Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*

Guy Katz, Clark Barrett, David Dill, Kyle Julian and Mykel Kochenderfer

Stanford University, USA
{guyk, clarkbarrett, dill, kjulian3, mykel}@stanford.edu

Abstract. Deep neural networks have emerged as a widely used and effective means for tackling complex, real-world problems. However, a major obstacle in applying them to safety-critical systems is the great difficulty in providing formal guarantees about their behavior. We present a novel, scalable, and efficient technique for verifying properties of deep neural networks (or providing counter-examples). The technique is based on the simplex method, extended to handle the non-convex *Rectified Linear Unit* (*ReLU*) activation function, which is a crucial ingredient in many modern neural networks. The verification procedure tackles neural networks as a whole, without making any simplifying assumptions. We evaluated our technique on a prototype deep neural network implementation of the next-generation airborne collision avoidance system for unmanned aircraft (ACAS Xu). Results show that our technique can successfully prove properties of networks that are an order of magnitude larger than the largest networks verified using existing methods.

1 Introduction

神经网络越来越流行

Artificial neural networks [7, 31] have emerged as a promising approach for creating scalable and robust systems. Applications include speech recognition [9], image classification [22], game playing [32], and many others. It is now clear that software that may be extremely difficult for humans to implement can instead be created by training *deep neural networks* (*DNN*s), and that the performance of these DNNs is often comparable to, or even surpasses, the performance of manually crafted software. DNNs are becoming widespread, and this trend is likely to continue and intensify.

神经网络的安全遭到质疑，且很难手工验证

Great effort is now being put into using DNNs as controllers for safety-critical systems such as autonomous vehicles [4] and airborne collision avoidance systems for unmanned aircraft (ACAS Xu) [13]. DNNs are trained over a finite set of inputs and outputs and are expected to *generalize*, i.e. to behave correctly for previously-unseen inputs. However, it has been observed that DNNs can react in unexpected and incorrect ways to even slight perturbations of their inputs [33]. This unexpected behavior of DNNs is likely to result in unsafe systems, or restrict the usage of DNNs in safety-critical applications. Hence, there is an urgent

* This is the extended version of a paper with the same title that appeared at CAV 2017.

need for methods that can provide formal guarantees about DNN behavior. Unfortunately, manual reasoning about large DNNs is impossible, as their structure renders them incomprehensible to humans. Automatic verification techniques are thus sorely needed, but here, the state of the art is a severely limiting factor.

Verifying DNNs is a difficult problem. DNNs are large, non-linear, and non-convex, and verifying even simple properties about them is an NP-complete problem (see Section I of the appendix). DNN verification is experimentally beyond the reach of general-purpose tools such as *linear programming* (*LP*) solvers or existing *satisfiability modulo theories* (*SMT*) solvers [3, 10, 30], and thus far, 专用 dedicated tools have only been able to handle very small networks (e.g. a single hidden layer with only 10 to 20 hidden nodes [29, 30]).

The difficulty in proving properties about DNNs is caused by the presence of *activation functions*. A DNN is comprised of a set of layers of nodes, and the value of each node is determined by computing a linear combination of values from nodes in the preceding layer and then applying an activation function to the result. These activation functions are non-linear and render the problem non-convex. We focus here on DNNs with a specific kind of activation function, called a *Rectified Linear Unit* (*ReLU*) [26]. When the ReLU function is applied to a node with a positive value, it returns the value unchanged (the *active* case), but when the value is negative, the ReLU function returns 0 (the *inactive* case). ReLUs are very widely used [22, 24], and it has been suggested that their piecewise linearity allows DNNs to generalize well to previously unseen inputs [6, 7, 11, 26]. Past efforts at verifying properties of DNNs with ReLUs have had to make significant simplifying assumptions [3, 10] — for instance, by considering only small input regions in which all ReLUs are fixed at either the active or inactive state [3], hence making the problem convex but at the cost of being able to verify only an approximation of the desired property.

We propose a novel, scalable, and efficient algorithm for verifying properties of DNNs with ReLUs. We address the issue of the activation functions head-on, by extending the simplex algorithm — a standard algorithm for solving LP instances — to support ReLU constraints. This is achieved by leveraging the piecewise linear nature of ReLUs and attempting to gradually satisfy the constraints that they impose as the algorithm searches for a feasible solution. We call the algorithm *Reluplex*, for "ReLU with Simplex".

The problem's NP-completeness means that we must expect the worst-case performance of the algorithm to be poor. However, as is often the case with SAT and SMT solvers, the performance in practice can be quite reasonable; in particular, our experiments show that during the search for a solution, many of the ReLUs can be ignored or even discarded altogether, reducing the search space by an order of magnitude or more. Occasionally, Reluplex will still need to *split* on a specific ReLU constraint — i.e., guess that it is either active or inactive, and possibly backtrack later if the choice leads to a contradiction.

We evaluated Reluplex on a family of 45 real-world DNNs, developed as an early prototype for the next-generation airborne collision avoidance system for unmanned aircraft ACAS Xu [13]. These fully connected DNNs have 8 layers and 300 ReLU nodes each, and are intended to be run onboard aircraft. They take in sensor data indicating the speed and present course of the aircraft (the *ownship*) and that of any nearby intruder aircraft, and issue appropriate navigation advisories. These advisories indicate whether the aircraft is clear-of-conflict, in which case the present course can be maintained, or whether it should turn to avoid collision. We successfully proved several properties of these networks, e.g. that a clear-of-conflict advisory will always be issued if the intruder is sufficiently far away or that it will never be issued if the intruder is sufficiently close and on a collision course with the ownship. Additionally, we were able to prove certain *robustness* properties [3] of the networks, meaning that small adversarial perturbations do not change the advisories produced for certain inputs.

Our contributions can be summarized as follows. We (i) present Reluplex, an SMT solver for a theory of linear real arithmetic with ReLU constraints; (ii) show how DNNs and properties of interest can be encoded as inputs to Reluplex; (iii) discuss several implementation details that are crucial to performance and scalability, such as the use of floating-point arithmetic, bound derivation for ReLU variables, and conflict analysis; and (iv) conduct a thorough evaluation on the DNN implementation of the prototype ACAS Xu system, demonstrating the ability of Reluplex to scale to DNNs that are an order of magnitude larger than those that can be analyzed using existing techniques.

本文的其余部分安排如下。 我们从第2节中的DNN，SMT和单纯形的背景知识入手。第3节中介绍了抽象Reluplex算法，第4节中重点介绍了关键的实现细节。 然后，我们介绍了ACAS Xu系统及其原型DNN实现。 作为第5节中的案例研究，然后是第6节中的实验结果。第7节中讨论了相关工作，第8节中我们作了总结。

The rest of the paper is organized as follows. We begin with some background on DNNs, SMT, and simplex in Section 2. The abstract Reluplex algorithm is described in Section 3, with key implementation details highlighted in Section 4. We then describe the ACAS Xu system and its prototype DNN implementation that we used as a case-study in Section 5, followed by experimental results in Section 6. Related work is discussed in Section 7, and we conclude in Section 8.

2 Background

神经网络。 深度神经网络（DNN）由一个输入层，一个输出层以及介于两者之间的多个隐藏层组成。 一层由多个节点组成，每个节点都使用一组预定的权重连接到上一层的节点（请参见图1）。 权重选择至关重要，并且要在训练阶段进行（有关概述，请参见例如[7]）。 通过为输入分配值，然后将其通过网络转发，可以从上一层的值计算出每一层的值，最后得到输出的值。

Neural Networks. Deep neural networks (DNNs) are comprised of an input layer, an output layer, and multiple hidden layers in between. A layer is comprised of multiple nodes, each connected to nodes from the preceding layer using a predetermined set of weights (see Fig. 1). Weight selection is crucial, and is performed during a *training* phase (see, e.g., [7] for an overview). By assigning values to inputs and then feeding them forward through the network, values for each layer can be computed from the values of the previous layer, finally resulting in values for the outputs.

通过计算来自上一层的节点值的线性组合，然后应用非线性激活函数[7]，可以确定网络中每个隐藏节点的值。 在这里，我们关注整流线性单元（ReLU）激活函数[26]。 当将ReLU激活函数应用于某个节点时，该节点的值将计算为上一层与0之间的线性组合的最大值。因此，我们可以将ReLU视为函数ReLU（x）= max（0，x ）。

The value of each hidden node in the network is determined by calculating a linear combination of node values from the previous layer, and then applying a non-linear *activation function* [7]. Here, we focus on the Rectified Linear Unit (ReLU) activation function [26]. When a ReLU activation function is applied to a node, that node's value is calculated as the maximum of the linear combination of nodes from the previous layer and 0. We can thus regard ReLUs as the function $\text{ReLU}(x) = \max(0, x)$.

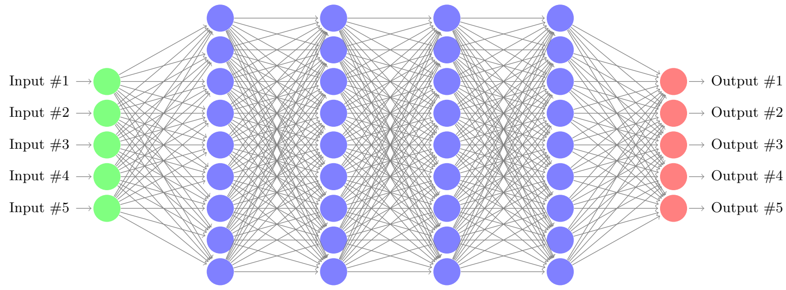


Fig. 1: A fully connected DNN with 5 input nodes (in green), 5 output nodes (in red), and 4 hidden layers containing a total of 36 hidden nodes (in blue).

Formally, for a DNN N, we use n to denote the number of layers and s_i to denote the size of layer i (i.e., the number of its nodes). Layer 1 is the input layer, layer n is the output layer, and layers $2, \ldots, n-1$ are the hidden layers. The value of the j-th node of layer i is denoted $v_{i,j}$ and the column vector $[v_{i,1}, \ldots, v_{i,s_i}]^T$ is denoted V_i. Evaluating N entails calculating V_n for a given assignment V_1 of the input layer. This is performed by 传播 propagating the input values through the network using predefined weights and biases, and applying the activation functions — ReLUs, in our case. Each layer $2 \leq i \leq n$ has a weight matrix W_i of size $s_i \times s_{i-1}$ and a bias vector B_i of size s_i, and its values are given by $V_i = \text{ReLU}(W_i V_{i-1} + B_i)$, with the ReLU function being applied element-wise. This rule is applied repeatedly for each layer until V_n is calculated. When the weight matrices $W_1, \ldots W_n$ do not have any zero entries, the network is said to be *fully connected* (see Fig. 1 for an illustration).

评估DNN N意味着用一组给定的输入分配V1来计算输出Vn.这是通过使用预定义的权重和偏差通过网络传播输入值并应用激活函数（在我们的情况下为ReLU）来执行的。

Fig. 2 depicts a small network that we will use as a running example. The network has one input node, one output node and a single hidden layer with two nodes. The bias vectors are set to 0 and are ignored, and the weights are shown for each edge. The ReLU function is applied to each of the hidden nodes. It is possible to show that, due to the effect of the ReLUs, the network's output is always identical to its input: $v_{31} \equiv v_{11}$.

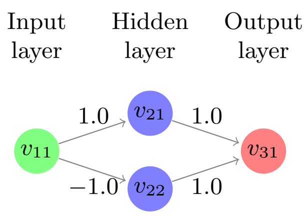


Fig. 2: A small neural network.

这里解释一下：如果一个Σ-formula pi被一些I中的解释满足或没有I中的解释满足，则我们说是T-satisfiable的或T-unsatisfiable的。

Satisfiability Modulo Theories. We present our algorithm as a theory solver in the context of satisfiability modulo theories (SMT).[1] A *theory* is a pair $T = (\Sigma, \mathbf{I})$ where Σ is a signature and $\mathbf{I}$ is a class of Σ-interpretations, the *models* of T, that is closed under variable reassignment. A Σ-formula φ is T-*satisfiable* (resp., T-*unsatisfiable*) if it is satisfied by some (resp., no) interpretation in $\mathbf{I}$. In this paper, we consider only *quantifier-free* formulas. The SMT problem is the problem of determining the T-satisfiability of a formula for a given theory T.

