property is often quite weak in DNNs [1,4]. Also, modern neural networks can contain millions of parameters and generally have very high dimensional input and output spaces. These characteristics pose significant challenges to formally verifying even simple properties of DNNs. Existing formal methods must be specialized to cope with the unique challenges inherent to the formal verification of these high dimensional systems. This thesis presents and evaluates three novel techniques for improving formal verification of safety properties of neural networks: (1) specification-guided dimension reduction, (2) refutation-based abstraction through adversarial example generation, and (3) a framework for optimizing existing formal methods with refutation-based abstraction, input partitioning, parallelism, and customized data structures.

1.1 Specification-Guided Search

Dimension reduction is commonly used in fields such as computer vision or information retrieval to reduce the search space while maintaining important properties of the original dimensionality. Neural networks often deal with high dimensional data and are therefore great candidates for dimension reduction techniques. Huang et. al., in their tool called DLV [2], proposed a dimension reduction method based on ranking dimensions by their approximate contribution to the classification output of a network. Chapter 4 details a novel approach to dimension reduction that applies more generally to the classification problem, makes use of the classification regions defined by the training data or "specification" of the network, and shows improvement in run time and the production of more *convincing* adversarial examples over DLV.

1.2 Abstraction Through Adversarial Input Generation

Hypothesizing that adversarial inputs are quite common and well-distributed through the input space, the work of chapter 5 proposes the novel idea of performing refutation-based abstraction through adversarial input generation as an optimization to more exhaustive and expensive formal verification techniques. Adversarial example generation algorithms have been able to achieve very high success rates and some are even constant time algorithms [1,5,6]. A new algorithm, called RFGSM, is introduced that maintains the success rate and algorithmic complexity of its predecessor but increases the output variance significantly so that it is capable of generating a more representative set of adversarial examples as an abstraction for a region under test.

1.3 Refutation-based Abstraction Verification Framework With Partitioning

This thesis also presents a novel framework that empowers a formal verification strategy with the optimizations of refutation-based abstraction, input partitioning, and parallelism. Input partitioning enables subregions to be verified in parallel and also causes the abstraction method to be distributed over the entire region under test. The framework is designed to be modular such that it can be configured easily to handle new abstraction, partition-

ing, and verification strategies. A full implementation of the framework as well as the abstraction, verification, and partitioning strategies is evaluated Section 5.2.

1.4 Contributions

The work presented in this thesis provides the following five major contributions to the field of formal verification of safety properties of neural networks:

- Analysis of the effect of a novel dimension reduction technique on the efficiency of a state-of-the-art search method
- Introduction of a refutation-based abstraction technique that maintains the success rate and algorithmic complexity of an advanced adversarial example generation algorithm while increasing its variance through dimension ranking heuristics
- Definition of a generic framework and associated structures and algorithms that augment formal verification techniques with refutation-based abstraction, input partitioning, and parallelism
- Results of the proposed techniques when tested on standard classification benchmarks and compared to contemporary methods
- Open source implementations of all the detailed algorithms with documentation describing their utility, function, and extensibility

1.5 Summary of Chapters

Chapter 2 gives a brief introduction to neural networks and describes common structures, uses, training algorithms, and difficulties associated with formally proving their safety properties. Formal verification is also introduced with an emphasis on the challenges associated with applying formal methods to neural networks. An example is given of how safety properties can be rigorously encoded as well as a brief overview of the basic functionality of a solver in formal methods.

Chapter 3 is a literature review of the field of formal verification of safety properties of neural networks. Vulnerabilities called adversarial examples are reviewed along with work that has been done to reliably generate and characterize them. State-of-the-art techniques for verifying formalized properties of neural networks are compared and categorized to show how the work presented in Chapters 4 and 5 fits with the current state of the field. Challenges inherent to formally verifying neural networks are explained and an overview of how the work in this thesis addresses those challenges is provided.

Chapter 4 describes a novel dimension reduction technique that leverages classification regions defined by the training data or "specification" for a neural network. An implementation of this proposed algorithm is incorporated into a state-of-the-art verification tool called DLV [2] and evaluated for its effect on the tool's efficiency. Metrics for quantifying the results are introduced that aid in creating explanations for the new method's performance. The proposed dimension reduction technique shows improvements in run time and average confidence of adversarial examples on two of the three tested datasets and networks.

Building on the conclusions from the dimension reduction experiments, **Chapter 5** introduces a framework for adding optimizations such as region abstraction, parallel execution, and problem partitioning to formal verification strategies for neural networks. This chapter introduces the topics of region abstraction through adversarial example generation, dimension ranking as a generalization of dimension reduction, and input partitioning and how it enables parallelism. Several region abstraction and partitioning strategies are proposed and evaluated for desirable properties and the best performing methods are incorporated into the framework for a final test.

This thesis is organized into two main experiments detailed in Chapters 4 and 5 respectively. Results for each experiment are reported in these two chapters. **Chapter 6** summarizes the conclusions from each of these experiments, draws attention to important takeaways, and describes the future work associated with the findings.

CHAPTER 2
BACKGROUND

2.1 Deep Neural Networks

Deep neural networks (DNNs) are collections of operations and internal parameters that transform inputs through a series of layers and attempt to extract higher level abstractions and models of the data to produce outputs that approximate a function often exemplified by a set of labeled data points. DNNs are often described as being composed of neurons followed by activation functions which associate them, in structure and function, with the brain. *Neurons* are often linear operations which transform a value from a previous layer, with its associated weight and bias, and pass the result on to an *activation function* which is often a nonlinear operation such as a sigmoid or a *rectified linear unit* (ReLU). DNNs have been shown to excel in fields such as object classification in images, reinforcement learning, natural language processing, and computer vision. DNNs provide the structure for performing a function and/or extracting meaningful features but the actual learning is enabled by an appropriate loss function, access to data representative of the target function and associated output values, and a training algorithm. Loss functions provide a scalar value quantifying the difference between the current output and the target output. The training algorithm uses the representative data and loss function to make modifications to the internal parameters of the network in an attempt to minimize the loss function. Training a neural network is often a long process, requiring many iterations to converge to a satisfactory approximation.

The functions being modeled by modern DNNs are often extremely difficult to explicitly formalize (hence the use of function approximation and machine learning techniques). Likewise, properties of these functions are equally difficult to formalize [7]. For example, in image classification, how would one formally define a pedestrian? Given an image of a

pedestrian, most people could correctly label the image as containing a pedestrian and even provide reasons why but would these explanations be a complete specification of the concept? How would one mathematically formulate the explanations? If defining these formal specifications was possible, would there be a need for a machine learning solution? Because of these challenges, a neural network is generally evaluated by presenting it with labeled data that did not participate in its training and then observing how well the network is able to generalize to this new dataset. This evaluation technique is essentially equivalent to a non-exhaustive case testing scheme and leaves much to be desired. This thesis is aimed at providing strong formal guarantees of robustness properties of neural networks.

2.1.1 Adversarial Robustness

It has been shown that neural networks are susceptible to adversarial inputs that cause the network to break its specification. Adversarial inputs can be small perturbations of stable inputs that cause large and unexpected changes on the output. Figure 2.1 is the archetypal example of an adversarial input produced by imperceptibly perturbing a known input. These small perturbations can be especially dangerous because they can be modeled by random noise inherently found in most real world systems (e.g. rain drops on a camera lens). The set of all potential adversarial examples produced by slight manipulations to a known image is called a *region* and can be represented by a polytope in the input space. Adversarial robustness is a desirable property of neural networks that specifies that, for a given stable input and a region around it, there do not exist adversarial inputs inside the region. Formally verifying this property over increasing region sizes can give a robustness "metric" and counterexamples found during verification can be used to further train the network and increase its robustness [1]. Adversarial robustness for the input classification problem is also easily formalized and is defined in Equation 2.1. α is a point in the input space with known and correct classification. f represents the function of the neural network that transforms the input into a classification output y. η_α represents the region containing α over which the robustness property is to be verified.

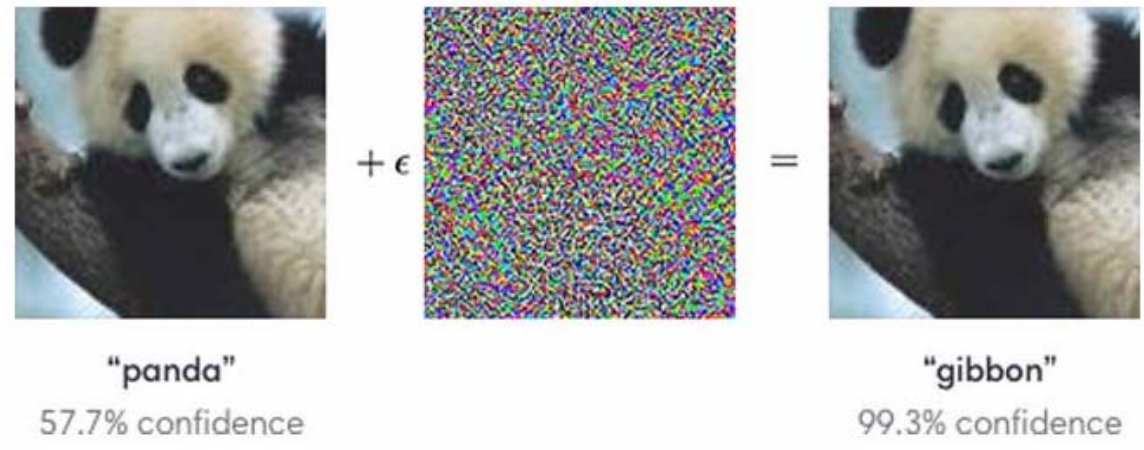


Fig. 2.1: Example of an adversarial input refuting the robustness property [1]

$$\forall x \in \eta_\alpha : \nexists\, y = f(x) \quad s.t.\ y \neq f(\alpha) \tag{2.1}$$

2.2 Formal Verification

To formally verify a system is to prove the satisfiability of mathematically defined properties. The field of formal verification has found great utility in static and dynamic analysis of software programs and circuits. Formal verification techniques are used to provide strong guarantees for safety-critical systems. There are many branches to this field including model checking, reachability analysis, Boolean satisfiability problems (SAT), satisfiability modulo theories, etc. Figure 2.2 is an overview of the function of a formal verifier. Formal descriptions of both a model and a property are given at the input and the verifier decides whether the property is satisfiable given the constraints and dynamics of the model. If the property is satisfiable then a proof or satisfying assignment is returned. If the property can not be satisfied then the verifier returns a counterexample.

Formal verification techniques are often specialized and tuned to deal with the challenges of specific applications. For example, *satisfiability modulo theories* (SMT) solvers are conglomerations of many background theories that specialize in solving decision problems in certain classes [8]. Given a set of linear inequalities and constraints that belong to the *linear*

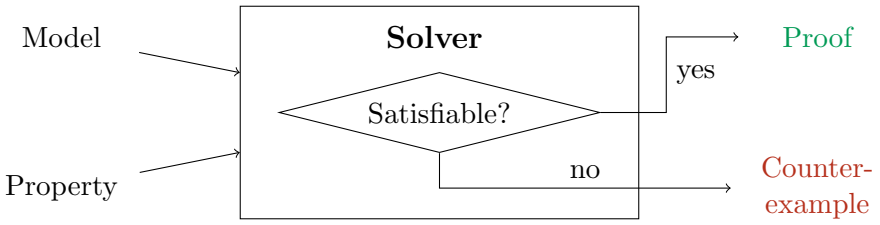


Fig. 2.2: Diagram of formal verification

programming class of problems, an SMT solver may use the *Simplex* [9] algorithm to decide satisfiability. Simplex is a decision method for solving conjunctions of linear inequalities introduced in 1947 and still used widely today. In another case, given a Boolean formula, the SMT solver may use a SAT solver to report a satisfying assignment or a counterexample.

Formally verifying properties of neural networks is a new field with unique challenges. Interesting properties of neural networks are quite difficult to formalize. Neural networks often have both high input dimensionality and a large number of internal parameters/operations. Also, neural networks often contain nonlinear operations that make them difficult to verify using existing techniques. Research in this field is focused on adapting existing verification techniques to cope with these new problems.

For example, Guy Katz, author of Reluplex [3], described how safety properties of an airplane navigation system implemented with a neural network could be formally encoded. Given a network N: $\bar{x} \rightarrow \bar{y} \quad (\bar{x} \in \mathbb{R}^m,\ \bar{y} \in \mathbb{R}^n)$, an input property $P(\bar{x})$ which characterizes the inputs being verified, and an output property $Q(\bar{y})$ which characterizes undesirable behavior, does there exist an input $\bar{x}_a$ and associated output $\bar{y}_a = N(\bar{x}_a)$ such that $\bar{x}_a$ satisfies P and $\bar{y}_a$ satisfies Q? In Katz's example, the input, $\bar{x}$, is a vector space where each dimension is associated with an input measurement such as the "distance to another airplane" and $\bar{y}$ is a vector where each element is associated with a control recommendation. The control recommendation associated with the largest value in $\bar{y}$ is the action taken. The following question is an interesting property of this system that may be verified: "If all other planes are far enough away, will the system always recommend no control changes, as it

should?" If $\bar{x}[0]$ represents the distance to another plane then P could be encoded as follows: $\bar{x}[0] \geq \mathit{SAFE\_DISTANCE}$. In this same example, if $\bar{y}[0]$ is the output recommending no control changes, Q can be encoded as follows: $(\bar{y}[0] \leq \bar{y}[1]) \vee (\bar{y}[0] \leq \bar{y}[2]) \vee (\bar{y}[0] \leq \bar{y}[3]) \vee \cdots \vee (\bar{y}[0] \leq \bar{y}[n])$. The formal model of the neural network and the conjunction of properties P and Q can now be given to an SMT solver that will search for a satisfying assignment to $\bar{x}$ [10]. The formal models of neural networks often contain so many parameters that modern SMT solvers cannot feasibly verify even simple properties, like this example, in reasonable amounts of time.

CHAPTER 3
LITERATURE REVIEW

3.1 Adversarial Examples

It has been shown that most modern DNNs and linear classifiers are susceptible to adversarial examples [4] [1] [11]. Adversarial examples are defined as perturbations of known inputs that result in large and unexpected changes on the output. Goodfellow et. al. have shown that adversarial examples can transcend neural network architectures and even training datasets by testing generated adversarial examples on different networks trained on different subsets of data [1]. Susceptibility to adversarial perturbations has been attributed to linear behavior in high dimensional space [1] and feature embedding instability [12].

Many techniques have been employed to make neural networks more robust. Goodfellow et. al. [1] show that training on adversarial examples can help to regularize a network and, with this rational, develop a method of generating adversarial examples called the *Fast Gradient Sign Method* (FGSM). This method is a constant time algorithm and produces examples with high confidence when applied to many networks and datasets including GoogLeNet on ImageNet images.

Papernot et al. develop a similar method based on a function they define as the forward derivative, related to the Jacobian, of a neural network called the *Jacobian Saliency Map Algorithm* (JSMA) [5]. JSMA has another feature that allows adversarial inputs to be generated for a specific target class. They show results where MNIST [13] images (handwritten digit dataset) are modified to cause the network to misclassify the images as each of the other available classes in turn. Another significant contribution of their work is a categorization of the adversarial goals. Papernot et. al. suggest four goals that successively increase in difficulty and intricacy: (1) confidence reduction, (2) misclassification, (3) targeted misclassification, and (4) source/target misclassification. JSMA falls into the

final category due to its ability to reliably manipulate a source image toward a target class. JSMA is a bounded iterative algorithm with no guarantee of successful misclassification but shows promising experimental results. FGSM, mentioned previously, pertains to the second goal because it is not directed and simply attempts to alter the classification of the input under consideration.

Kurakin et. al. introduce the *Iterative Least-Likely Class Method* for producing adversarial inputs [6]. Using their method as a template, adversarial examples can be created in real world scenarios (e.g. physically modifying a road sign) that can reliably fool neural networks. They also demonstrate that, due to an experimentally observed transferability property [1], adversarial examples can transcend differences in neural network architecture and training data. This finding enables black box attacks where adversarial examples can be generated on one neural network and then applied to another without knowledge of the second's properties. Kurakin, Goodfellow, and Bengio further reveal the utility of adversarial example generation along with some interesting properties [14]. Using adversarial training, they were able to "significantly increase robustness against adversarial examples generated by FGSM" for an Inception v3 model. They brought more light to the transferability property of adversarial inputs to different models, architectures, and networks, even when trained on different subsets of a dataset. Their experiments also show that networks with higher numbers of parameters are generally more robust to adversarial examples compared to networks of the same architecture but fewer parameters. A slight modification to their FGSM algorithm revealed a single step source/target misclassification algorithm. This algorithm suffers from a relatively low success rate. The Iterative Least-Likely Class Method is an iterative version of this single step algorithm that can achieve misclassification success in 99% of cases.

Attack algorithms such as FGSM and JSMA are "white box" approaches that require knowledge of the local gradient around the input to be perturbed. Tramèr et. al. [15] discover that local loss function gradients can contain "sharp" curvature artifacts that do not correctly represent the direction of steepest ascent on a larger scale. They suggest that

a small random step should be prepended to these gradient-dependent white box algorithms in order to initially escape the sharp featured region and move to an area where the local gradient is more representative of the gradient on a larger scale. This modified version of FGSM is named R+FGSM. The input is first perturbed in a random direction and then FGSM is applied on the new input. Results show that R+FGSM causes higher error rates on five different MNIST classifiers. These results lead the authors to state that often "local gradients learned by the adversarially trained model are worse than random directions for finding adversarial examples!" This statement highlights a weakness in the significance of very localized loss function gradients. Their findings reveal that adding randomness did not significantly decrease the success rate of these one step algorithms. This is a key insight used to justify some of the related work described later in this paper.

3.2 Formal Verification of Neural Networks

Formal verification is "a field of computer science and engineering concerned with the rigorous mathematical specification, design, and verification of systems" [7]. Seshia et al. identify five challenges to using formal methods to verify machine learning techniques. The two most applicable challenges to neural networks are [7]:

1. Computational engines (SAT, SMT (see Acronyms), and model checking), in their current state, do not scale well to the high dimensionality of machine learning techniques (e.g. state space explosion).

2. The formal specification of correctness is difficult to explicitly define in systems that are meant to model human cognition. This is demonstrated by questions like, "What is the formal specification of an object such as an apple and how can that formal specification be applied to image data?"

Due to these challenges and others, the more general safety properties (e.g. proof of correctness over all possible scene changes or camera orientations in image classification) are not currently formally verified but are partially quantified using testing and verification

datasets. Much of the research in the formal verification of safety properties of neural networks has focused on adversarial robustness. The adversarial robustness property is formally specified and, under certain constraints on the number of parameters in the network and/or the activation functions, traditional SAT/SMT solvers have shown competency [3].

In a review of methods for verifying properties of neural networks, Liu et. al. [16] divide several state-of-art methods into three categories which describe the techniques taken advantage of to perform verification: (1) Search, (2) Optimization, and (3) Reachability.

3.2.1 Search Methods

Search methods attempt to falsify the assertion by searching for a counter-example. Some methods search the input space and use dimension reducing heuristics to make this a more feasible approach [2, 17]. Others encode the property as a Boolean satisfiability problem and use SAT or SMT solvers to guide the search [3]. Search methods often carry strong assumptions about the specific functions implementing the neural network or the conditions of the safety property being verified. These strong assumptions can effect the completeness of the algorithms.