Given a theory T with signature Σ, the DPLL(T) architecture [27] provides a generic approach for determining the T-satisfiability of Σ-formulas. In DPLL(T), a Boolean satisfiability (SAT) engine operates on a Boolean abstraction of the formula, performing Boolean propagation, case-splitting, and Boolean conflict resolution. The SAT engine is coupled with a dedicated *theory solver*, which checks the T-satisfiability of the decisions made by the SAT engine. *Splitting-on-demand* [1] extends DPLL(T) by allowing theory solvers to delegate case-splitting to the SAT engine in a generic and modular way. In Section 3, we present our algorithm as a deductive calculus (with splitting rules) operating on conjunctions of literals. The DPLL(T) and splitting-on-demand mechanisms can then be used to obtain a full decision procedure for arbitrary formulas.

线性实数和单纯形

在DNN的上下文中，一个特别相关的理论是实数运算的理论，我们将其称为TR。TR由包含所有有理数常数的符号和符号{+，?，·，≤，≥}组成，并与实数的标准模型配对。 我们关注线性公式：TR上的公式具有以下附加限制：乘法符号·仅在其操作数中的至少一个是有理常数时才能出现。对于B来说，线性原子总是可以写成A的形式，其中X是一组变量的集合，而ci和d是有理常数。

Linear Real Arithmetic and Simplex. In the context of DNNs, a particularly relevant theory is that of real arithmetic, which we denote as $\mathcal{T}_{\mathbb{R}}$. $\mathcal{T}_{\mathbb{R}}$ consists of the signature containing all rational number constants and the symbols $\{+, -, \cdot, \leq, \geq\}$, paired with the standard model of the real numbers. We focus on *linear* formulas: formulas over $\mathcal{T}_{\mathbb{R}}$ with the additional restriction that the multiplication symbol $\cdot$ can only appear if at least one of its operands is a rational constant. Linear atoms can always be rewritten into the form $\sum_{x_i \in \mathcal{X}} c_i x_i \bowtie d$, for $\bowtie \in \{=, \leq, \geq\}$, where $\mathcal{X}$ is a set of variables and c_i, d are rational constants.

单纯形法[5]是用于确定线性原子连接的可满足性的标准且高效的决策程序。我们的算法扩展了单纯形，因此，我们从原始算法的抽象演算入手（更详尽的描述，请参见[34]）。 演算规则在我们称为配置的数据结构上运行。对于给定的变量集X={x1,...,xn}，单纯形配置是专有符号{SAT，UNSAT}之一或元组<B,T,l,u,a>。初始构型是从输入原子的合体得出的，如下所示：对于每个原子atom，引入一个新的基本变量b，将b的等式添加到tableau中，并添加d作为b的边界（上限，下限 或两者，取决于符号）。 所有变量的初始分配均设置为0，以确保所有表格方程式均成立（尽管可能会违反变量界限）。

The simplex method [5] is a standard and highly efficient decision procedure for determining the $\mathcal{T}_{\mathbb{R}}$-satisfiability of conjunctions of linear atoms.[2] Our algorithm extends simplex, and so we begin with an abstract calculus for the original algorithm (for a more thorough description see, e.g., [34]). The rules of the calculus operate over data structures we call *configurations*. For a given set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$, a simplex configuration is either one of the distinguished symbols $\{\mathtt{SAT}, \mathtt{UNSAT}\}$ or a tuple $\langle \mathcal{B}, T, l, u, \alpha \rangle$, where: $\mathcal{B} \subseteq \mathcal{X}$ is a set of basic variables; T, the *tableau*, contains for each $x_i \in \mathcal{B}$ an equation $x_i = \sum_{x_j \notin \mathcal{B}} c_j x_j$; l, u are mappings that assign each variable $x \in \mathcal{X}$ a lower and an upper bound, respectively; and α, the *assignment*, maps each variable $x \in \mathcal{X}$ to a real value. The initial configuration (and in particular the initial tableau T_0) is derived from a conjunction of input atoms as follows: for each atom $\sum_{x_i \in \mathcal{X}} c_i x_i \bowtie d$, a new basic variable b is introduced, the equation $b = \sum_{x_i \in \mathcal{X}} c_i x_i$ is added to the

与SMT的大多数处理方法一致，我们假定具有相等性的多种一阶逻辑是我们的基本形式主义。

[1] Consistent with most treatments of SMT, we assume many-sorted first-order logic with equality as our underlying formalism (see, e.g., [2] for details).

[2] There exist SMT-friendly extensions of simplex (see e.g. [16]) which can handle $\mathcal{T}_{\mathbb{R}}$-satisfiability of arbitrary literals, including strict inequalities and disequalities, but we omit these extensions here for simplicity (and without loss of generality).

$$\textsf{Pivot}_1 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) < l(x_i), \quad x_j \in \text{slack}^+(x_i)}{T := \mathit{pivot}(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\textsf{Pivot}_2 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) > u(x_i), \quad x_j \in \text{slack}^-(x_i)}{T := \mathit{pivot}(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\textsf{Update} \quad \frac{x_j \notin \mathcal{B}, \quad \alpha(x_j) < l(x_j) \vee \alpha(x_j) > u(x_j), \quad l(x_j) \le \alpha(x_j) + \delta \le u(x_j)}{\alpha := \mathit{update}(\alpha, x_j, \delta)}$$

$$\textsf{Failure} \quad \frac{x_i \in \mathcal{B}, \quad (\alpha(x_i) < l(x_i) \ \wedge \ \text{slack}^+(x_i) = \emptyset) \vee (\alpha(x_i) > u(x_i) \ \wedge \ \text{slack}^-(x_i) = \emptyset)}{\texttt{UNSAT}}$$

$$\textsf{Success} \quad \frac{\forall x_i \in \mathcal{X}.\ l(x_i) \le \alpha(x_i) \le u(x_i)}{\texttt{SAT}}$$

Fig. 3: Derivation rules for the abstract simplex algorithm.

tableau, and d is added as a bound for b (either upper, lower, or both, depending on $\bowtie$). The initial assignment is set to 0 for all variables, ensuring that all tableau equations hold (though variable bounds may be violated).

可以将表格T视为一个矩阵，每个基本变量表示为非基本变量的线性组合。T的行对应于B中的基变量，其列对应于非基变量。对于B中的变量xi和不在B中的变量xj，我们用Ti,j表示方程xi中xj的系数cj。可以通过以下方式更改表格：pivot基变量xi（离开变量）与非基变量xj（进入变量），其Ti,j不等于0。pivot(T,i,j)函数返回一个新的表格，其中等式xi被等式xj代替，并且在每个其他等式中xj的每次出现都被新等式的右侧代替（所得表达式也被标准化为保留表格形式）。变量分配α通过应用于非基本变量的更新操作进行更改：对于xj不∈B，update(α,xj,δ)操作返回与α相同的更新分配α'，除了α'(xj)，对于每个xi∈B，我们都有α'。为了简化以后的演示，我们也表示：

The tableau T can be regarded as a matrix expressing each of the basic variables (variables in $\mathcal{B}$) as a linear combination of non-basic variables (variables in $\mathcal{X} \setminus \mathcal{B}$). The rows of T correspond to the variables in $\mathcal{B}$ and its columns to those of $\mathcal{X} \setminus \mathcal{B}$. For $x_i \in \mathcal{B}$ and $x_j \notin \mathcal{B}$ we denote by $T_{i,j}$ the coefficient c_j of x_j in the equation $x_i = \sum_{x_j \notin \mathcal{B}} c_j x_j$. The tableau is changed via pivoting: the switching of a basic variable x_i (the *leaving* variable) with a non-basic variable x_j (the *entering* variable) for which $T_{i,j} \neq 0$. A $\mathit{pivot}(T, i, j)$ operation returns a new tableau in which the equation $x_i = \sum_{x_k \notin \mathcal{B}} c_k x_k$ has been replaced by the equation $x_j = \frac{x_i}{c_j} - \sum_{x_k \notin \mathcal{B}, k \neq j} \frac{c_k}{c_j} x_k$, and in which every occurrence of x_j in each of the other equations has been replaced by the right-hand side of the new equation (the resulting expressions are also normalized to retain the tableau form). The variable assignment α is changed via *update* operations that are applied to non-basic variables: for $x_j \notin \mathcal{B}$, an $\mathit{update}(\alpha, x_j, \delta)$ operation returns an updated assignment α' identical to α, except that $\alpha'(x_j) = \alpha(x_j) + \delta$ and for every $x_i \in \mathcal{B}$, we have $\alpha'(x_i) = \alpha(x_i) + \delta \cdot T_{i,j}$. To simplify later presentation we also denote:

$$\text{slack}^+(x_i) = \{x_j \notin \mathcal{B} \mid (T_{i,j} > 0 \wedge \alpha(x_j) < u(x_j)) \vee (T_{i,j} < 0 \wedge \alpha(x_j) > l(x_j))$$
$$\text{slack}^-(x_i) = \{x_j \notin \mathcal{B} \mid (T_{i,j} < 0 \wedge \alpha(x_j) < u(x_j)) \vee (T_{i,j} > 0 \wedge \alpha(x_j) > l(x_j))$$

在图3中以有保护的分配形式提供了单纯形演算的规则。如果规则的所有前提都适用于S，则该规则适用于配置S。规则的结论描述了如何更改S的每个组成部分（如果有的话）。当S0是对S应用规则的结果时，我们说S推导S0。 每个Si得出Si+1的配置Si序列称为推导。

The rules of the simplex calculus are provided in Fig. 3 in *guarded assignment form*. A rule applies to a configuration S if all of the rule's premises hold for S. A rule's conclusion describes how each component of S is changed, if at all. When S' is the result of applying a rule to S, we say that S derives S'. A sequence of configurations S_i where each S_i derives S_{i+1} is called a *derivation*.

更新规则（具有适当的δ值）用于强制非基本变量满足其范围。基本变量不能直接更新。 相反，如果基本变量xi太小或太大，则分别应用Pivot1或Pivot2规则，以使用非基本变量xj对其进行旋转。

The Update rule (with appropriate values of δ) is used to enforce that non-basic variables satisfy their bounds. Basic variables cannot be directly updated. Instead, if a basic variable x_i is too small or too great, either the $\textsf{Pivot}_1$ or the $\textsf{Pivot}_2$ rule is applied, respectively, to pivot it with a non-basic variable x_j. This

这使xi成为非基变量，因此可以使用Update规则来调整其分配。pivot仅被允许当xj提供slack时，也就是说，可以调整xj的assignment以使xi更接近它的边界而不会违反其自身的边界。当然，一旦发生pivot并且使用Update规则将xi带入其边界内，其他变量（例如现在的基变量xj）可能被发送在它们的边界外，在这种情况下，必须在以后的迭代中对其进行更正。如果基变量超出边界，但没有任何非基变量可以对其slack，那么将应用Failure规则，并且该问题是不可满足的。因为tableau仅可通过缩放和添加行来改变，所以满足其方程式的assignment的集合始终保持与T0一致。同样，update操作保证了α继续满足T的等式。因此，如果所有变量都在边界之内，则可以应用Success规则，表明α构成了一个原始问题的满足分配。

makes x_i non-basic so that its assignment can be adjusted using the Update rule. Pivoting is only allowed when x_j affords *slack*, that is, the assignment for x_j can be adjusted to bring x_i closer to its bound without violating its own bound. Of course, once pivoting occurs and the Update rule is used to bring x_i within its bounds, other variables (such as the now basic x_j) may be sent outside their bounds, in which case they must be corrected in a later iteration. If a basic variable is out of bounds, but none of the non-basic variables affords it any slack, then the Failure rule applies and the problem is unsatisfiable. Because the tableau is only changed by scaling and adding rows, the set of variable assignments that satisfy its equations is always kept identical to that of T_0. Also, the *update* operation guarantees that α continues to satisfy the equations of T. Thus, if all variables are within bounds then the Success rule can be applied, indicating that α constitutes a satisfying assignment for the original problem.