Katz et. al. develop a solution for the first challenge by optimizing an SMT solver for ReLU activation functions [3]. ReLU functions are piecewise linear and composed of two linear functions representing "off" and "passthrough" states. Realizing that presenting a normal SMT solver with a DNN with n ReLU nodes would result in the solver splitting the problem into 2^n sub-problems immediately [3], Katz et al. implement into the solver the ability to wait to resolve ReLU functions until they are shown to break the model a certain number of times. By doing this, they show that, in general, only about 20% of the ReLU nodes needed to be treated as piecewise linear and thereby reduce the complexity of the verification problem enough to handle large ReLU networks [3]. This tool is called Reluplex because it is a modification of Simplex [18], an algorithm for solving problems in linear programming, optimized for ReLU functions. This method is a very important breakthrough in the field but still is limited to piecewise linear activation functions. In 2019, Katz et. al. introduce a verification framework built on Reluplex, called Marabou [19], that

utilizes parallelism to handle larger neural networks. They also explain that the framework is adapted to other activation functions through the use of piecewise linear approximations.

Huang et. al. attack the same challenge from a different perspective. Realizing that exploring the entire region of the input space being verified for robustness exhaustively to search for adversarial examples is implausible, they develop the idea of manipulations (i.e. small, discrete changes) to the tested input that model perturbations. By repeatedly applying these nondeterministic manipulations, the region is discretized into a manageable set of testable activations that can be searched for adversarial examples. The tool developed from this innovation is called Deep Learning Verifier or DLV [2]. Also, to further decrease the number of searched dimensions, DLV employs a heuristic ranking of dimensions based on the probability that a dimension, when manipulated, would result in an adversarial example. The z3 SMT solver [20] is used to encode manipulations and map tested inputs through layers of the network.

Realizing that DLV provided guarantees but only with respect to a set of discrete manipulations, Wicker and Huang et al. move next into the area of testing. Limiting research to networks processing image data, they develop a method for using features extracted using a *Scale-Invariant Feature Transform* (SIFT) [21] to guide a search for adversarial examples. By using SIFT, an algorithm which models the feature extraction of human perception, their method generates a saliency map of pixels and then samples from that distribution. The generated saliency map assigns high probability to the strongest features and pixels are sampled from this distribution then manipulate to find adversarial examples [17]. This method shows results where adversarial examples are found faster than the method implemented by DLV and is sound but not complete in proving nonexistence of adversarial examples.

3.2.2 Optimization Methods

Optimization methods attempt to falsify the assertion by treating the function represented by the neural network as a constraint in an optimization problem [16]. *Mixed integer linear programming*, MILP, has been a popular approach to verifying the adversarial ro-

bustness property of neural networks that has shown great promise. Tjeng et. al. reveal a tool named MIPVerify [22] which is able to verify the robustness property of networks with over 100,000 ReLU functions. Their technique is sound and complete but also limited to piecewise linear feed-forward neural networks. This limitation is common to many other verification techniques that utilize MILP. NSVerify, proposed by Lomuscio et. al. [23], is a very similar approach that shares the same limitations.

3.2.3 Reachability Methods

Because exhaustive search does not scale well to higher dimensional networks without the compromise of discretization, research has been done into bounds analysis of neural networks. Reachability methods use properties of neural networks, such as Lipschitz continuity, to provide bounds on the output. If successful, the bounds can indicate that certain unsafe states are unreachable by any inputs in a given region and that the region is therefore safe.

The AI^2 tool [24,25] (now subsumed by the tool, ERAN[1]) creates abstract transformers for common neural network layers. Verification is done by letting polyhedra representing input regions propagate through abstract layers of a DNN then performing reachability analysis on the resulting polyhedra. This method, however, may introduce spurious counter-examples due to the neural network layer abstraction techniques.

Ruan et. al. prove that many common network layers and activation functions are Lipschitz continuous [26]. When a function is Lipschitz continuous, its derivative can be bounded by a value K such that $||\frac{\partial f(x)}{\partial x} \leq K||$ [26]. Any K that fits the above definition is called a Lipschitz constant. K_{best} is defined as follows $K_{best} = max(\frac{\partial f(x)}{\partial x})$. By using a value, K, close to K_{best}, their method produces an overestimated upper and lower bound for an output dimension with respect to a set of input dimensions. This is an extremely strong safety guarantee but it does rely on a close approximation of K_{best} to be useful in most cases. They outline an algorithm for iteratively refining a Lipschitz constant to approach

[1] https://github.com/eth-sri/eran

K_{best} but it is quite expensive and scales with the number of dimensions in the input space. This method has been implemented in the DeepGo tool.

Wang et. al. [27] present a method, called ReluVal, of providing bounds on the output of ReLU networks through symbolic interval analysis. They make use of interval arithmetic, Lipschitz continuity, symbolic intervals, and iterative interval refinement to find rigorous bounds on the DNN output. Their method requires that the network is composed of linear transformations such as addition and multiplication and very simple non-linear functions such as ReLU which limits its scope and applicability. Also, as mentioned previously, determining accurate Lipschitz constants for neural networks with high dimensional input spaces is an expensive operation and this operation is required to make useful refinements to output bounds. The authors introduce the idea of iterative interval refinement which is the process of splitting regions in the input into smaller subregions to get more accurate bounds for each. This processes is predicated on the principle that "the dependency error for Lipschitz continuous functions decreases as the width of intervals decreases [27]." Iterative refinement plays a key role in the success of the algorithms presented in this work and is an extremely useful tool for analyzing large regions on the input space of neural networks.

Weng et. al. produce two algorithms, Fast-Lin and Fast-Lip. They follow a similar objective as [26] but achieve a very accurate lower bound for a given output and set of input dimensions in tens of seconds for very large ReLU networks on a single CPU [28]. The Fast-Lin and Fast-Lip methods outperform their previous method CLEVER [29], linear programming based methods LP and LP-Full, Reluplex, Lipschitz based method Op-norm, and attack algorithms *Attacks*. However, they are currently limited to feed-forward ReLU networks.

3.3 Differentiation

Many of the tools above require that the neural networks being verified have certain properties such as Lipschitz continuity, be composed of strictly ReLU or simple piecewise linear activations, etc. This thesis proposes verification methods that do not place restrictions on the network architecture or operations other than those required by stochastic

gradient descent. Also, the framework presented in this work is modular and compatible with many of the above mentioned verification techniques. If a certain technique with assumptions on the network in question functions better in a certain case, that verification strategy can be used rather than the default, non-assuming strategy. It can be said that the proposed verification framework is oblivious to the specific verification strategy.

Another key difference between this proposed framework and the other methods that have been mentioned is its focus on adversarial example generation and reporting. In the tests and results demonstrated in this work, it is observed that adversarial examples are rather common, easily generated, and well distributed through the input space of neural networks. With this in mind, one must ask oneself, "If the likelihood of a region being free of adversarial examples is quite low, what results of a verification of that region will be most useful to me?" Many of the methods mentioned above focus on reporting either a successful verification (no adversarial examples found) or the first counter-example discovered. With the likelihood of the prior being quite low, the common result is a single counter-example. As shown by Goodfellow et. al. [1], retraining neural networks on adversarial examples can improve their robustness. The proposed framework focuses on generating as *many* adversarial examples as possible so that the results can have better utility in continued training.

The proposed framework also implements a novel optimization that combines the fields of adversarial example generation and formal verification of adversarial robustness. Adversarial example generation algorithms, such as FGSM [1] and JSMA [5], have achieved extremely high success rates on rapidly generated attacks. This work details a modification to FGSM that (1) maintains its success rate, and (2) increases the variance of repeatedly generated adversarial examples. This optimization reduces the need for more expensive formal verification techniques by generating counter-examples.

CHAPTER 4

SPECIFICATION-GUIDED SEARCH HEURISTIC

Providing safety guarantees to DNNs is challenging [7] due to their high dimensionality, difficult to formalize properties, and stochastic training methods. The high dimensionality of image data makes it computationally impractical for direct verification via search methods. To alleviate this problem, the *Deep Learning Verifier* (DLV) [2] selects "features", i.e., subsets of the total dimensions in the input space, that offer the highest probability of producing adversarial examples. It selects dimensions with the greatest absolute difference from the average value of all the dimensions in the input vector. This dimension selection method is summarized in Algorithm 4.1. The key assumption of this technique is that dimensions containing values that are vastly different from the average value of all the input dimensions contain the most defining information for the network's classification of that input. This assumption is plausible in the following case. Given an image of a brown cat in a green grass field, the image is to be classified as a cat and the pixels in the image that contain the important features of a cat are those most different from the overall average color or value (i.e., the green grass). However, this assumption may fail in cases where inputs are made up of data points with differing scales and meanings. For example, if one dimension of an input contains the price of a house and another the number of rooms in the house, the average value does not correctly represent the value of background or non-defining data.

4.1 Methodology

This chapter presents a feature dimension selection method guided by training data which improves efficiency and reliability over DLV. Here the specification refers to the training dataset for data-driven models such as DNNs [30]. The proposed method assumes that (1) class regions can be characterized by averaging all training set samples pertaining to

a given class, and (2) that dimensions containing the most defining information dominate the magnitude of the difference vector between the input under verification and the "characteristic input" of the closest incorrect class. This method explores the selected feature dimensions that best define this difference to efficiently generate adversarial examples. The implementation of this method extends the DLV tool, and it either detects adversarial examples with respect to local input perturbations, or guarantees non-existence of misclassifications with the same guarantees as DLV, namely safety with respect to a region and a set of manipulations.

Algorithm 4.1: DLV feature dimensions selection algorithm

Input: dims: Number of dimensions to select
Input: a: Tested input
Input: n: Length of a
Output: Dims: Set of dimension indices to be manipulated

$D = [\,]$
Dims $= \emptyset$
avg $= \frac{1}{n}\sum_{i=1}^{n} a[i]$
for $j = 1;\ j \leqslant n;\ j = j + 1$ **do**
  D.append((|avg$-a[j]$|, j))
Sort D by the |avg$-a[j]$| element in descending order
for $k = 1;\ k \leqslant dims;\ k = k + 1$ **do**
  Dims.insert($D[k][2]$)
return Dims

Improving on the feature dimension optimization described by [2] and the findings of [1, 17], we introduce a specification-guided feature dimension selection algorithm that efficiently generates and detects adversarial examples with higher confidence. This method is based on the observation that adversarial examples are often classified consistently (as the same class) by networks trained on different subsets of the training data and with different architectures [1]. It can be reasonably expected that searching for adversarial examples should be guided by the classification regions, assuming that the adversarial examples' common class is often the class closest to but not associated with the tested input.

Algorithm 4.2 describes the specification-guided feature dimension selection method.

The test input (input around which adversarial robustness is being evaluated) is a_i, where i references the index of the class to which it pertains. First, each set of training data pertaining to class j is averaged into a single characteristic input c_j. The distance d_j is the difference between each c_j and the input under test a_i. All d_j's, except d_i, are placed in a list D. d_i is excluded because the goal is to find the nearest class to perturb other than class i, to which a pertains. The distance vector d_{min} with the smallest magnitude is selected out of D and used to calculate the feature dimensions. Then the first $dims$ dimensions of d_{min} with the largest magnitude are chosen to be manipulated. These dimensions are then explored using the same method as the original DLV tool, namely a set of nondeterministic modifications to the selected feature dimensions repeatedly applied to cover the region around the tested input.

It should be noted that the proposed method inherits the formal guarantees of the original DLV method [2]. As the basic strategy of DLV is a discrete search, the guarantee provided is safety (absence of adversarial examples) with respect to a region and the set of manipulations used to create the discrete tested values.

Figure 4.1 gives a graphical illustration of the proposed algorithm. Although the figure is two-dimensional, the algorithm applies in n dimensional space. The colored regions represent the true classification regions for a given classification problem. Each characteristic input c_j $(1 \leqslant j \leqslant 5)$ is approximated through averaging training data pertaining to that region or class. a_4 is the input being tested for adjacent adversarial examples. Each difference vector d_j is calculated by subtracting a_4 from each c_j. In this case, d_4 is removed from consideration as it pertains to the same class as the tested input a_4. The shortest remaining distance vector is d_3 by visual inspection. The direction of d_3 is dominated mostly by its horizontal component. The proposed method is built on the assumption that the horizontal dimension is therefore the most likely to reveal adversarial examples when manipulated.

4.2 Results

The following metrics are evaluated and compared to the DLV tool [2], whose feature dimension selection algorithm is detailed in Algorithm 4.1. The implementation of the

Algorithm 4.2: Specification-guided feature dimensions selection

Input: $dims$: number of dimensions to select
Input: a_i: Test input which pertains to class i
Input: $\bar{X}$: Dictionary of training data
Input: n: Number of classes of training data
Output: Dims: Set of dimension indices to be manipulated

$D = [\,]$
$L = [\,]$
Dims $= \emptyset$
for $j = 1;\ j \leqslant n;\ j = j + 1$ **do**
    if $i = j$ **then**
        continue; // Do not consider distance vector d_i
    c_j =avg($\bar{X}_j$); // $\bar{X}_j$: all training data labeled as class j
    $d_j = c_j - a_i$; // d_j: distance (class average to tested input)
    D.append(($||d_j||, d_j$)); // Append tuple (magnitude and d_j)
Sort D by $||d_j||$ element in ascending order
$d_{min} = D[1][2]$; // Shortest distance between any c_j and a_i
$m =$ number of elements in d_{min}
for $k = 1;\ k \leqslant m;\ k = k + 1$ **do**
    $t = (d_{min}[k], k)$
    L.append(t)
Sort L by first element of t in descending order
for $l = 1;\ l \leqslant dims;\ l = l + 1$ **do**
    Dims.insert($L[l][2]$); // Insert dimension index from sorted L
return Dims

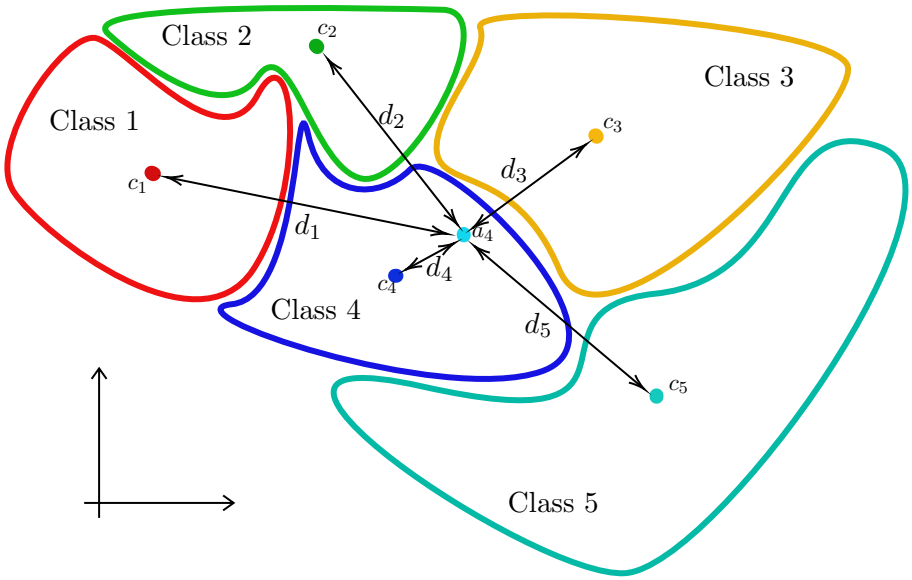


Fig. 4.1: Graphical view of Algorithm 4.2

proposed method is freely available on Github. [1]

- **Running time** measures the time from algorithm start to finding the first adversarial example. The proposed method is expected to reduce the running time by manipulating dimensions that contain more defining information between classes and are therefore more likely to result in adversarial examples.
- **Manipulation percentage** is a measure of the percentage difference, pixel by pixel, between the original input and the found adversarial example. Ideally, the proposed method would result in adversarial examples that have a lower manipulation percentage.
- **Euclidean distance** is the distance between the adversarial example found and the original input interpreting the two as points in the input space. The goal of the proposed approach is to lower this measure such that adversarial examples are found more quickly and closer to the tested input.
- **Adversarial example confidence** is the confidence of the adversarial example detected on the network output. This metric is more narrow than the others in that it only applies to a neural network trained for a classification problem. A high confidence value means that the network had a high certainty of the misclassification. Adversarial examples found with higher confidences are more valuable because they represent extremely weak points for the network. The proposed solution aims at increasing this metric.

The key assumption of this work is that the feature dimensions found by the specification-guided search heuristic have a higher probability of producing adversarial examples when manipulated than those found by the DLV method (shown in Algorithm 4.1). Given that the results of both techniques are collected on the same hardware and very similar system states, the key indicator of success is a faster running time on average. The other metrics (manipulation percentage, Euclidean distance, and adversarial example confidence) are interesting to observe as they shed some light on the results but do not directly impact the validity of the assumption.

[1] https://github.com/formal-verification-research/DLV_intellifeatures

The two methods are evaluated on the following three datasets: Modified National Institute of Standards and Technology (*MNIST*), Canadian Institute for Advanced Research 10 (*CIFAR-10*), and German Traffic Sign Recognition Benchmark (*GTSRB*). MNIST is a dataset containing images of hand-written digits 0-9. CIFAR-10 is a subset of the "80 million tiny images" dataset and contains 10 classes of small colour images of objects like cars, airplanes, and animals. GTSRB is a dataset containing 43 classes of the German traffic signs. Results are generated on a machine with an AMD Ryzen Threadripper 12-Core 3.5 GHz Processor and 132 GB memory, running Debian GNU/Linux 9 (stretch). Both tools are provided with the same test inputs.

4.2.1 MNIST

Table 4.1 shows results comparison using five feature dimensions on the MNIST dataset. Results are averages of twenty test inputs. Column "IntelliFeature" reports results for the proposed method. It achieves almost fifty percent increase in the adversarial example confidence and slight decrease in the Euclidean distance, without losing much running time or manipulation percentage.

Table 4.1: MNIST results comparison with five feature dimensions.

	DLV	**IntelliFeature**	**Improvement**
Running Time (seconds)	39	43	-10.26%
Manipulation Percentage	10.04%	11.42%	-13.75%
Euclidean Distance	7.68	6.71	12.63%
Adversarial Example Confidence	34.24%	51.06%	49.12%

Table 4.2 shows significant improvements achieved by the proposed method. The number of feature dimensions is increased to ten in this experiment. Results are averaged across 50 test images. Running time decreases by over ten percent, as a result of lower manipulation percentages and nearly twenty-five percent drop in the Euclidean distance. In addition to the performance gain, confidence in adversarial examples improves over forty-three percent. Comparing results in Table 4.1 and 4.2, it can be concluded that the specification-guided method significantly improves scalability to higher dimensional features than the original

DLV. We believe this is due to the original assumption that the selected feature dimensions have a higher probability of revealing adversarial examples than those chosen using the method described in Algorithm 4.1.

Table 4.2: MNIST results comparison with ten feature dimensions.

	DLV	**IntelliFeature**	**Improvement**
Running Time (seconds)	689	620	10.01%
Manipulation Percentage	9.97%	9.34%	6.32%
Euclidean Distance	7.75	5.85	24.52%
Adversarial Example Confidence	44.00%	63.00%	43.18%

4.2.2 CIFAR-10

CIFAR-10 is a labeled subset of the 80 million tiny images dataset with 60,000 images in 10 classes. A convolutional neural network consisting of 18 layers was trained on the dataset, and then evaluated using the proposed specification-guided method. The results are averaged across 50 test images, and are enumerated in Table 4.3.

Table 4.3: CIFAR-10 results comparison with five feature dimensions.