众所周知，单纯形演算是合理的（即，如果派生以SAT或UNSAT结尾，则原始问题分别是可以满足或不满足的）并且是完整的（始终存在从任何起始conf到SAT或UNSAT的结束派生）。如果在应用过渡规则时使用某些策略，尤其是在存在多个选项时选择离开和输入变量，则可以保证终止。还已知变量选择策略会对性能产生巨大影响。 我们注意到，上述单纯形的版本通常被称为第一阶段单纯形，并且通常随后是第二阶段，在第二阶段中，根据代价函数对解决方案进行了优化。但是，由于我们仅考虑可满足性，因此不需要第二阶段。

It is well-known that the simplex calculus is *sound* [34] (i.e. if a derivation ends in `SAT` or `UNSAT`, then the original problem is satisfiable or unsatisfiable, respectively) and *complete* (there always exists a derivation ending in either `SAT` or `UNSAT` from any starting configuration). Termination can be guaranteed if certain strategies are used in applying the transition rules — in particular in picking the leaving and entering variables when multiple options exist [34]. Variable selection strategies are also known to have a dramatic effect on performance [34]. We note that the version of simplex described above is usually referred to as *phase one* simplex, and is usually followed by a *phase two* in which the solution is optimized according to a cost function. However, as we are only considering satisfiability, phase two is not required.

3 From Simplex to Reluplex

第2节中描述的单纯形算法是解决可以编码为原子连接的问题的有效方法。不幸的是，虽然DNN的权重，偏差和某些属性可以通过这种方式进行编码，但非线性ReLU函数却不行。

当理论求解器(sat solver)在SMT求解器中运行时，输入原子可以嵌入到任意布尔结构中。天真的方法是使用析取来编码ReLU，这是可能的，因为ReLU是分段线性的。但是，此编码要求SMT求解器中的SAT引擎枚举不同的情况。 在最坏的情况下，对于具有n个ReLU节点的DNN，求解器最终将问题分解为2^n个子问题，每个子问题都是一个原子的合取。正如我们和其他人所观察到的那样，在实践中也可以看到这种理论上的最坏情况行为，因此该方法仅适用于非常小的网络。 将DNN编码为混合整数问题时，会发生类似现象（请参见第6节）。

The simplex algorithm described in Section 2 is an efficient means for solving problems that can be encoded as a conjunction of atoms. Unfortunately, while the weights, biases, and certain properties of DNNs can be encoded this way, the non-linear ReLU functions cannot.

When a theory solver operates 运行 within an SMT solver, input atoms can be embedded 嵌入 in arbitrary Boolean structure. A naïve 天真 approach is then to encode ReLUs using disjunctions, which is possible because ReLUs are piecewise linear. However, this encoding requires the SAT engine within the SMT solver to enumerate the different cases. In the worst case, for a DNN with n ReLU nodes, the solver ends up splitting the problem into 2^n sub-problems, each of which is a conjunction of atoms. As observed by us and others [3, 10], this theoretical worst-case behavior is also seen in practice, and hence this approach is practical only for very small networks. A similar phenomenon occurs when encoding DNNs as mixed integer problems (see Section 6).

我们采取了不同的途径，并将理论TR扩展为实数和ReLU的理论TRR。 TRR与TR几乎相同，不同之处在于TRR的字母表另外包括二进制谓词ReLU，其解释为：ReLU(x,y)iff y = max(0,x)。然后假定公式包含的原子是线性不等式或ReLU谓词对线性项的应用。

We take a different route and extend the theory $\mathcal{T}_{\mathbb{R}}$ to a theory $\mathcal{T}_{\mathbb{R}R}$ of reals and ReLUs. $\mathcal{T}_{\mathbb{R}R}$ is almost identical to $\mathcal{T}_{\mathbb{R}}$, except that its signature additionally includes the binary predicate ReLU with the interpretation: ReLU(x, y) iff $y =$

max $(0, x)$. Formulas are then assumed to contain atoms that are either linear inequalities or applications of the ReLU predicate to linear terms.

DNN及其（线性）属性可以被直接编码为TRR原子的合取。主要思想是将单个ReLU节点v编码为一对变量vb和vf，然后断言ReLU(vb,vf)。vb是向后变量，用于表示v与上一层节点的连接；而前向变量vf用于x与下一层的连接（请参见图4）。本节的其余部分专门介绍一种有效的算法Reluplex，用于确定此类原子公式的合取的可满足性。

DNNs and their (linear) properties can be directly encoded as conjunctions of $\mathcal{T}_{\mathbb{R}R}$-atoms. The main idea is to encode a single ReLU node v as a *pair* of variables, v^b and v^f, and then assert ReLU(v^b, v^f). v^b, the *backward-facing* variable, is used to express the connection of v to nodes from the preceding layer; whereas v^f, the *forward-facing* variable, is used for the connections of x to the following layer (see Fig. 4). The rest of this section is devoted to presenting an efficient algorithm, Reluplex, for deciding the satisfiability of a conjunction of such atoms. such atoms中的atom指什么？？？

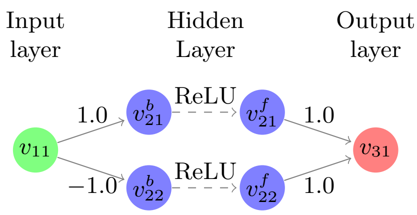


Fig. 4: The network from Fig. 2, with ReLU nodes split into backward- and forward-facing variables.

与单纯形一样，Reluplex允许变量在迭代寻找可行的变量assignment时暂时超出其边界。但是，Reluplex还允许变量作为ReLU pair的成员暂时违反ReLU语义。然后，在进行迭代时，Reluplex会反复选择超出边界或违反ReLU的变量，并使用Pivot和Update操作对其进行更正。

The Reluplex Procedure. As with simplex, Reluplex allows variables to temporarily violate their bounds as it iteratively looks for a feasible variable assignment. However, Reluplex also allows variables that are members of ReLU pairs to temporarily violate the ReLU semantics. Then, as it iterates, Reluplex repeatedly picks variables that are either out of bounds or that violate a ReLU, and corrects them using Pivot and Update operations.

对于给定的变量集X = {x1，。。。，xn}，Reluplex配置是专有符号{SAT，UNSAT}之一或元组hB，T，l，u，α，Ri，其中B，T，l，u和α像以前一样，而R ?X×X是ReLU连接的集合。 除了hx，yi∈R iff ReLU（x，y）是一个原子外，还像以前一样获得原子连接的初始构型。 Reluplex中还包含单纯形转换规则Pivot1，Pivot2和Update，因为它们是为处理越界违规而设计的。 我们用ReluSuccess规则替换Success规则，并添加用于处理ReLU违规的规则，如图5所示。Updateb和Updatef规则允许分别通过更新向后或向前的变量来纠断开的ReLU连接，前提是这些变量是非基变量。 PivotForRelu规则允许对ReLU中出现的基变量进行旋转，以便可以应用Updateb或Updatef（当ReLU中的两个变量都是基变量并且它们的assignment不满足ReLU语义时，这是必需的）。ReluSplit规则用于在某些确定的ReLU连接上进行splitting，猜测它们是处于活动状态（通过设置l(xi)=0）还是处于非活动状态（通过设置u(xi)=0）。

For a given set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$, a Reluplex configuration is either one of the distinguished symbols $\{\texttt{SAT}, \texttt{UNSAT}\}$ or a tuple $\langle \mathcal{B}, T, l, u, \alpha, R \rangle$, where $\mathcal{B}, T, l, u$ and α are as before, and $R \subset \mathcal{X} \times \mathcal{X}$ is the set of ReLU connections. The initial configuration for a conjunction of atoms is also obtained as before except that $\langle x, y \rangle \in R$ iff ReLU(x, y) is an atom. The simplex transition rules Pivot_1, Pivot_2 and Update are included also in Reluplex, as they are designed to handle out-of-bounds violations. We replace the $\mathsf{Success}$ rule with the $\mathsf{ReluSuccess}$ rule and add rules for handling ReLU violations, as depicted in Fig. 5. The Update_b and Update_f rules allow a broken ReLU connection to be corrected by updating the backward- or forward-facing variables, respectively, provided that these variables are non-basic. The $\mathsf{PivotForRelu}$ rule allows a basic variable appearing in a ReLU to be pivoted so that either Update_b or Update_f can be applied (this is needed to make progress when both variables in a ReLU are basic and their assignments do not satisfy the ReLU semantics). The $\mathsf{ReluSplit}$

$$\mathsf{Update}_b \quad \frac{x_i \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i)), \quad \alpha(x_j) \geq 0}{\alpha := update(\alpha, x_i, \alpha(x_j) - \alpha(x_i))}$$

$$\mathsf{Update}_f \quad \frac{x_j \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i))}{\alpha := update(\alpha, x_j, \max(0, \alpha(x_i)) - \alpha(x_j))}$$

$$\mathsf{PivotForRelu} \quad \frac{x_i \in \mathcal{B}, \quad \exists x_l.\ \langle x_i, x_l \rangle \in R \vee \langle x_l, x_i \rangle \in R, \quad x_j \notin \mathcal{B}, \quad T_{i,j} \neq 0}{T := pivot(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\mathsf{ReluSplit} \quad \frac{\langle x_i, x_j \rangle \in R, \quad l(x_i) < 0, \quad u(x_i) > 0}{u(x_i) := 0 \qquad l(x_i) := 0}$$

$$\mathsf{ReluSuccess} \quad \frac{\forall x \in \mathcal{X}.\ l(x) \leq \alpha(x) \leq u(x), \quad \forall \langle x^b, x^f \rangle \in R.\ \alpha(x^f) = \max(0, \alpha(x^b))}{\mathtt{SAT}}$$

Fig. 5: Additional derivation rules for the abstract Reluplex algorithm.

rule is used for splitting on certain ReLU connections, guessing whether they are active (by setting $l(x_i) := 0$) or inactive (by setting $u(x_i) := 0$).

引入拆分意味着导数不再是线性的。使用派生树的概念，我们可以证明Reluplex是健全且完备的（请参阅附录第二节）。在实践中，可以通过SAT引擎按需拆分来管理拆分[1]。可以通过eagerly应用ReluSplit规则直到不再适用，然后分别解决每个派生的子问题，来模拟本节开头提到的navie的方法（与混合整数求解器中的分支和剪切技术一样，这种简化也保证了终止）。但是，一种更具可扩展性的策略是尝试首先使用Updateb和Updatef规则修复损坏的ReLU对，并仅在对特定ReLU对的更新数量超过某个阈值时才进行拆分。 从直觉上讲，这很可能将拆分限制为“ 有问题的 ” ReLU对，同时可以保证终止（请参见附录第三节）。 更多详细信息，请参见第6节。

Introducing splitting means that derivations are no longer linear. Using the notion of derivation trees, we can show that Reluplex is sound and complete (see Section II of the appendix). In practice, splitting can be managed by a SAT engine with splitting-on-demand [1]. The naïve approach mentioned at the beginning of this section can be simulated by applying the ReluSplit rule eagerly until it no longer applies and then solving each derived sub-problem separately (this reduction trivially guarantees termination just as do branch-and-cut techniques in mixed integer solvers [28]). However, a more scalable strategy is to try to fix broken ReLU pairs using the Update_b and Update_f rules first, and split only when the number of updates to a specific ReLU pair exceeds some threshold. Intuitively, this is likely to limit splits to "problematic" ReLU pairs, while still guaranteeing termination (see Section III of the appendix). Additional details appear in Section 6.