	DLV	**IntelliFeature**	**Improvement**
Running Time (seconds)	852	694	18.54%
Manipulation Percentage	23.35%	8.32%	64.37%
Euclidean Distance	6.11	5.80	5.07%
Adversarial Example Confidence	46.73%	45.19%	-3.30%

These results show an interesting shift believed to be an artifact of the fundamental difference between intensity data (MNIST) and color data (CIFAR-10). When feature dimensions are selected using the proposed specification-guided method in intensity data, manipulations on those features move only in two directions, higher or lower intensity. This significantly increases the probability of a manipulation moving in the direction of another classification region, and therefore transforming the image to look increasingly like the next closest region. In three-channel or color data, introducing hue and saturation as two new degrees of freedom lowers the probability that a manipulation chosen by the algorithm will

increase the likeness between an image and the next closest class. This is believed to be the reason why a relatively consistent average adversarial confidence is seen in CIFAR-10 and an extremely large confidence gap exists for MNIST.

The specification-guided method demonstrates superiority in selecting dimensions that are most likely to reveal adversarial examples when manipulated, indicating that the original assumption holds.

4.2.3 GTSRB

The GTSRB dataset consists of more than 50,000 labeled images in 43 classes. A convolutional neural network with 24 layers was trained on this dataset and evaluated using the proposed method to generate the results in Table 4.4. Results show the average across 50 test images.

Table 4.4: GTSRB results comparison with five feature dimensions.

	DLV	**IntelliFeature**	**Improvement**
Running Time (seconds)	825	152	81.58%
Manipulation Percentage	21.39%	6.44%	69.89%
Euclidean Distance	4.01	6.02	-50.12%
Adversarial Example Confidence	48.11%	54.21%	12.68%

Table 4.4 shows significant improvement in running time, manipulation percentage, and adversarial example confidence of our method. The low manipulation percentage indicates that manipulations are confined to a small subset of the dimensions of the image. The higher Euclidean distance implies that manipulations occur repeatedly to that small subset of dimensions rather than being spread across the input, thereby moving further from the initial input at a faster rate. This small subset of dimensions is a result of the feature dimension selection algorithm. These results show that the proposed specification-guided method produces dimensions with high probability of containing adversarial examples.

4.3 Conclusion

Extending the idea of "feature dimensions" introduced by Huang et. al. [2], the

specification-guided dimension ranking heuristic demonstrates improvements over the method detailed in Algorithm 4.1 by selecting dimensions that, when manipulated, have higher probability of revealing adversarial examples. This conclusion manifests in a faster run time on two of the three tested datasets and significant improvements in the confidence of generated adversarial examples.

This experiment shows that dimension reduction is a useful tool in adversarial search and that different heuristics for selecting dimensions can have significant effects on the efficiency of the search itself as well as the quality of found adversarial examples. Dimension reduction does introduce problems with the completeness of the algorithm and should therefore be used as an optimization with a fallback to a more complete formal verification method.

CHAPTER 5

REFUTATION-BASED ABSTRACTION PARTITIONING FRAMEWORK FOR FORMAL VERIFICATION OF DEEP NEURAL NETWORKS

The experiments detailed in Chapter 4 highlighted several properties of adversarial examples and neural networks that guide the work of this chapter. The following properties were concluded from those experiments:

1. Dimension ranking heuristics can have a significant effect on the success rate of a search for adversarial examples.

2. Dimension ranking can reduce dimensionality while maintaining or increasing efficiency.

3. Adversarial examples are often quite prevalent and easy to find and/or generate.

These observations sparked the hypothesis that proving a region in the input space to be unsafe can be made more efficient by first generating and testing potential adversarial examples and then falling back to robust verification on failure. The generated potential adversarial examples become a refutation-based abstraction of the region they cover that, when verified, results in either strong counter examples of the robustness property or an indication that more exhaustive verification is required.

Iteratively applying the process of refutation-based abstraction with a formal verification fallback to subsequent partitions of a region results in the following positive outcomes: (1) a covering set of adversarial examples that can be used in retraining to increase adversarial robustness, (2) known safe regions when refined small enough to formally verify, and (3) known unsafe regions which can be used for dataset augmentation. This chapter presents a novel framework that implements this algorithm while providing for easily interchangeable strategies for performing abstraction, partitioning, and verification. Evaluation is performed on a number of proposed abstraction strategies including a modification to

FGSM [1], called *RFGSM*, that allows for a greater variance of generated adversarial examples, thereby increasing their coverage of and ability to represent the abstracted regions. The effect of different partitioning strategies is analyzed and results of the framework are presented for three different datasets and associated deep neural networks: MNIST, GTSRB, and CIFAR-10.

5.1 Framework Description

The framework operates on three fundamental principles: verification, partitioning, and abstraction. During verification, a region is assigned a label of `safe`, `unsafe`, or `unknown`. The verification strategy should scale in algorithmic complexity directly proportional to the size of the region and should be able to determine if a definite solution (`safe` or `unsafe`) can be found in a *reasonable* amount of time. Through experimentation, a reasonable amount of time is on the order of seconds or less. If the verification strategy cannot determine a solution quickly on the current region size it returns `unknown`.

When a region is too large to be verified, it is partitioned into a set of smaller subregions, each of which potentially allows more efficient verification. Partitioning strategies are discussed in more depth in Section 5.1.8.

Abstraction in this refutation-based verification framework is built around adversarial example generation. It is the process of mapping a region to a finite set of testable points representing the most unsafe characteristics of the region. Verification then boils down to testing these representative points. An unsafe region is quickly determined when one test point is found to cause misclassification. The unsafe region then waits to be partitioned in subsequent iterations to pinpoint the exact unsafe subregion(s). This refutation-based abstraction is an optimization that reduces the frequency of invoking the expensive verification process by rapidly eliminating unsafe regions through adversarial example generation.

The framework is open source[1] and built to be easily extended to new strategies of abstraction, partitioning and verification as well as new applications by supporting the Tensorflow Protocol Buffer API.

[1] https://github.com/formal-verification-research/ARFramework

Algorithm 5.1: Abstraction Partitioning Algorithm

Data: P: set of unverified regions
Data: S: set of known safe regions
Data: U: set of known unsafe regions with associated counterexample
Data: E: set of found adversarial examples
Input: I: Initial region

$P = \{I\}$; $U = \emptyset$; $E = \emptyset$; $S = \emptyset$;
while $(P \neq \emptyset \vee U \neq \emptyset) \wedge \neg SIGINT$ **do**
  if $P \neq \emptyset$ **then**
    $r \in P$; $P = P - r$;
    if $|r| \leq 0$ **then continue**;
    ver_result, α = verify(r); // α: counter example
    if *ver_result* == *SAFE* **then** $S = S + r$;
    else if *ver_result* == *UNSAFE* **then**
      Partition r into $R = \{r_1, r_2, \ldots, r_n\}$
      $\alpha \in r_x$ where $x \in [1, n]$
      $R = R - r_x$; $P = P + R$;
      $U = U + \{(r_x, \alpha)\}$
      $E = E + \alpha$
    else if *ver_result* == *UNKNOWN* **then**
      Partition r into $R = \{r_1, r_2, \ldots, r_n\}$;
      for $i = 1; i \leq n; i = i + 1$ **do**
        Abstract r_i to a set of testable points $A = \{a_1, a_2, \ldots, a_m\}$;
        for $j = 1; j \leq m; j = j + 1$ **do**
          pointIsSafe = test_point_safety(a_j);
          if *pointIsSafe* **then**
            continue;
          $U = U + \{(r_i, a_j)\}$
          $E = E + a_j$; $R = R - r_i$;
          break;
      $P = P + R$
  else if $U \neq \emptyset$ **then**
    $(r, a) \in U$; $U = U - (r, a)$; // pop element from U
    if $|r| \leq 1$ **then continue**;
    Partition r into $R = \{r_1, r_2, \ldots, r_n\}$
    $a \in r_x$ where $x \in [1, n]$
    $R = R - r_x$; // remove subregion known to be unsafe
    $P = P + R$; $U = U + \{(r_x, a)\}$
return $(|P|, E, U, |U|, S, |S|)$; // Incremental Report

5.1.1 Data Structures

The framework manipulates data in the form of *regions* and *points* and organizes them in four main data structures. P is the set of all unverified regions. U is the set of all regions proven to be unsafe with their corresponding counterexamples as shown in Algorithm 5.1: $(r, \alpha) \in U$, where r is the region and α is the counterexample. S is the set of all regions proven to be safe, i.e., free of adversarial examples. E is the set of all adversarial examples found. The state of these data structures represents the incremental results of the framework as it runs. The framework terminates on two conditions: either it received an interrupt signal or both P and U are empty, meaning that all regions have been verified, all adversarial examples in the initial region have been reported and stored in E, and all safe regions are in S.

5.1.2 Lexicographical Tree Data Structure

Through the use of adversarial example generation algorithms, the framework quickly eliminates the unsafe regions containing these examples. Naive search through a potentially large set of regions for those containing the generated points is $O(mnh)$, where m is the number of dimensions in each point, n is the number of regions, and h is the number of generated points. As m and h remain constant as the size of the problem grows, the algorithmic complexity reduces to $O(n)$. Often performance improvement in a search can be achieved by exploiting principles such as ordering or hashing of elements. A hash table would be an effective optimization due to near constant insert, search, and delete operations, on average, but because the stored element (region) and the search criteria (point) do not always hash to the same value, a hash table cannot be used. Self-balancing binary trees offer efficient implementations of these operations but require an ordering between stored elements. Both regions and points are multidimensional structures. A common method for ordering multidimensional data is through a lexicographical comparison such as alphabetical ordering. When ordered using this predicate in a self-balancing binary tree, the search operation is reduced to $O(log(n))$. Therefore, it is very efficient, even at large scales, for searching and removing the region containing a specific generated adversarial

example. Therefore, even at very large scales, the operation of finding and removing the region containing a specific generated adversarial example is quite inexpensive.

5.1.3 Concurrency

Since no data dependency exists between any unique regions, they can be verified in parallel. By simply serializing the read and write operations on the four data structures P, S, E, and U, this framework can scale to any number of concurrent processes. Due to the synchronization overhead on the data structures, the observed speedup is near linear in the number of threads.

5.1.4 Initialization Parameters

The inputs to the framework include defining the initial hyperrectangular region I, domain range, granularity, and initial activation. The *domain range* is an upper and lower bound on each dimension in the input space. This range enforces that, for the application of the data and network being verified, the search space is limited to values in that domain. In the case of 8-bit images, the domain range for each dimension should be $[0, 255]$. Granularity enforces the discreteness of the input space. When data is converted to a digital format it inherits the precision features of the storage medium. In the case of 8-bit images, the original data may be continuous but the storage medium forces the data into a discrete range. *Granularity* is defined as the distance between two valid values of each dimension in the input space. As almost all implementations of neural networks operate in a digital environment, it is reasonable to assume the existence of a nonzero granularity vector. A *valid point* must lie inside the domain range and obey the granularity defined by the application and the initial activation. The initial activation is the point around which safety is tested and is, by declaration, a valid point. A point is *unsafe* if the classification of the initial activation and that of the point disagree. Once configuration has completed, the initial region I is placed into the set of unverified regions P and the framework begins execution.

5.1.5 Main Procedure

Giving priority to regions in P (unverified), a region r is removed from either P or U. If no valid points exist in r or the only valid point in r is an adversarial example, the region is discarded and execution continues from the top-most loop (line 2). If the region r has not been verified, it is sent to the verification strategy (line 6), discussed in greater detail in Section 5.1.9. The verification strategy returns `unknown`, `unsafe`, or `safe`. In the `safe` case (line 7), r is placed in S because it is a known safe region and execution then returns to the top-most loop. If a region is found to be `unsafe` (line 8), the verification strategy returns a counterexample α. r is then partitioned into a set of regions $R = \{r_1, \ldots, r_n\}$, where n is a parameter defined in the partitioning strategy. The region containing α, denoted in Algorithm 5.1 as r_x (line 10), is removed from R and placed into U, along with α. α is also stored in E as a record of found adversarial examples. Execution then returns to the top-most loop.

The most common outcome is `unknown` (line 14), indicating that the region is too large to be verified and must be partitioned into a set of subregions $R = \{r_1, \ldots, r_n\}$. Each subregion is then abstracted to a set of testable points $A = \{a_1, \ldots, a_m\}$ to be checked for misclassification. On discovering an unsafe abstraction point a_j, its enclosing region r_i is removed from R and placed in U (line 22) along with the counterexample a_j. No action is taken on safe abstraction points. After all subregions in R have been abstracted and tested, all remaining regions are added to P to be verified in subsequent iterations. Any adversarial example found is stored in E. If at any time the program receives a user interrupt, an incremental report is produced based on the state of P, S, E, and U.

5.1.6 Dimension Ranking

In both the abstraction and partitioning steps, a heuristic ranking of dimensions can improve on base performance. We propose the following four methods for performing this operation: random ordering, largest-first, Intellifeatures [31], and gradient-based ordering. Random ordering is used as a control to measure the baseline performance of a dimension ranking partitioning strategy. Largest-first orders dimensions by the magnitude of the dif-

ference between its upper and lower bounds and is valuable in maintaining close relative range magnitudes during partitioning. Intellifeatures, described in detail in Chapter 4, uses knowledge about the classification regions represented in the training dataset to select dimensions that predominantly define the difference between the current point and the next closest distinct classification region. The gradient-based approach uses the magnitude of the gradient of the cost function with respect to the current input to order dimensions. Algorithm 5.2 describes the gradient-based method. These strategies are compared in Section 5.2 by their effect on the performance of both abstraction and partitioning.

Algorithm 5.2: Gradient-based Dimension Ranking Heuristic

Input: r: Region under examination
Input: $g(x, y)$: Gradient of cost function with respect to input x
Input: y: Label/ground truth of desired class of verification region
Input: n: Number of dimensions in r and g
Output: Dims: Ordered set of dimension indices

```
gradient = g(c, y);
t = [];
for j = 1; j ⩽ n; j = j + 1 do
    t.append((|gradient[j]|, j));
Sort t in descending order by the |gradient[j]| element
Dims = [t[1][2], t[2][2], ..., t[n][2]]
```

5.1.7 Abstraction Strategies

The goal of an abstraction strategy is to avoid excessive calls to the expensive verification procedure and to produce refutations to the adversarial robustness property. The ideal abstraction strategy maps a region to a set of covering points that are also likely adversarial examples. Implemented in the framework are three different abstraction strategies based on different assumptions of adversarial examples' characteristics: central point, random point, and RFGSM.

Central Point Abstraction

This abstraction strategy assumes that the central point is the most representative

of the entire region. This strategy performs well under the assumption that adversarial examples commonly exist in clusters.

Random Point Abstraction

Selecting points at random is a naive abstraction approach with the goal of finding adversarial examples. This strategy is often used more as a control for our experiments than a viable method. Under this strategy, points are selected from a uniform distribution over the region.

Randomized Fast Gradient Sign Method

Adversarial example generation algorithms have many of the same traits as the ideal abstraction strategy. Many of these algorithms accurately generate points with high probability of being adversarial examples. Some require intimate knowledge of the network under examination and others treat it as a black box [32]. Jacobian-based saliency map algorithm [5] is iterative in nature and can target certain misclassifications whereas the *fast gradient sign method* (FGSM) [1] is a constant time algorithm without a specified target. Defined by $x_{adv} = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$, FGSM uses the gradient of the cost function of the neural network with respect to the input to determine how to modify that input toward a class change (increase cost function). FGSM has limited output variance when generating multiple adversarial examples from the same base point x. When x is constant across multiple invocations of FGSM, the only source of variance in the output comes from the parameter ϵ. The generated adversarial examples are then only scaled along the direction defined by the sign of the gradient and represent a tiny portion of the abstracted region. Also, when regions are small, relative to the granularity, the number of valid abstractions that FGSM can generate is significantly reduced.

In this work, we propose an improved adversarial example generation algorithm, *randomized FGSM* (RFGSM), where dimensions are selected according to some heuristic to either follow FGSM or vary randomly. This new algorithm increases the variance of points produced by FGSM by modifying dimensions that contribute less to the classification out-

put of the network with random values. It is a promising abstraction strategy due to its high success rate, constant time algorithm, and output variance. Equation 5.1 shows the definition of RFGSM and the steps taken to generate a single adversarial example. x is a point in the input space $\mathbb{R}^n$, $M \in \{0,1\}^n$ is a binary mask used to select dimensions to be manipulated by FGSM or random values in $R \in \{-1,1\}^n$. Dimension ranking heuristics can be used to set M such that dimensions contributing the most to the classification output of the neural network are selected for FGSM.

$$x_{adv} = x + \vec{\epsilon} \odot [\text{sign}(\nabla_x J(\theta, x, y)) \odot M + R \odot (1 - M)] \tag{5.1}$$

The success of RFGSM relies on solving the following major issues: (1) adversarial examples must be valid according to the granularity and domain range of the application; (2) the success rate of FGSM drops as partitioned subregions become smaller; and (3) selecting the dimension ranking heuristic to best maintain success rate while increasing output variance.

FGSM does not produce points that naturally abide by the granularity of the application. RFGSM selects $\vec{\epsilon}$ to be a multiple of the granularity then implements a rounding operation to ensure generated points are valid. A point $\vec{a}$ generated by FGSM is rounded to the nearest valid point $\vec{a_{new}}$ using Equation 5.2. For each dimension i, it first finds the difference between a_i of $\vec{a}$ and v_i of the valid point $\vec{v}$, and then divides this difference by the granularity g_i before rounding to the nearest integer. This rounded value is the quantity of directional steps of the magnitude of the granularity to take from the test point $\vec{a}$ to reach the valid point $\vec{v}$. $\vec{v}$ is the initial test input provided in the framework configuration. Multiplying by the granularity produces the amount with which to modify $\vec{v}$ to get $\vec{a_{new}}$, the closest valid point to $\vec{a}$.

$$\forall i \in [1,n],\ a_{i_{new}} = v_i + \lfloor \frac{(a_i - v_i)}{g_i} \rceil * g_i \tag{5.2}$$

When ranges on dimensions become smaller than the granularity of that dimension,

RFGSM tends to produce adversarial examples that lie outside the region. These counterexamples are still valuable and reported but no longer function as abstractions of that specific region. In these cases, the RFGSM algorithm falls back to another abstraction strategy such as random or central point abstraction. The former is implemented in the proposed framework.

5.1.8 Partitioning Strategies

Extremely aggressive partitioning strategies can rapidly increase the number of unverified regions and, thereby, the memory requirements for running the framework. Implemented in the framework is a dimension ranking partitioning strategy that is not aggressive and attempts to greedily select dimensions to subdivide based on a dimension ranking heuristic. The results of different heuristics on the overall performance of the framework can be seen in Table 5.2.

5.1.9 Verification Strategies

The ideal verification strategy scales in algorithmic complexity with the size of the region and can perform an iterative verification of a partition of a region in less than or equal time than a verification of that region. Many verification strategies for neural networks are tied to the number of dimensions in the input space or number of neurons, but do not scale with size of region. A verification strategy that fits these ideal specifications and has sublinear computational complexity with respect to region size is the main focus of future work for this framework. It is the main focus because, by uninterrupted program termination, the verification strategy will have been called on the entire partition of the original region. In order for full termination to occur in a reasonable amount of time, the verification strategy must be optimized to fit these ideal characteristics.

Discrete Exhaustive Search

To this point, our research has led us to one verification strategy that scales linearly with region size. Discrete exhaustive search, even exploiting data parallelism, is expensive

and only viable on regions containing fewer than 10^6 valid points. As it is the only option matching most of the characteristics of the ideal strategy, the framework contains an implementation of discrete exhaustive search that returns `unknown` when invoked on regions containing more valid points than a threshold ($\approx 10^4$).