为了说明派生规则的用法，我们使用Reluplex来解决一个简单的示例。 考虑图4中的网络，并假设我们希望检查是否可以满足v11∈[0,1]和v31∈[0.5,1]。我们知道网络输出的输入保持不变（v31恒等于v11），我们期望Reluplex能够得出SAT。初始Reluplex conf通过引入新的基本变量a1，a2，a3并使用如下的等式对network进行编码。

Example. To illustrate the use of the derivation rules, we use Reluplex to solve a simple example. Consider the network in Fig. 4, and suppose we wish to check whether it is possible to satisfy $v_{11} \in [0, 1]$ and $v_{31} \in [0.5, 1]$. As we know that the network outputs its input unchanged ($v_{31} \equiv v_{11}$), we expect Reluplex to be able to derive `SAT`. The initial Reluplex configuration is obtained by introducing new basic variables a_1, a_2, a_3, and encoding the network with the equations:

$$a_1 = -v_{11} + v_{21}^b \qquad a_2 = v_{11} + v_{22}^b \qquad a_3 = -v_{21}^f - v_{22}^f + v_{31}$$

The equations above form the initial tableau T_0, and the initial set of basic variables is $\mathcal{B} = \{a_1, a_2, a_3\}$. The set of ReLU connections is $R = \{\langle v_{21}^b, v_{21}^f \rangle, \langle v_{22}^b, v_{22}^f \rangle\}$. The initial assignment of all variables is set to 0. The lower and upper bounds of the basic variables are set to 0, in order to enforce the equalities that they represent. The bounds for the input and output variables are set according to the problem at hand; and the hidden variables are unbounded, except that forward-facing variables are, by definition, non-negative:

根据定义前向变量应该是负的

variable	v_{11}	v^b_{21}	v^f_{21}	v^b_{22}	v^f_{22}	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0	0	0	0	0	0	0	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

Starting from this initial configuration, our search strategy is to first fix any out-of-bounds variables. Variable v_{31} is non-basic and is out of bounds, so we perform an Update step and set it to 0.5. As a result, a_3, which depends on v_{31}, is also set to 0.5. a_3 is now basic and out of bounds, so we pivot it with v^f_{21}, and then update a_3 back to 0. The tableau now consists of the equations:

$$a_1 = -v_{11} + v^b_{21} \qquad a_2 = v_{11} + v^b_{22} \qquad v^f_{21} = -v^f_{22} + v_{31} - a_3$$

And the assignment is $\alpha(v^f_{21}) = 0.5$, $\alpha(v_{31}) = 0.5$, and $\alpha(v) = 0$ for all other variables v. At this point, all variables are within their bounds, but the ReluSuccess rule does not apply because $\alpha(v^f_{21}) = 0.5 \neq 0 = \max(0, \alpha(v^b_{21}))$.

The next step is to fix the broken ReLU pair $\langle v^b_{21}, v^f_{21} \rangle$. Since v^b_{21} is non-basic, we use Update_b to increase its value by 0.5. The assignment becomes $\alpha(v^b_{21}) = 0.5$, $\alpha(v^f_{21}) = 0.5$, $\alpha(v_{31}) = 0.5$, $\alpha(a_1) = 0.5$, and $\alpha(v) = 0$ for all other variables v. All ReLU constraints hold, but a_1 is now out of bounds. This is fixed by pivoting a_1 with v_{11} and then updating it. The resulting tableau is:

$$v_{11} = v^b_{21} - a_1 \qquad a_2 = v^b_{21} + v^b_{22} - a_1 \qquad v^f_{21} = -v^f_{22} + v_{31} - a_3$$

Observe that because v_{11} is now basic, it was eliminated from the equation for a_2 and replaced with $v^b_{21} - a_1$. The non-zero assignments are now $\alpha(v_{11}) = 0.5$, $\alpha(v^b_{21}) = 0.5$, $\alpha(v^f_{21}) = 0.5$, $\alpha(v_{31}) = 0.5$, $\alpha(a_2) = 0.5$. Variable a_2 is now too large, and so we have a final round of pivot-and-update: a_2 is pivoted with v^b_{22} and then updated back to 0. The final tableau and assignments are:

$$\begin{aligned} v_{11} &= v^b_{21} - a_1 \\ v^b_{22} &= -v^b_{21} + a_1 + a_2 \\ v^f_{21} &= -v^f_{22} + v_{31} - a_3 \end{aligned}$$

variable	v_{11}	v^b_{21}	v^f_{21}	v^b_{22}	v^f_{22}	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0.5	0.5	0.5	-0.5	0	0.5	0	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

并且该算法会在找到满意且可行的方案后停止。一个主要的结论是，我们从未分割任何ReLU连接。相反，只需更新调整ReLU变量就足够了。

and the algorithm halts with the feasible solution it has found. A key observation is that we did not ever split on any of the ReLU connections. Instead, it was sufficient to simply use updates to adjust the ReLU variables as needed.

4 Efficiently Implementing Reluplex

接下来，我们讨论三种显着提高Reluplex性能的技术：使用更严格的边界推导，冲突分析和浮点算法。附录IV部分讨论了第四种技术，近似法。

We next discuss three techniques that significantly boost the performance of Reluplex: use of tighter bound derivation, conflict analysis and floating point arithmetic. A fourth technique, under-approximation, is discussed in Section IV of the appendix.

四种显著提高速度的技术

单纯形和Reluplex过程自然会随着搜索的进行而推导更严格的变量范围。

...
在整个执行过程中，以下规则可用于得出xi的更严格边界，而与当前分配无关。
...

边界缩紧

Tighter Bound Derivation. The simplex and Reluplex procedures naturally lend themselves to deriving tighter variable bounds as the search progresses [16]. Consider a basic variable $x_i \in \mathcal{B}$ and let $\mathrm{pos}(x_i) = \{x_j \notin \mathcal{B} \mid T_{i,j} > 0\}$ and $\mathrm{neg}(x_i) = \{x_j \notin \mathcal{B} \mid T_{i,j} < 0\}$. Throughout the execution, the following rules can be used to derive tighter bounds for x_i, regardless of the current assignment:

$$\textsf{deriveLowerBound}\quad \frac{x_i \in \mathcal{B},\quad l(x_i) < \sum_{x_j \in \mathrm{pos}(x_i)} T_{i,j} \cdot l(x_j) + \sum_{x_j \in \mathrm{neg}(x_i)} T_{i,j} \cdot u(x_j)}{l(x_i) := \sum_{x_j \in \mathrm{pos}(x_i)} T_{i,j} \cdot l(x_j) + \sum_{x_j \in \mathrm{neg}(x_i)} T_{i,j} \cdot u(x_j)}$$

$$\textsf{deriveUpperBound}\quad \frac{x_i \in \mathcal{B},\quad u(x_i) > \sum_{x_j \in \mathrm{pos}(x_i)} T_{i,j} \cdot u(x_j) + \sum_{x_j \in \mathrm{neg}(x_i)} T_{i,j} \cdot l(x_j)}{u(x_i) := \sum_{x_j \in \mathrm{pos}(x_i)} T_{i,j} \cdot u(x_j) + \sum_{x_j \in \mathrm{neg}(x_i)} T_{i,j} \cdot l(x_j)}$$

推导出的边界以后可以用于推导其他更严格的边界。

The derived bounds can later be used to derive additional, tighter bounds.

当为ReLU变量推导出更严格的界限时，有时可以消除这些RELU变量，即将其固定为活动或不活动状态而不会分裂。对于ReLU对xf = ReLU(xb)，发现l(xb)或l(xf)严格为正意味着在任何可行的解决方案中，此ReLU连接都将处于激活状态。 同样，发现u(xb)<0表示始终处于非激活状态。严格的缩紧操作会产生开销，并且单纯形实现经常会稀疏地使用它们。但是，在Reluplex中，消除ReLU节点的好处证明了开销是值得的。可以通过试探法确定要执行的约束缩紧的实际量； 我们将在第6节中描述使用的启发式方法。

When tighter bounds are derived for ReLU variables, these variables can sometimes be eliminated, i.e., fixed to the active or inactive state, without splitting. For a ReLU pair $x^f = \mathrm{ReLU}(x^b)$, discovering that either $l(x^b)$ or $l(x^f)$ is strictly positive means that in any feasible solution this ReLU connection will be active. Similarly, discovering that $u(x^b) < 0$ implies inactivity.

Bound tightening operations incur overhead, and simplex implementations often use them sparsely [16]. In Reluplex, however, the benefits of eliminating ReLUs justify the cost. The actual amount of bound tightening to perform can be determined heuristically; we describe the heuristic that we used in Section 6.

边界推导会导致一些情况，其中我们了解到对于某些变量x，l(x)> u(x)。 此类矛盾使Reluplex可以立即撤消先前的拆分（或者如果不存在先前的拆分，则回答UNSAT）。 但是，在许多情况下，不仅可以以前的拆分可以撤销。例如，如果到目前为止我们已经执行了8个嵌套拆分，则x的冲突边界可能是数字5的拆分的直接结果，但只是刚刚被发现。在这种情况下，我们可以立即撤消数字8、7和6的拆分。这是冲突分析的一种特殊情况，这是SAT和SMT求解器中的标准技术[25]。

边界缩紧产生的冲突分析

Derived Bounds and Conflict Analysis. Bound derivation can lead to situations where we learn that $l(x) > u(x)$ for some variable x. Such contradictions allow Reluplex to immediately undo a previous split (or answer `UNSAT` if no previous splits exist). However, in many cases more than just the previous split can be undone. For example, if we have performed 8 nested splits so far, it may be that the conflicting bounds for x are the direct result of split number 5 but have only just been discovered. In this case we can immediately undo splits number 8, 7, and 6. This is a particular case of *conflict analysis*, which is a standard technique in SAT and SMT solvers [25].

SMT求解器通常使用精确的算法（而不是浮点算法）来避免舍入误差并确保稳健性。 不幸的是，精确计算通常比其浮点等效项至少慢一个数量级。 在大型DNN上调用Reluplex可能需要数百万次Pivot运算，每个运算都涉及有理数的乘法和除法，并且可能使用大的分子或分母，因此使用浮点算法对于可伸缩性很重要。

浮点运算

Floating Point Arithmetic. SMT solvers typically use precise (as opposed to floating point) arithmetic to avoid roundoff errors and guarantee soundness. Unfortunately, precise computation is usually at least an order of magnitude slower than its floating point equivalent. Invoking Reluplex on a large DNN can require millions of pivot operations, each of which involves the multiplication and division of rational numbers, potentially with large numerators or denominators — making the use of floating point arithmetic important for scalability.

有一些标准技术可以在使用浮点数实现单纯形时使舍入误差保持较小，我们将其纳入实现中。 例如，一种重要的实践是在Pivot时，试图避免涉及极小数的倒置。

There are standard techniques for keeping the roundoff error small when implementing simplex using floating point, which we incorporated into our implementation. For example, one important practice is trying to avoid Pivot operations involving the inversion of extremely small numbers [34].

To provide increased confidence that any roundoff error remained within an acceptable range, we also added the following safeguards: (i) After a certain

为了增强对任何舍入误差仍在可接受范围内的信心，我们还添加了以下保护措施：（i）在经过一定数量的Pivot步骤之后，我们将测量累积的舍入误差；（ii）如果误差超过阈值M，我们将使用初始表格T0恢复当前表格T的系数。

number of Pivot steps we would measure the accumulated roundoff error; and (ii) If the error exceeded a threshold M, we would *restore* the coefficients of the current tableau T using the initial tableau T_0.

可以通过将非基变量的当前assignment值插入初始表格T0的方程中，并使用它们来计算每个基变量xi的值，来测量累积舍入误差。然后测量这些值与当前分配值α（xi）相差多少。我们将累积舍入误差定义为：

Cumulative roundoff error can be measured by plugging the current assignment values for the non-basic variables into the equations of the initial tableau T_0, using them to calculate the values for every basic variable x_i, and then measuring by how much these values differ from the current assignment $\alpha(x_i)$. We define the cumulative roundoff error as:

$$\sum_{x_i \in \mathcal{B}_0} |\alpha(x_i) - \sum_{x_j \notin \mathcal{B}_0} T_{0_{i,j}} \cdot \alpha(x_j)|$$

通过从T0开始并执行短的一系列Pivot步骤来恢复T，这将导致与T中的基变量集相同。通常，将T0转换为T的最短Pivot步骤序列比这一系列步骤要短得多。紧随其后的是Reluplex-因此，尽管它也使用浮点算法执行，但其舍入误差较小。

T is restored by starting from T_0 and performing a short series of Pivot steps that result in the same set of basic variables as in T. In general, the shortest sequence of pivot steps to transform T_0 to T is much shorter than the series of steps that was followed by Reluplex — and hence, although it is also performed using floating point arithmetic, it incurs a smaller roundoff error.

当使用浮点算法时，Tableau恢复技术可增强我们对算法结果的信心，但不会保证可靠性。 使用浮点算法时提供真实的可靠性仍然是未来的目标（请参见第8节）

The tableau restoration technique serves to increase our confidence in the algorithm's results when using floating point arithmetic, but it does not guarantee soundness. Providing true soundness when using floating point arithmetic remains a future goal (see Section 8).