5.2 Results

Two tests were conducted to evaluate the performance of the framework using different abstraction and partitioning strategies. Section 5.2.1 shows how well different dimension ranking heuristics maintain success rate of adversarial example generation in RFGSM. Section 5.2.2 details the incremental report of the framework after running for five minutes with different partitioning strategies. Results are generated on a machine with an AMD Ryzen Threadripper 12-Core 3.5 GHz Processor and 132 GB memory, running Ubuntu Linux (v18.04.3). All timed tests were run using 10 threads processing regions concurrently.

5.2.1 RFGSM - Dimension Ranking Evaluation

While all three abstraction strategies mentioned above are implemented in the framework, the tests below show results for RFGSM with a random point fallback. In previous experiments, the central point and random point abstraction strategies performed significantly worse than RFGSM, leading to a focus on variants of RFGSM as more viable abstraction strategies.

Figure 5.1 charts the success rate of RFGSM with different dimension ranking heuristics as the *balance factor* decreases. The balance factor is the ratio of dimensions to be manipulated using FGSM. A balance factor equal to 1 reduces to pure FGSM whereas a balance factor equal to 0 reduces to pure random manipulation. Dimension ranking heuristics that maintain success rate at lower balance factors offer greater output variance and are more desirable. Also, each dataset has an associated *verification radius* which is half the range of each dimension of the verification region. A verification radius of 0.1 means that generated adversarial examples may lie in the following region: $\forall i \in I,\ [i-0.1, i+0.1]$ where I is the value of the initial activation (central point of the verification region) and i

is the value of any dimension in I.

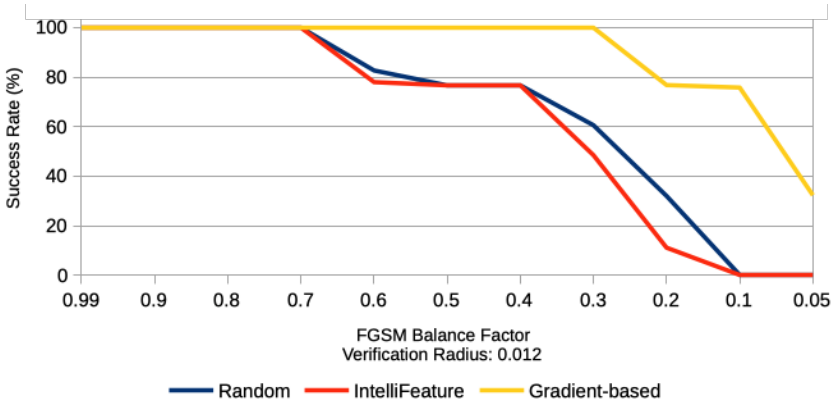
Across the three different datasets, the gradient-based dimension ranking heuristic maintains higher success rate than the other two methods as balance factor decreases. In general, IntelliFeature and random dimension ranking perform similarly. Figure 5.1(c) shows that IntelliFeature maintains relatively high success rate at low balance factors. Because MNIST is simple intensity data, IntelliFeature seems to be able to select dimensions that define the differences between classes easier on this data than on the RGB data (GTSRB and CIFAR-10). Figure 5.1(a) shows an interesting property of the CIFAR-10 neural network and/or dataset. RFGSM is very successful even at low balance factors with an extremely small verification radius. This means that the network under test is not robust to adversarial examples and even small and mostly random manipulations of a known input may result in adversarial examples with high probability.

The gradient-based dimension ranking heuristic demonstrates superiority over the others in maximizing output variance while maintaining success rate. Adversarial examples generated using RFGSM are more representative of the regions they abstract, because they are not limited to merely the direction defined by the gradient. Note that the tests for Figure 5.1 were conducted outside the effects of the rest of the framework to better isolate the performance of the RFGSM abstraction strategy.

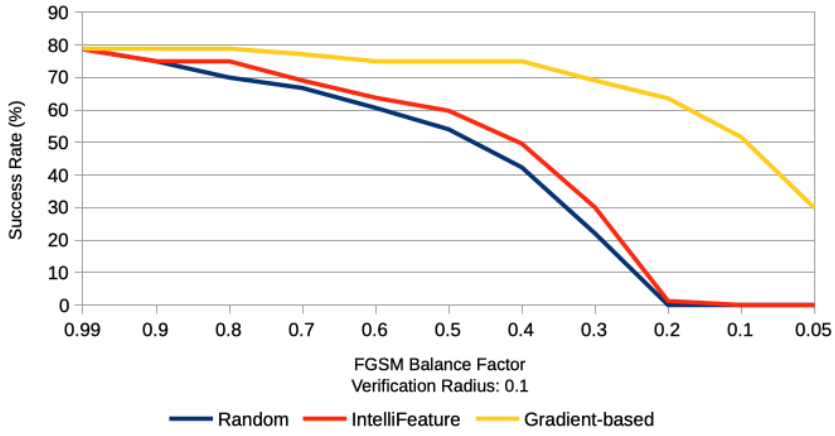
Table 5.1 shows the incremental results of the entire framework when running with these three abstraction strategies for five minutes. RFGSM with the gradient-based dimension ranking heuristic consistently generates the most adversarial examples, maintains a smaller number of unverified regions, and functions well even at low balance factors and verification radii. The largest-first dimension ranking heuristic was not tested as an RFGSM abstraction strategy because the size of the range of a dimension does not have any apparent connection with its contribution to the classification output.

5.2.2 Partitioning Strategy Evaluation

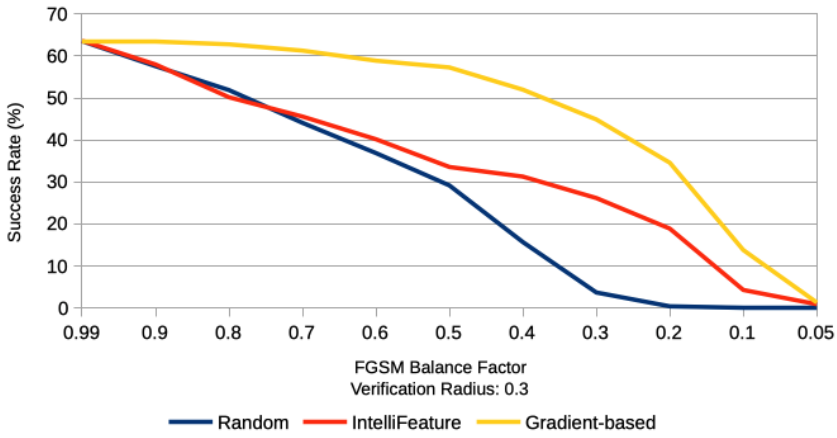
Using the best abstraction strategy from the previous tests (RFGSM and gradient-based), the dimension ranking partitioning strategy is evaluated in Table 5.2. Metrics



(a) CIFAR-10



(b) GTSRB



(c) MNIST

Fig. 5.1: RFGSM success rate with three different dimension ranking heuristics as balance factor decreases.

Table 5.1: Results for using three dimension ranking heuristics for RFGSM.

	MNIST			GTSRB			CIFAR-10		
Balance Factor	0.5			0.4			0.3		
Verification Radius	0.3			0.1			0.012		
	$\|P\|$	$\|U\|$	$\|E\|$	$\|P\|$	$\|U\|$	$\|E\|$	$\|P\|$	$\|U\|$	$\|E\|$
Random	81624	48	70	13824	1249	1366	36679	3	12
Gradient-based	80225	538	1077	1	15579	63958	13545	19240	98989
IntelliFeature	81154	71	117	12085	2604	2969	36182	2	5

indicating good partitioning strategies are increased numbers of adversarial examples found ($|E|$) and relatively low numbers of unverified regions ($|P|$). If $|P|$ is large, this is an indicator of a partitioning strategy that is too aggressive and will quickly exhaust the memory resources of the computing system. If $|E|$ is small, the partitioning strategy may be too greedy where the same dimensions are subdivided over multiple iterations, causing RFGSM to become ineffective and fallback to random point abstraction with a much lower success rate.

Table 5.2 shows that the largest first partitioning strategy performs best after running for five minutes with a balance factor of 0.7 and a verification radius of 0.4. It ensures that all the largest dimensions are subdivided first, thereby balancing the abstraction method and avoiding fallback less frequently. This contributes to the high number of adversarial examples and low number of unverified regions. Random ranking performs well for this same reason but may repeat the same dimensions for subdivision, explaining its lesser performance. The other two methods commonly repeat dimensions enabling the fallback abstraction strategy almost immediately. During the test of gradient-based and IntelliFeature, all the adversarial examples found are generated in the first few seconds of execution. The best algorithm pair for partitioning and abstraction demonstrated by the test results is RFGSM with the gradient-based heuristic and dimension ranking partitioning algorithm with the largest first strategy.

5.3 Conclusion

A dimension reduction algorithm is equivalent to a dimension ranking algorithm with

Table 5.2: Four dimension ranking heuristics for partitioning results.

	MNIST			GTSRB			CIFAR-10		
	$\|P\|$	$\|U\|$	$\|E\|$	$\|P\|$	$\|U\|$	$\|E\|$	$\|P\|$	$\|U\|$	$\|E\|$
Random	87823	114	481	0	13768	98568	7710	16858	223105
Gradient-based	842	5	20	312	86	932	382	195	7394
IntelliFeature	474	4	16	284	341	3623	61	349	7145
Largest First	41502	6982	12160	2	14756	113822	0	32638	251722

a truncation operation. In the experiments detailed in this chapter, dimension reduction is generalized to dimension ranking. Several methods for ranking dimensions based on a heuristic estimating a dimension's contribution to the classification output are tested to see how the success rate of adversarial example generation is affected by increasing levels of random noise. The best performing method is a novel modification of FGSM [1], called RFGSM, that orders dimensions by the magnitude of the gradient of the loss function of the neural network with respect to the input. RFGSM with the gradient-based dimension ranking heuristic enables a constant time adversarial example generation algorithm to produce more examples with significantly higher variance for a single input while maintaining its high success rate. These properties qualify RFGSM to act as a successful refutation-based abstraction strategy in the novel verification framework.

Due to the observed prevalence of adversarial examples in these experiments and those of [1,4,5], refutation-based verification shows promise as an optimization for more robust and expensive verification algorithms. This chapter describes and evaluates a novel refutation-based abstraction and partitioning verification framework that utilizes adversarial example generation algorithms to *abstract* regions and subregions in the input space to a small set of testable points with high probability of being adversarial examples. These refutation-based abstractions can be verified extremely quickly and often result in avoiding a call to the more expensive and complete formal verification strategy housed by the framework. Several abstraction techniques are analyzed for their ability to cover the region they abstract, thereby acting as a more representative abstraction, and produce adversarial examples. The methods that perform the best are integrated into an implementation of the framework and

results are reported on three classification neural networks trained on well-known benchmark datasets in the field of artificial intelligence.

CHAPTER 6

CONCLUSION

Given their demonstrated weakness to adversarial inputs, great potential for modeling difficult-to-formalize problems, and growing use in safety-critical applications, the motivation to provide formal guarantees of safety properties of neural networks is self-evident. Chapters 4 and 5 describe a series of hypotheses and experiments investigating the local adversarial robustness of neural networks. First is an exploration of the effect of a novel dimension reduction heuristic on the effectiveness of a state-of-the-art search method. Concurrently, the issue of dimension ranking based on heuristics estimating the magnitude of the contribution of a dimension to a classification output is analyzed.

Based on the results of these first experiments and guided by the findings of leaders in this field, a refutation-based abstraction verification framework is presented that leverages adversarial example generation, partitioning, parallelism, and specialized data structures to house formal verification techniques and enhance them with these optimizations. Each of the pieces in this framework are analyzed with a focus on techniques to perform highly representative refutation-based abstraction. Multiple novel techniques are pitted against each other in tests that reveal their strengths and constraints which ultimately contribute to an analysis of the overall performance of the framework consisting of the best abstraction, partitioning, and verification strategies. The main findings and contributions of this work are as follows:

- Analysis of the effect of a novel dimension reduction technique on the efficiency of a state-of-the-art search method
- Introduction of a refutation-based abstraction technique that maintains the success rate and algorithmic complexity of an advanced adversarial example generation algorithm while increasing its variance through dimension ranking heuristics

- Definition of a generic framework and associated structures and algorithms that augment formal verification techniques with refutation-based abstraction, input partitioning, and parallelism
- Results of the proposed techniques when tested on standard classification benchmarks and compared to contemporary methods
- Open source implementations of all the detailed algorithms with documentation describing their utility, function, and extensibility

6.1 IntelliFeature

The specification-guided dimension reduction technique, IntelliFeature, showed improvement over DLV [2] in the metrics of run time and adversarial example confidence. The tests in Chapter 4 led to the conclusions that specification-guided dimension reduction can improve the efficiency of a search algorithm and report adversarial examples to which a neural network is more susceptible. The open source IntelliFeature tool is available at https://github.com/formal-verification-research/DLV_intellifeatures. The observed prevalence of adversarial examples from these tests also led to the conclusion that adversarial example generation could be used as a refutation-based optimization for formal verification.

6.2 Refutation-based Abstraction Framework

Chapter 5 introduced a novel approach of using adversarial example generation as a refutation-based region abstraction strategy. RFGSM with the proposed gradient-based dimension ranking heuristic maintains the success rate of its predecessor, FGSM, while significantly increasing the variance of produced adversarial inputs. The framework, even when housing a relatively naive verification strategy, produces a large quantity of adversarial examples that, because of the input partitioning, are well-distributed through the region under question. A case is made for the utility of generating many adversarial examples versus the traditional approach of reporting the first found counterexample. Evaluation

is performed on standard datasets and neural network architectures in the artificial intelligence community. The entire implementation of the refutation-based framework along with all the abstraction and partitioning strategies is open source and can be found at https://github.com/formal-verification-research/ARFramework.

6.3 Future Work

As mentioned in Chapter 5, the fallback formal verification strategy for the refutation-based abstraction framework is the main bottleneck in the algorithm. Discrete exhaustive search is currently used because it scales in algorithmic complexity linearly with respect to the region size. This method is very expensive and quickly becomes intractable. A key direction of future work is an investigation into verification strategies, more efficient than discrete exhaustive search, that scale with region size. This ideal verification strategy is the missing piece to the puzzle that will greatly improve the performance of the framework proposed in this work.

A key goal of the abstraction framework is to generate as many adversarial examples as possible and that those adversarial examples be a covering set of the robustness region. Goodfellow et. al. show that neural networks can have increased robustness to these adversarial inputs when trained on them [1]. An issue arises that no formal guarantees are currently provided by retraining on adversarial examples. Answering the question, "Does there exist an iterative retraining algorithm with convergence guarantees for the adversarial robustness property?" is a key focus of future work that would give true significance to the output of the refutation-based abstraction framework.

More efficient and reliable refutation-based abstraction techniques would also significantly improve the performance of the framework. RFGSM has shown great promise in increasing variance while maintaining accuracy but it is limited by a maximum observed success rate of about 80%. Future work should focus on improving abstraction techniques. Also, abstraction methods that more accurately model the properties of the region would be very valuable. Currently, adversarial example generation can, at best, only model the refutation of the robustness property. An abstraction that is significantly easier to verify

and that, when verified and found safe, implies that the original region is also safe would be a huge leap forward.

REFERENCES

[1] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[2] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, 2017, pp. 3–29. [Online]. Available: https://doi.org/10.1007/978-3-319-63387-9_1

[3] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, 2017, pp. 97–117. [Online]. Available: https://doi.org/10.1007/978-3-319-63387-9_5

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[5] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 372–387. [Online]. Available: https://doi.org/10.1109/EuroSP.2016.36

[6] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *CoRR*, vol. abs/1607.02533, 2016. [Online]. Available: http://arxiv.org/abs/1607.02533

[7] S. A. Seshia and D. Sadigh, "Towards verified artificial intelligence," *CoRR*, vol. abs/1606.08514, 2016. [Online]. Available: http://arxiv.org/abs/1606.08514

[8] D. Kroening and O. Strichman, *Decision Procedures: An Algorithmic Point of View*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[9] G. B. Dantzig, A. Orden, and P. Wolfe, "The generalized simplex method for minimizing a linear form under linear inequality restraints." *Pacific J. Math.*, vol. 5, no. 2, pp. 183–195, 1955. [Online]. Available: https://projecteuclid.org:443/euclid.pjm/1103044531

[10] G. Katz, "Verification of machine learning programs," 30 Jul 2018. [Online]. Available: https://www.youtube.com/watch?v=Reo5REo71GU

[11] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *CoRR*, vol. abs/1606.06565, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[12] S. Zheng, Y. Song, T. Leung, and I. J. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 4480–4488. [Online]. Available: https://doi.org/10.1109/CVPR.2016.485

[13] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[14] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *CoRR*, vol. abs/1611.01236, 2016. [Online]. Available: http://arxiv.org/abs/1611.01236

[15] F. Tramèr, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=rkZvSe-RZ

[16] C. Liu, T. Arnon, C. Lazarus, C. W. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *CoRR*, vol. abs/1903.06758, 2019. [Online]. Available: http://arxiv.org/abs/1903.06758

[17] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, 2018, pp. 408–426. [Online]. Available: https://doi.org/10.1007/978-3-319-89960-2_22

[18] G. B. Dantzig, 1946.

[19] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett, "The marabou framework for verification and analysis of deep neural networks," in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds. Cham: Springer International Publishing, 2019, pp. 443–452.

[20] L. de Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340. [Online]. Available: https://github.com/Z3Prover/z3

[21] W. Burger and M. J. Burge, *Scale-Invariant Feature Transform (SIFT)*. London: Springer London, 2016, pp. 609–664. [Online]. Available: https://doi.org/10.1007/978-1-4471-6684-9_25

[22] V. Tjeng and R. Tedrake, "Verifying neural networks with mixed integer programming," *CoRR*, vol. abs/1711.07356, 2017. [Online]. Available: http://arxiv.org/abs/1711.07356

[23] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *CoRR*, vol. abs/1706.07351, 2017. [Online]. Available: http://arxiv.org/abs/1706.07351

[24] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE, 2018, pp. 3–18. [Online]. Available: https://doi.org/10.1109/SP.2018.00058

[25] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev, "An abstract domain for certifying neural networks," *PACMPL*, vol. 3, no. POPL, pp. 41:1–41:30, 2019. [Online]. Available: https://dl.acm.org/citation.cfm?id=3290354

[26] W. Ruan, X. Huang, and M. Kwiatkowska, "Reachability analysis of deep neural networks with provable guarantees," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, 2018, pp. 2651–2659. [Online]. Available: https://doi.org/10.24963/ijcai.2018/368

[27] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1599–1614. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi

[28] L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for ReLU networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5276–5285. [Online]. Available: http://proceedings.mlr.press/v80/weng18a.html

[29] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BkUHlMZ0b

[30] A. Banks and R. Ashmore, "Requirements assurance in machine learning," in *Proceedings of the AAAI Workshop on Artificial Intelligence Safety (SafeAI2019)*, 2019.

[31] J. Smith, X. Huang, V. Swaminathan, and Z. Zhang, "Improving deep neural network verification using specification-guided search," in *2nd Workshop on Formal Methods for ML-Enabled Autonomous Systems*, 2019.

[32] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *CoRR*, vol. abs/1602.02697, 2016. [Online]. Available: http://arxiv.org/abs/1602.02697