5 Case Study: The ACAS Xu System

机载防撞系统对于确保飞机的安全运行至关重要。交通警报和防撞系统（TCAS）是针对商用飞机之间的空中碰撞而开发的，目前已在全球所有大型商用飞机上使用。最近的工作集中在创建一个新系统，称为机载防撞系统X（ACAS X）。该系统采用的方法涉及解决部分可观察的马尔可夫决策过程以优化警报逻辑并进一步减少空中碰撞的可能性，同时将不必要的警报降到最低。

Airborne collision avoidance systems are critical for ensuring the safe operation of aircraft. The *Traffic Alert and Collision Avoidance System* (*TCAS*) was developed in response to midair collisions between commercial aircraft, and is currently mandated on all large commercial aircraft worldwide [23]. Recent work has focused on creating a new system, known as *Airborne Collision Avoidance System X* (*ACAS X*) [18, 19]. This system adopts an approach that involves solving a partially observable Markov decision process to optimize the alerting logic and further reduce the probability of midair collisions, while minimizing unnecessary alerts [18, 19, 21].

ACAS X的无人驾驶型，称为ACAS Xu，可产生水平机动警报。到目前为止，ACAS Xu的开发工作一直集中在使用大型查找表来将传感器的测量结果映射到建议中。但是，此表需要2GB以上的内存。对于经认证的航空电子硬件的内存要求存在担忧。为了克服这一挑战，探索了一种DNN表示形式来代替表格。初步结果表明，在不损害安全性的情况下，内存需求显着减少。实际上，由于其连续性，DNN方法有时会优于离散查找表。最近，为了减少查找时间，DNN方法得到了进一步改进，单个DNN被45个DNN的阵列所取代。结果，原始的2GB表现在可以用需要少于3MB内存的高效DNN代替。

The unmanned variant of ACAS X, known as ACAS Xu, produces horizontal maneuver advisories. So far, development of ACAS Xu has focused on using a large lookup table that maps sensor measurements to advisories [13]. However, this table requires over 2GB of memory. There is concern about the memory requirements for certified avionics hardware. To overcome this challenge, a DNN representation was explored as a potential replacement for the table [13]. Initial results show a dramatic reduction in memory requirements without compromising safety. In fact, due to its continuous nature, the DNN approach can sometimes outperform the discrete lookup table [13]. Recently, in order to reduce lookup time, the DNN approach was improved further, and the single DNN was replaced by an array of 45 DNNs. As a result, the original 2GB table can now be substituted with efficient DNNs that require less than 3MB of memory.

ACAS Xu的DNN实施提出了新的认证挑战。证明一组输入不会产生错误的警报对于认证系统可用于安全关键设置至关重要。以前的认证方法包括在150万次模拟遭遇中对系统进行详尽的测试，但这不足以证明连续DNN中不存在错误行为。这凸显了验证DNN的需求，并使ACAS Xu DNN成为应用Reluplex的主要候选人。

A DNN implementation of ACAS Xu presents new certification challenges. Proving that a set of inputs cannot produce an erroneous alert is paramount for certifying the system for use in safety-critical settings. Previous certification methodologies included exhaustively testing the system in 1.5 million simulated encounters [20], but this is insufficient for proving that faulty behaviors do not exist within the continuous DNNs. This highlights the need for verifying DNNs and makes the ACAS Xu DNNs prime candidates on which to apply Reluplex.

ACAS Xu系统将输入变量映射到行动警报上。为每个警报分配一个分数，最低分数对应最佳行动。输入状态由七个维度（如图6所示）组成，这些维度代表根据传感器测量值确定的信息：
（i） ρ ：自身到入侵者的距离；
（ii） θ ：相对于自身前进方向与入侵者的夹角；
（iii） ψ：入侵者相对于自身航向的航向角；
（iv）Vown：自身速度；
(v) Vint：入侵者的速度；
（vi） τ ：直至垂直分离消失的时间；
（vii）a_prev：上一个警报。

有五个输出代表可以为该拥有权提供的不同水平咨询：冲突消除（COC），右弱，右强，左弱或左强。 衡量强和弱的平均航向率分别为1.5度/s和3.0度/s。

Network Functionality. The ACAS Xu system maps input variables to action advisories. Each advisory is assigned a score, with the lowest score corresponding to the best action. The input state is composed of seven dimensions (shown in Fig. 6) which represent information determined from sensor measurements [19]: (i) ρ: Distance from ownship to intruder; (ii) θ: Angle to intruder relative to ownship heading direction; (iii) ψ: Heading angle of intruder relative to ownship heading direction; (iv) v_{own}: Speed of ownship; (v) v_{int}: Speed of intruder; (vi) τ: Time until loss of vertical separation; and (vii) a_{prev}: Previous advisory. There are five outputs which represent the different horizontal advisories that can be given to the ownship: Clear-of-Conflict (COC), weak right, strong right, weak left, or strong left. Weak and strong mean heading rates of 1.5 °/s and 3.0 °/s, respectively.

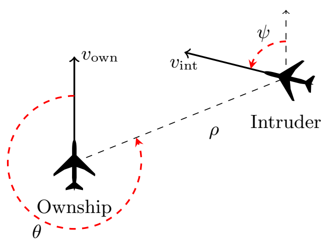


Fig. 6: Geometry for ACAS Xu Horizontal Logic Table

通过离散化τ和aprev可以生成45个DNN的列表，并为每个离散组合生成网络。因此，这些列表中的每一个网络都有五个输入（其他维度每个都有一个）和五个输出。 DNN都是全连接的，使用ReLU激活功能，并具有6个隐藏层，每个隐藏层共有300个ReLU节点。

The array of 45 DNNs was produced by discretizing τ and a_{prev}, and producing a network for each discretized combination. Each of these networks thus has five inputs (one for each of the other dimensions) and five outputs. The DNNs are fully connected, use ReLU activation functions, and have 6 hidden layers with a total of 300 ReLU nodes each.

希望验证ACAS Xu网络在各个输入域中为输出警报分配了正确的分数。 图7通过显示正面面对场景的俯视图说明了这种属性，其中，如果入侵者位于该位置，则每个像素都将上色以表示最佳操作。我们希望DNN输出的警报在这些区域中保持一致。

Network Properties. It is desirable to verify that the ACAS Xu networks assign correct scores to the output advisories in various input domains. Fig. 7 illustrates this kind of property by showing a top-down view of a head-on encounter scenario, in which each pixel is colored to represent the best action if the intruder were at that location. We expect the DNN's advisories to be consistent in each of these regions; however, Fig. 7 was generated from a finite set

但是，图7是从一组有限的输入样本生成的，并且可能存在其他输入，这些输入会产生错误的警报，从而可能导致冲突。 因此，我们使用Reluplex来从DNN上的以下类别中证明属性：
（i）系统没有给出不必要的转向警报
（ii）警报区域是统一的，并且没有不一致的警报；
（iii）对于高τ值，不会出现强烈警报。

of input samples, and there may exist other inputs for which a wrong advisory is produced, possibly leading to collision. Therefore, we used Reluplex to prove properties from the following categories on the DNNs: (i) The system does not give unnecessary turning advisories; (ii) Alerting regions are uniform and do not contain inconsistent alerts; and (iii) Strong alerts do not appear for high τ values.

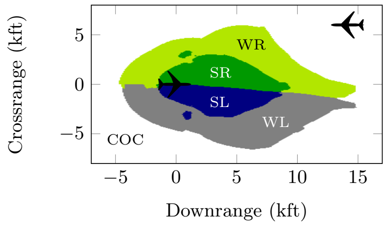


Fig. 7: Advisories for a head-on encounter with $a_{\text{prev}} = \text{COC}, \tau = 0\,\text{s}$.

6 Evaluation

我们使用了Reluplex的概念验证实现来检查45个ACAS Xu DNN的真实属性。 我们的实现包含三个主要逻辑组件：
（i）一个单纯形引擎，用于提供核心功能，例如表格表示以及数据透视和更新操作；
（ii）Reluplex引擎，用于驱动搜索并执行绑定推导，ReLU数据透视和ReLU更新；
（iii）提供按需拆分服务的简单SMT核心。
对于单纯形引擎，我们使用了GLPK开源LP求解器并进行了一些修改，例如，以便允许Reluplex核心对由GLPK计算的表格方程式进行约束紧缩。我们的实现以及本节中描述的实验可在线获得。

We used a proof-of-concept implementation of Reluplex to check realistic properties on the 45 ACAS Xu DNNs. Our implementation consists of three main logical components: (i) A simplex engine for providing core functionality such as tableau representation and pivot and update operations; (ii) A Reluplex engine for driving the search and performing bound derivation, ReLU pivots and ReLU updates; and (iii) A simple SMT core for providing splitting-on-demand services. For the simplex engine we used the GLPK open-source LP solver[3] with some modifications, for instance in order to allow the Reluplex core to perform bound tightening on tableau equations calculated by GLPK. Our implementation, together with the experiments described in this section, is available online [14].

我们的搜索策略是首先反复修复任何越界违规，然后才纠正违规的ReLU约束（可能引入新的越界违规）。每次执行Pivot操作后，我们都会对输入变量进行边界紧缩，并且每隔几千次Pivot步骤，就对表格中的所有方程式进行更彻底的边界紧缩。严格的边界推导被证明是非常有用的，而且我们经常观察到，在分解大约10%的ReLU变量后，它导致消除了所有剩余的ReLU。我们计算了ReLUpair通过Updateb或Updatef固定或通过PivotForRelu进行旋转的次数，并且仅在此数目达到5（凭经验确定表现良好的数字）时才进行拆分。我们还实现了冲突分析和后跳。最后，我们每隔5000个Pivot步骤检查一次累积的舍入误差（由于使用了双精度浮点算术），如果误差超过10^6，则恢复表格。下文所述的大多数实验都需要进行两次或更少的tableau还原。

Our search strategy was to repeatedly fix any out-of-bounds violations first, and only then correct any violated ReLU constraints (possibly introducing new out-of-bounds violations). We performed bound tightening on the entering variable after every pivot operation, and performed a more thorough bound tightening on all the equations in the tableau once every few thousand pivot steps. Tighter bound derivation proved extremely useful, and we often observed that after splitting on about 10% of the ReLU variables it led to the elimination of all remaining ReLUs. We counted the number of times a ReLU pair was fixed via Update_b or Update_f or pivoted via $\mathsf{PivotForRelu}$, and split only when this number reached 5 (a number empirically determined to work well). We also implemented conflict analysis and back-jumping. Finally, we checked the accumulated round-off error (due to the use of double-precision floating point arithmetic) after every

[3] `www.gnu.org/software/glpk/`

5000 Pivot steps, and restored the tableau if the error exceeded 10^{-6}. Most experiments described below required two tableau restorations or fewer.

我们首先将Reluplex的实现与最先进的求解器进行比较：CVC4，Z3，Yices和MathSat SMT以及Gurobi LP（请参见表1）。我们在2个ACAS Xu网络（任意选择）上以4小时超时运行了所有求解器，试图求解8个简单可满足的特性，对于固定的输出变量x和常数c，每种形式x≥c。SMT求解器的性能通常很差，只有Yices和MathSat分别成功求解了两个实例。我们将结果归因于这些求解器对编码ReLU的缺乏直接支持，以及对精确算术的使用。 Gurobi快速解决了3个实例，但其余所有实例均超时。它的日志表明，只要Gurobi不需要进行case-splitting，它的速度就会很快。但是只要问题需要进行case-splitting，Gurobi就会超时。 Reluplex能够解决所有8个实例。有关我们使用的SMT和LP编码，请参见附录的第V节。

We began by comparing our implementation of Reluplex to state-of-the-art solvers: the CVC4, Z3, Yices and MathSat SMT solvers and the Gurobi LP solver (see Table 1). We ran all solvers with a 4 hour timeout on 2 of the ACAS Xu networks (selected arbitrarily), trying to solve for 8 simple satisfiable properties $\varphi_1, \ldots, \varphi_8$, each of the form $x \geq c$ for a fixed output variable x and a constant c. The SMT solvers generally performed poorly, with only Yices and MathSat successfully solving two instances each. We attribute the results to these solvers' lack of direct support for encoding ReLUs, and to their use of precise arithmetic. Gurobi solved 3 instances quickly, but timed out on all the rest. Its logs indicated that whenever Gurobi could solve the problem without case-splitting, it did so quickly; but whenever the problem required case-splitting, Gurobi would time out. Reluplex was able to solve all 8 instances. See Section V of the appendix for the SMT and LP encodings that we used.

Table 1: Comparison to SMT and LP solvers. Entries indicate solution time (in seconds).

	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8
CVC4	-	-	-	-	-	-	-	-
Z3	-	-	-	-	-	-	-	-
Yices	1	37	-	-	-	-	-	-
MathSat	2040	9780	-	-	-	-	-	-
Gurobi	1	1	1	-	-	-	-	-
Reluplex	8	2	7	7	93	4	7	9

接下来，我们使用Reluplex来测试一组10个定量属性。 下文所述的属性在附录VI中正式定义。 表2列出了每个属性的测试网络数（指定为属性的一部分），测试结果和总持续时间（以秒为单位）。 “堆栈”和“拆分”列分别列出了达到的嵌套案例拆分的最大深度（在测试网络中平均）和执行的案例拆分的总数。 对于每个属性，我们都寻找一个违反该属性的输入。 因此，一个SAT结果表明一个财产持有，而一个SAT结果表明它不持有。 在SAT情况下，令人满意的分配是违反属性的输入的示例。

Next, we used Reluplex to test a set of 10 quantitative properties $\phi_1, \ldots, \phi_{10}$. The properties, described below, are formally defined in Section VI of the appendix. Table 2 depicts for each property the number of tested networks (specified as part of the property), the test results and the total duration (in seconds). The *Stack* and *Splits* columns list the maximal depth of nested case-splits reached (averaged over the tested networks) and the total number of case-splits performed, respectively. For each property, we looked for an input that would violate it; thus, an `UNSAT` result indicates that a property holds, and a `SAT` result indicates that it does not hold. In the `SAT` case, the satisfying assignment is an example of an input that violates the property.

属性φ1指出，如果入侵者距离自身远且比自己慢得多，则COC警报的分数将始终低于某个固定阈值（请记住，最佳动作的分数最低）。特性φ2指出，在类似条件下，COC分数永远不会达到最高，这意味着它绝不会是最糟糕的行为。 人们发现此属性不适用于35个网络，但后来被确定为可以接受的行为：DNN在产生与之前产生的相同建议方面有很大的偏见，这甚至会导致COC以外的其他建议。 如果先前的建议不是COC，则请远离入侵者。 属性φ3和φ4处理入侵者直接领先于所有权的情况，并声明DNN永远不会发出COC咨询。

Property ϕ_1 states that if the intruder is distant and is significantly slower than the ownship, the score of a COC advisory will always be below a certain fixed threshold (recall that the best action has the lowest score). Property ϕ_2 states that under similar conditions, the score for COC can never be maximal, meaning that it can never be the worst action to take. This property was discovered not to hold for 35 networks, but this was later determined to be acceptable behavior: the DNNs have a strong bias for producing the same advisory they

Table 2: Verifying properties of the ACAS Xu networks.

	Networks	Result	Time	Stack	Splits
ϕ_1	41	UNSAT	394517	47	1522384
	4	TIMEOUT			
ϕ_2	1	UNSAT	463	55	88388
	35	SAT	82419	44	284515
ϕ_3	42	UNSAT	28156	22	52080
ϕ_4	42	UNSAT	12475	21	23940
ϕ_5	1	UNSAT	19355	46	58914
ϕ_6	1	UNSAT	180288	50	548496
ϕ_7	1	TIMEOUT			
ϕ_8	1	SAT	40102	69	116697
ϕ_9	1	UNSAT	99634	48	227002
ϕ_{10}	1	UNSAT	19944	49	88520

had previously produced, and this can result in advisories other than COC even for far-away intruders if the previous advisory was also something other than COC. Properties ϕ_3 and ϕ_4 deal with situations where the intruder is directly ahead of the ownship, and state that the DNNs will never issue a COC advisory.

Properties ϕ_5 through ϕ_{10} each involve a single network, and check for consistent behavior in a specific input region. For example, ϕ_5 states that if the intruder is near and approaching from the left, the network advises “strong right”. Property ϕ_7, on which we timed out, states that when the vertical separation is large the network will never advise a strong turn. The large input domain and the particular network proved difficult to verify. Property ϕ_8 states that for a large vertical separation and a previous “weak left” advisory, the network will either output COC or continue advising “weak left”. Here, we were able to find a counter-example, exposing an input on which the DNN was inconsistent with the lookup table. This confirmed the existence of a discrepancy that had also been seen in simulations, and which will be addressed by retraining the DNN. We observe that for all properties, the maximal depth of nested splits was always well below the total number of ReLU nodes, 300, illustrating the fact that Reluplex did not split on many of them. Also, the total number of case-splits indicates that large portions of the search space were pruned.

属性φ7（已超时）指出，当垂直间距较大时，网络将永远不会建议大转弯。 大的输入域和特定的网络证明难以验证。属性φ8指出，对于较大的垂直间距和先前的“弱左”建议，网络将输出COC或继续建议“弱左”。 在这里，我们能够找到一个反例，公开了DNN与查找表不一致的输入。 这证实了在模拟中也已经看到的差异的存在，并且将通过对DNN进行重新培训来解决。 我们观察到，对于所有属性，嵌套拆分的最大深度始终远低于ReLU节点的总数300，这说明了Reluplex并未在其中许多拆分的事实。 同样，case split的总数表明搜索空间的大部分已被修剪。

Another class of properties that we tested is *adversarial robustness* properties. DNNs have been shown to be susceptible to adversarial inputs [33]: correctly classified inputs that an adversary slightly perturbs, leading to their misclassification by the network. Adversarial robustness is thus a safety consideration, and adversarial inputs can be used to train the network further, making it more robust [8]. There exist approaches for finding adversarial inputs [3, 8], but the ability to verify their absence is limited.

我们测试的另一类属性是对抗性鲁棒性。事实证明，DNN容易受到对抗性输入的影响：正确分类的输入会引起对抗的轻微干扰，从而导致网络对其进行错误分类。 因此，对抗性的鲁棒性是安全考虑，并且对抗性输入可用于进一步训练网络，从而使其更加鲁棒。 存在寻找对抗性输入的方法，但是验证其缺失的能力是有限的。

We say that a network is δ-*locally-robust* at input point $\boldsymbol{x}$ if for every $\boldsymbol{x'}$ such that $\|\boldsymbol{x}-\boldsymbol{x'}\|_\infty \leq \delta$, the network assigns the same label to $\boldsymbol{x}$ and $\boldsymbol{x'}$. In the case of the ACAS Xu DNNs, this means that the same output has the lowest score

Table 3: Local adversarial robustness tests. All times are in seconds.

	$\delta = 0.1$		$\delta = 0.075$		$\delta = 0.05$		$\delta = 0.025$		$\delta = 0.01$		Total
	Result	Time	Result	Time	Result	Time	Result	Time	Result	Time	Time
Point 1	`SAT`	135	`SAT`	239	`SAT`	24	`UNSAT`	609	`UNSAT`	57	1064
Point 2	`UNSAT`	5880	`UNSAT`	1167	`UNSAT`	285	`UNSAT`	57	`UNSAT`	5	7394
Point 3	`UNSAT`	863	`UNSAT`	436	`UNSAT`	99	`UNSAT`	53	`UNSAT`	1	1452
Point 4	`SAT`	2	`SAT`	977	`SAT`	1168	`UNSAT`	656	`UNSAT`	7	2810
Point 5	`UNSAT`	14560	`UNSAT`	4344	`UNSAT`	1331	`UNSAT`	221	`UNSAT`	6	20462

for both $\boldsymbol{x}$ and $\boldsymbol{x'}$. Reluplex can be used to prove local robustness for a given $\boldsymbol{x}$ and δ, as depicted in Table 3. We used one of the ACAS Xu networks, and tested combinations of 5 arbitrary points and 5 values of δ. `SAT` results show that Reluplex found an adversarial input within the prescribed neighborhood, and `UNSAT` results indicate that no such inputs exist. Using binary search on values of δ, Reluplex can thus be used for approximating the optimal δ value up to a desired precision: for example, for point 4 the optimal δ is between 0.025 and 0.05. It is expected that different input points will have different local robustness, and the acceptable thresholds will thus need to be set individually.

最后，我们提到了对抗性鲁棒性的另一个变体，我们称之为全局对抗性鲁棒性，也可以通过Relu plex解决。尽管针对特定x度量了本地对抗性鲁棒性，但全局对抗性鲁棒性同时适用于所有输入。这是通过对两个相关的DNN的并排副本N1和N2进行编码来表示的，分别对单独的输入变量x1和x2进行操作，以使x2表示x1的对抗性扰动。然后，我们可以检查kx1∞x2k∞≤ δ 是否暗示DNN的两个副本产生相似的输出。形式上，我们要求如果N1和N2分别分配输出值p1和p2，则|p1∞p2 | ≤。如果这对于每个输出都成立，那么我们说该网络是-global-robust。与本地变体相比，更难证明全局对抗性的鲁棒性，因为对网络的两个副本进行编码会导致两倍的ReLU节点，并且问题不仅限于较小的输入域。我们仅能在小型网络上证明全球对抗能力。改进此技术的可扩展性尚待将来工作。

Finally, we mention an additional variant of adversarial robustness which we term *global adversarial robustness*, and which can also be solved by Reluplex. Whereas local adversarial robustness is measured for a specific $\boldsymbol{x}$, global adversarial robustness applies to all inputs simultaneously. This is expressed by encoding two side-by-side copies of the DNN in question, N_1 and N_2, operating on separate input variables $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, respectively, such that $\boldsymbol{x_2}$ represents an adversarial perturbation of $\boldsymbol{x_1}$. We can then check whether $\|\boldsymbol{x_1} - \boldsymbol{x_2}\|_\infty \leq \delta$ implies that the two copies of the DNN produce similar outputs. Formally, we require that if N_1 and N_2 assign output a values p_1 and p_2 respectively, then $|p_1 - p_2| \leq \epsilon$. If this holds for every output, we say that the network is ϵ-globally-robust. Global adversarial robustness is harder to prove than the local variant, because encoding two copies of the network results in twice as many ReLU nodes and because the problem is not restricted to a small input domain. We were able to prove global adversarial robustness only on small networks; improving the scalability of this technique is left for future work.

7 Related Work

在[29]中，作者提出了一种方法来验证具有S型激活函数的神经网络的属性。他们用分段线性近似替换激活函数，然后调用黑盒SMT求解器。 当发现虚假的反例时，可以对近似值进行细化。 作者强调了扩展该技术的困难，并且只能处理最多具有20个隐藏节点的小型网络[30]。

In [29], the authors propose an approach for verifying properties of neural networks with sigmoid activation functions. They replace the activation functions with piecewise linear approximations thereof, and then invoke black-box SMT solvers. When spurious counter-examples are found, the approximation is refined. The authors highlight the difficulty in scaling-up this technique, and are able to tackle only small networks with at most 20 hidden nodes [30].

[3]的作者提出了一种在具有ReLU的DNN中寻找本地对抗性示例的技术。给定输入点x，他们将问题编码为线性程序并调用黑盒LP解算器。通过考虑x的足够小的邻域来规避激活函数问题，其中所有ReLU都固定在活动或非活动状态，从而使问题凸出。因此，不清楚如何解决一个x上一个或多个ReLU位于有效和无效状态之间的边界。相反，Reluplex可用于ReLU可能具有多个可能状态的输入域。

The authors of [3] propose a technique for finding local adversarial examples in DNNs with ReLUs. Given an input point $\boldsymbol{x}$, they encode the problem as a linear program and invoke a black-box LP solver. The activation function issue is circumvented by considering a sufficiently small neighborhood of $\boldsymbol{x}$, in which all ReLUs are fixed at the active or inactive state, making the problem convex. Thus, it is unclear how to address an $\boldsymbol{x}$ for which one or more ReLUs are on the boundary between active and inactive states. In contrast, Reluplex can be used on input domains for which ReLUs can have more than one possible state.

在最近的一篇论文中[10]，作者提出了一种证明DNN的局部对抗鲁棒性的方法。对于特定的输入点x，作者尝试通过离散化来证明x邻域中的一致标记：他们将无限邻域简化为一组有限的点，并检查这些点的标记是否一致。然后，此过程逐层通过网络传播。虽然从某种意义上说该技术是通用的，因为它不是为特定的激活函数量身定制的，但离散化过程意味着任何UNSAT结果仅以有限集正确表示其无限域的假设为模。相比之下，我们的技术可以确保离散点之间没有隐藏的不规则性。

In a recent paper [10], the authors propose a method for proving the local adversarial robustness of DNNs. For a specific input point $\boldsymbol{x}$, the authors attempt to prove consistent labeling in a neighborhood of $\boldsymbol{x}$ by means of discretization: they reduce the infinite neighborhood into a finite set of points, and check that the labeling of these points is consistent. This process is then propagated through the network, layer by layer. While the technique is general in the sense that it is not tailored for a specific activation function, the discretization process means that any `UNSAT` result only holds modulo the assumption that the finite sets correctly represent their infinite domains. In contrast, our technique can guarantee that there are no irregularities hiding between the discrete points.

最

后，在[12]中，作者采用混合技术来分析以查找表形式给出的ACAS X控制器，以寻求识别不会发生碰撞的安全输入区域。将我们的技术与[12]的技术相结合将是有趣的，以验证遵循DNN提供的建议确实可以避免冲突。

Finally, in [12], the authors employ hybrid techniques to analyze an ACAS X controller given in lookup-table form, seeking to identify *safe input regions* in which collisions cannot occur. It will be interesting to combine our technique with that of [12], in order to verify that following the advisories provided by the DNNs indeed leads to collision avoidance.

8 Conclusion and Next Steps

我们提出了一种新颖的决策算法，用于解决具有ReLU激活功能的深度神经网络上的查询。该技术基于扩展单纯形算法以支持非凸型ReLU，该方式允许它们的输入和输出暂时不一致，然后在算法进行时固定。为了保证端接，可能需要拆分一些ReLU连接-但是在许多情况下，这不是必需的，从而可以提供有效的解决方案。我们在验证ACAS Xu网络特性方面的成功表明，该技术在验证实际DNN方面具有很大的潜力。

We presented a novel decision algorithm for solving queries on deep neural networks with ReLU activation functions. The technique is based on extending the simplex algorithm to support the non-convex ReLUs in a way that allows their inputs and outputs to be temporarily inconsistent and then fixed as the algorithm progresses. To guarantee termination, some ReLU connections may need to be split upon — but in many cases this is not required, resulting in an efficient solution. Our success in verifying properties of the ACAS Xu networks indicates that the technique holds much potential for verifying real-world DNNs.

将来，我们计划增加该技术的可扩展性。除了对实施进行工程上的改进外，我们还计划探索应用Reluplex规则的更好策略，并采用先进的冲突分析技术来减少所需的案例分割数量。另一个方向是在不损害性能的情况下提供更好的稳健性保证，例如通过使用精确算术重播浮点解或通过生成可外部检查的正确性证明。最后，我们计划扩展我们的方法，以使用其他种类的层来处理DNN。我们推测，应用于ReLU的机制可以应用于其他分段线性层，例如最大池化层。

In the future, we plan to increase the technique's scalability. Apart from making engineering improvements to our implementation, we plan to explore better strategies for the application of the Reluplex rules, and to employ advanced conflict analysis techniques for reducing the amount of case-splitting required. Another direction is to provide better soundness guarantees without harming performance, for example by replaying floating-point solutions using precise arithmetic [17], or by producing externally-checkable correctness proofs [15]. Finally, we plan to extend our approach to handle DNNs with additional kinds of layers. We speculate that the mechanism we applied to ReLUs can be applied to other piecewise linear layers, such as max-pooling layers.

Acknowledgements. We thank Neal Suchy from the Federal Aviation Administration, Lindsey Kuper from Intel and Tim King from Google for their valuable comments and support. This work was partially supported by a grant from Intel.

References

1. C. Barrett, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Splitting On Demand in SAT Modulo Theories. In *Proc. 13th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, pages 512–526, 2006.
2. C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2009.
3. O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi. Measuring Neural Net Robustness with Constraints. In *Proc. 30th Conf. on Neural Information Processing Systems (NIPS)*, 2016.
4. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars, 2016. Technical Report. `http://arxiv.org/abs/1604.07316`.
5. G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
6. X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proc. 14th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.
7. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
8. I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples, 2014. Technical Report. `http://arxiv.org/abs/1412.6572`.
9. G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
10. X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety Verification of Deep Neural Networks, 2016. Technical Report. `http://arxiv.org/abs/1610.06940`.
11. K. Jarrett, K. Kavukcuoglu, and Y. LeCun. What is the Best Multi-Stage Architecture for Object Recognition? In *Proc. 12th IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2146–2153, 2009.
12. J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer. A Formally Verified Hybrid System for the Next-Generation Airborne Collision Avoidance System. In *Proc. 21st Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 21–36, 2015.
13. K. Julian, J. Lopez, J. Brush, M. Owen, and M. Kochenderfer. Policy Compression for Aircraft Collision Avoidance Systems. In *Proc. 35th Digital Avionics Systems Conf. (DASC)*, pages 1–10, 2016.
14. G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex, 2017. `https://github.com/guykatzz/ReluplexCav2017`.
15. G. Katz, C. Barrett, C. Tinelli, A. Reynolds, and L. Hadarean. Lazy Proofs for DPLL(T)-Based SMT Solvers. In *Proc. 16th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 93–100, 2016.

16. T. King. *Effective Algorithms for the Satisfiability of Quantifier-Free Formulas Over Linear Real and Integer Arithmetic*. PhD Thesis, 2014.
17. T. King, C. Barret, and C. Tinelli. Leveraging Linear and Mixed Integer Programming for SMT. In *Proc. 14th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 139–146, 2014.
18. M. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*, chapter Optimized Airborne Collision Avoidance, pages 259–276. MIT, 2015.
19. M. Kochenderfer and J. Chryssanthacopoulos. Robust Airborne Collision Avoidance through Dynamic Programming. Project Report ATC-371, Massachusetts Institute of Technology, Lincoln Laboratory, 2011.
20. M. Kochenderfer, M. Edwards, L. Espindle, J. Kuchar, and J. Griffith. Airspace Encounter Models for Estimating Collision Risk. *AIAA Journal on Guidance, Control, and Dynamics*, 33(2):487–499, 2010.
21. M. Kochenderfer, J. Holland, and J. Chryssanthacopoulos. Next Generation Airborne Collision Avoidance System. *Lincoln Laboratory Journal*, 19(1):17–33, 2012.
22. A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
23. J. Kuchar and A. Drumm. The Traffic Alert and Collision Avoidance System. *Lincoln Laboratory Journal*, 16(2):277–296, 2007.
24. A. Maas, A. Hannun, and A. Ng. Rectifier Nonlinearities improve Neural Network Acoustic Models. In *Proc. 30th Int. Conf. on Machine Learning (ICML)*, 2013.
25. J. Marques-Silva and K. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.
26. V. Nair and G. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. 27th Int. Conf. on Machine Learning (ICML)*, pages 807–814, 2010.
27. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM (JACM)*, 53(6):937–977, 2006.
28. M. Padberg and G. Rinaldi. A Branch-And-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *IEEE Transactions on Computers*, 33(1):60–100, 1991.
29. L. Pulina and A. Tacchella. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Proc. 22nd Int. Conf. on Computer Aided Verification (CAV)*, pages 243–257, 2010.
30. L. Pulina and A. Tacchella. Challenging SMT Solvers to Verify Neural Networks. *AI Communications*, 25(2):117–135, 2012.
31. M. Riesenhuber and P. Tomaso. Hierarchical Models of Object Recognition in Cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.
32. D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, and S. Dieleman. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.
33. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks, 2013. Technical Report. `http://arxiv.org/abs/1312.6199`.
34. R. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 1996.

附录　　　　补充材料

Appendix: Supplementary Material

I Verifying Properties in DNNs with ReLUs is NP-Complete

Let N be a DNN with ReLUs and let φ denote a property that is a conjunction of linear constraints on the inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ of N, i.e. $\varphi = \varphi_1(\boldsymbol{x}) \wedge \varphi_2(\boldsymbol{y})$. We say that φ *is satisfiable on* N if there exists an assignment α for the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\alpha(\boldsymbol{y})$ is the result of propagating $\alpha(\boldsymbol{x})$ through N and α satisfies φ.

Claim. The problem of determining whether φ is satisfiable on N for a given DNN N and a property φ is NP-complete.

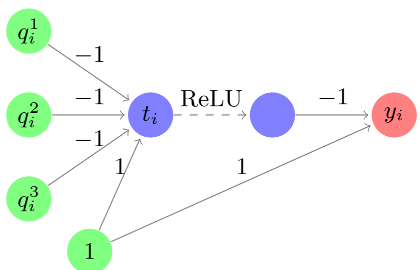
Proof. We first show that the problem is in NP. A satisfiability witness is simply an assignment $\alpha(\boldsymbol{x})$ for the input variables $\boldsymbol{x}$. This witness can be checked by feeding the values for the input variables forward through the network, obtaining the assignment $\alpha(\boldsymbol{y})$ for the output values, and checking whether $\varphi_1(\boldsymbol{x}) \wedge \varphi_2(\boldsymbol{y})$ holds under the assignment α.

Next, we show that the problem is NP-hard, using a reduction from the 3-SAT problem. We will show how any 3-SAT formula ψ can be transformed into a DNN with ReLUs N and a property φ, such that φ is satisfiable on N if and only if ψ is satisfiable.

Let $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ denote a 3-SAT formula over variable set $X = \{x_1, \ldots, x_k\}$, i.e. each C_i is a disjunction of three literals $q_i^1 \vee q_i^2 \vee q_i^3$ where the q's are variables from X or their negations. The question is to determine whether there exists an assignment $a : X \to \{0, 1\}$ that satisfies ψ, i.e. that satisfies all the clauses simultaneously.

For simplicity, we first show the construction assuming that the input nodes take the discrete values 0 or 1. Later we will explain how this limitation can be relaxed, so that the only limitation on the input nodes is that they be in the range $[0, 1]$.

We begin by introducing the *disjunction gadget* which, given nodes $q_1, q_2, q_3 \in \{0, 1\}$, outputs a node y_i that is 1 if $q_1 + q_2 + q_3 \geq 1$ and 0 otherwise. The gadget is shown below for the case that the q_i literals are all variables (i.e. not negations of variables):

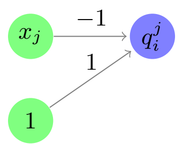


The disjunction gadget can be regarded as calculating the expression

$$y_i = 1 - \max\left(0, 1 - \sum_{j=1}^{3} q_i^j\right)$$

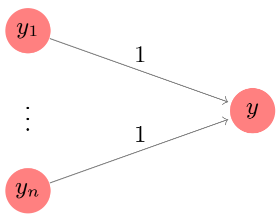
If there is at least one input variable set to 1, y_i will be equal to 1. If all inputs are 0, y_i will be equal to 0. The crux of this gadget is that the ReLU operator allows us to guarantee that even if there are multiple inputs set to 1, the output y_i will still be precisely 1.

In order to handle any negative literals $q_i^j \equiv \neg x_j$, before feeding the literal into the disjunction gadget we first use a *negation gadget*:



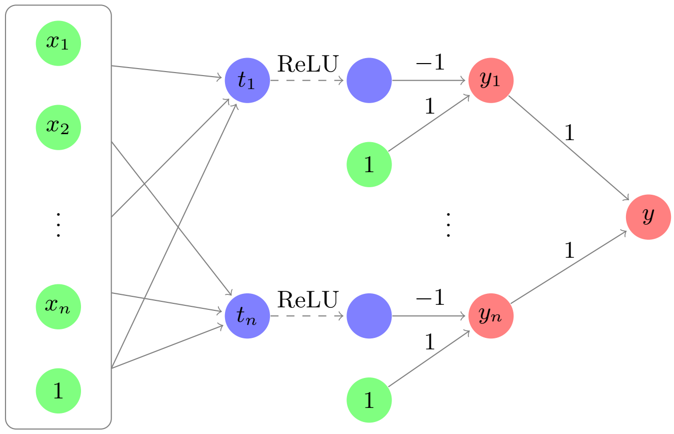
This gadget simply calculates $1 - x_j$, and then we continue as before.

The last part of the construction involves a *conjunction gadget*:



Assuming all nodes $y_1, \ldots, y_n$ are in the domain $\{0, 1\}$, we require that node y be in the range $[n, n]$. Clearly this holds only if $y_i = 1$ for all i.

Finally, in order to check whether all clauses $C_1, \ldots, C_n$ are simultaneously satisfied, we construct a disjunction gadget for each of the clauses (using negation gadgets for their inputs as needed), and combine them using a conjunction gadget:



where the input variables are mapped to each t_i node according to the definition of clause C_i. As we discussed before, node y_i will be equal to 1 if clause C_i is satisfied, and will be 0 otherwise. Therefore, node y will be in the range $[n, n]$ if and only if all clauses are simultaneously satisfied. Consequently, an input assignment $a : X \to \{0, 1\}$ satisfies the input and output constraints on the network if and only if it also satisfies the original ψ, as needed.

The construction above is based on the assumption that we can require that the input nodes take values in the discrete set $\{0, 1\}$, which does not fit our assumption that $\varphi_1(\boldsymbol{x})$ is a conjunction of linear constraints. We show now how this requirement can be relaxed.

Let $\epsilon > 0$ be a very small number. We set the input range for each variable x_i to be $[0, 1]$, but we will ensure that any feasible solution has $x_i \in [0, \epsilon]$ or $x_i \in [1-\epsilon, 1]$. We do this by adding to the network for each x_i an auxiliary gadget that uses ReLU nodes to compute the expression

$$\max(0, \epsilon - x) + \max(0, x - 1 + \epsilon),$$

and requiring that the output node of this gadget be in the range $[0, \epsilon]$. It is straightforward to show that this holds for $x \in [0, 1]$ if and only if $x \in [0, \epsilon]$ or $x \in [1-\epsilon, 1]$.

The disjunction gadgets in our construction then change accordingly. The y_i nodes at the end of each gadget will no longer take just the discrete values $\{0, 1\}$, but instead be in the range $[0, 3 \cdot \epsilon]$ if all inputs were in the range $[0, \epsilon]$, or in the range $[1-\epsilon, 1]$ if at least one input was in the range $[1-\epsilon, \epsilon]$.

If every input clause has at least one node in the range $[1-\epsilon, 1]$ then all y_i nodes will be in the range $[1-\epsilon, 1]$, and consequently y will be in the range $[n(1-\epsilon), n]$. However, if at least one clause does not have a node in the range $[1-\epsilon, 1]$ then y will be smaller than $n(1-\epsilon)$ (for $\epsilon < \frac{1}{n+3}$). Thus, by requiring that $y \in [n(1-\epsilon), n]$, the input and output constraints will be satisfiable on the network if and only if ψ is satisfiable; and the satisfying assignment can be constructed by treating every $x_i \in [0, \epsilon]$ as 0 and every $x_i \in [1-\epsilon, 1]$ as 1. $\square$

II The Reluplex Calculus is Sound and Complete

我们将派生树定义为一棵树，其中每个节点都是一个配置，其子级（如果有）是通过对其应用派生规则之一而获得的。如果通过对D的叶子之一精确地应用一个推导规则从D获得D0，则推导树D推导得出推导树D0。推导是推导树的序列Di，使得D0仅具有一个节点，每个Di推导Di + 1。反驳是指以一棵树结尾的派生，其所有叶子都是UNSAT。见证人是一棵以一棵树结尾的派生，其中至少一棵叶子是SAT。如果ϕ是原子的连接，如果D中的初始树包含从ϕ初始化的配置，则说D是ϕ的派生。如果每当来自ϕ的推导D是反驳或见证时，ϕ分别相应地是不满足或可满足的，那么演算就是声音。如果始终存在从任何ϕ开始的反驳或见证，那么微积分就完成了。

为了证明Reluplex演算是正确的，我们首先证明以下引理：

We define a *derivation tree* as a tree where each node is a configuration whose children (if any) are obtained by applying to it one of the derivation rules. A derivation tree D *derives* a derivation tree D' if D' is obtained from D by applying exactly one derivation rule to one of D's leaves. A *derivation* is a sequence D_i of derivation trees such that D_0 has only a single node and each D_i derives D_{i+1}. A *refutation* is a derivation ending in a tree, all of whose leaves are `UNSAT`. A *witness* is a derivation ending in a tree, at least one of whose leaves is `SAT`. If ϕ is a conjunction of atoms, we say that $\mathcal{D}$ is a derivation from ϕ if the initial tree in $\mathcal{D}$ contains the configuration initialized from ϕ. A calculus is sound if, whenever a derivation $\mathcal{D}$ from ϕ is either a refutation or a witness, ϕ is correspondingly unsatisfiable or satisfiable, respectively. A calculus is *complete* if there always exists either a refutation or a witness starting from any ϕ.

In order to prove that the Reluplex calculus is sound, we first prove the following lemmas:

Lemma 1. *Let $\mathcal{D}$ denote a derivation starting from a derivation tree D_0 with a single node $s_0 = \langle \mathcal{B}_0, T_0, l_0, u_0, \alpha_0, R_0 \rangle$. Then, for every derivation tree D_i appearing in $\mathcal{D}$, and for each node $s = \langle \mathcal{B}, T, l, u, \alpha, R \rangle$ appearing in D_i (except for the distinguished nodes `SAT` and `UNSAT`), the following properties hold:*

(i) an assignment satisfies T_0 if and only if it satisfies T; and
(ii) the assignment α satisfies T (i.e., α satisfies all equations in T).

Proof. The proof is by induction on i. For $i = 0$, the claim holds trivially (recall that α_0 assigns every variable to 0). Now, suppose the claim holds for some i and consider D_{i+1}. D_{i+1} is equivalent to D_i except for the addition of one or more nodes added by the application of a single derivation rule d to some node s with tableau T. Because s appears in D_i, we know by the induction hypothesis that an assignment satisfies T_0 iff it satisfies T and that α satisfies T. Let s' be a new node (not a distinguished node `SAT` or `UNSAT`) with tableau T' and assignment α', introduced by the rule d. Note that d cannot be ReluSuccess or Failure as these introduce only distinguished nodes, and that if d is deriveLowerBound, deriveUpperBound, or ReluSplit, then both the tableau and the assigment are unchanged, so both properties are trivially preserved.

Suppose d is Pivot_1, Pivot_2 or PivotForRelu. For any of these rules, $\alpha' = \alpha$ and $T' = \mathit{pivot}(T, i, j)$ for some i and j. Observe that by definition of the *pivot* operation, the equations of T logically entail those of T' and vice versa, and so they are satisfied by exactly the same assignments. From this observation, both properties follow easily.

The remaining cases are when d is Update, Update_b or Update_f. For these rules, $T' = T$, from which property (i) follows trivially. For property (ii), we first note that $\alpha' = \mathit{update}(\alpha, x_i, \delta)$ for some i and δ. By definition of the *update* operation, because α satisfied the equations of T, α' continues to satisfy these equations and so (because $T' = T$) α' also satisfies T'. ∎

Lemma 2. *Let $\mathcal{D}$ denote a derivation starting from a derivation tree D_0 with a single node $s_0 = \langle \mathcal{B}_0, T_0, l_0, u_0, \alpha_0, R_0 \rangle$. If there exists an assignment α^* (not necessarily α_0) such that α^* satisfies T_0 and $l_0(x_i) \leq \alpha^*(x_i) \leq u_0(x_i)$ for all i, then for each derivation tree D_i appearing in $\mathcal{D}$ at least one of these two properties holds:*

(i) D_i has a `SAT` *leaf.*
(ii) D_i has a leaf $s = \langle \mathcal{B}, T, l, u, \alpha, R \rangle$ (that is not a distinguished node `SAT` *or* `UNSAT`*) such that $l(x_i) \leq \alpha^*(x_i) \leq u(x_i)$ for all i.*

Proof. The proof is again by induction on i. For $i = 0$, property (ii) holds trivially. Now, suppose the claim holds for some i and consider D_{i+1}. D_{i+1} is equivalent to D_i except for the addition of one or more nodes added by the application of a single derivation rule d to a leaf s of D_i.

Due to the induction hypothesis, we know that D_i has a leaf $\bar{s}$ that is either a `SAT` leaf or that satisfies property (ii). If $\bar{s} \neq s$, then $\bar{s}$ also appears in D_{i+1}, and the claim holds. We will show that the claim also holds when $\bar{s} = s$. Because none of the derivation rules can be applied to a `SAT` or `UNSAT` node, we know that node s is not a distinguished `SAT` or `UNSAT` node, and we denote $s = \langle \mathcal{B}, T, l, u, \alpha, R \rangle$.

If d is ReluSuccess, D_{i+1} has a `SAT` leaf and property (i) holds. Suppose d is Pivot$_1$, Pivot$_2$, PivotForRelu, Update, Update$_b$ or Update$_f$. In any of these cases, node s has a single child in D_{i+1}, which we denote $s' = \langle \mathcal{B}', T', l', u', \alpha', R' \rangle$. By definition of these derivation rules, $l'(x_j) = l(x_j)$ and $u'(x_j) = u(x_j)$ for all j. Because node s satisfies property (ii), we get that s' is a leaf that satisfies property (ii), as needed.

Suppose that d is ReluSplit, applied to a pair $\langle x_i, x_j \rangle \in R$. Node s has two children in D_{i+1}: a state s^+ in which the lower bound for x_i is 0, and a state s^- in which the upper bound for x_i is 0. All other lower and upper bounds in s^+ and s^- are identical to those of s. It is straightforward to see that if $\alpha^*(x_i) \geq 0$ then property (ii) holds for s^+, and if $\alpha^*(x_i) \leq 0$ then property (ii) holds for s^-. Either way, D_{i+1} has a leaf for which property (ii) holds, as needed.

Next, consider the case where d is deriveLowerBound (the deriveUpperBound case is symmetrical and is omitted). Node s has a single child in D_{i+1}, which we denote $s' = \langle \mathcal{B}', T', l', u', \alpha', R' \rangle$. Let x_i denote the variable to which deriveLowerBound was applied. By definition, $l'(x_i) \geq l(x_i)$, and all other variable bounds are unchanged between s and s'. Thus, it suffices to show that $\alpha^*(x_i) \geq l'(x_i)$. Because α^* satisfies T_0, it follows from Lemma 1 that it satisfies T. By the induction hypothesis, $l(x_j) \leq \alpha^*(x_j) \leq u(x_j)$ for all j. The fact that $\alpha^*(x_i) \geq l'(x_i)$ then follows directly from the guard condition of deriveLowerBound.

The only remaining case is when d is the Failure rule. We explain why this case is impossible. Suppose towards contradiction that in node s the Failure rule is applicable to variable x_i, and suppose (without loss of generality) that $\alpha(x_i) < l(x_i)$. By the inductive hypothesis, we know that $l(x_j) \leq \alpha^*(x_j) \leq u(x_j)$ for all j, and by Lemma 1 we know that α^* satisfies T. Consequently, there must be a variable x_k such that $(T_{i,k} > 0 \wedge \alpha(x_k) < \alpha^*(x_k))$, or $(T_{i,k} < 0 \wedge \alpha(x_k) >$

$\alpha^*(x_k)$). But because all variables under α^* are within their bounds, this means that $slack^+(x_i) \neq \emptyset$, which is contradictory to the fact that the Failure rule was applicable in s. □

Lemma 3. *Let $\mathcal{D}$ denote a derivation starting from a derivation tree D_0 with a single node $s_0 = \langle \mathcal{B}_0, T_0, l_0, u_0, \alpha_0, R_0 \rangle$. Then, for every derivation tree D_i appearing in $\mathcal{D}$, and for each node $s = \langle \mathcal{B}, T, l, u, \alpha, R \rangle$ appearing in D_i (except for the distinguished nodes SAT and UNSAT), the following properties hold:*

(i) $R = R_0$; and
(ii) $l(x_i) \geq l_0(x_i)$ and $u(x_i) \leq u_0(x_i)$ for all i.

Proof. Property (i) follows from the fact that none of the derivation rules (except for ReluSuccess and Failure) changes the set R. Property (ii) follows from the fact that the only rules (except for ReluSuccess and Failure) that update lower and upper variable bounds are deriveLowerBound and deriveUpperBound, respectively, and that these rules can only increase lower bounds or decrease upper bounds.

We are now ready to prove that the Reluplex calculus is sound and complete.

Claim. The Reluplex calculus is sound.

Proof. We begin with the satisfiable case. Let $\mathcal{D}$ denote a witness for ϕ. By definition, the final tree D in $\mathcal{D}$ has a SAT leaf. Let $s_0 = \langle \mathcal{B}_0, T_0, l_0, u_0, \alpha_0, R_0 \rangle$ denote the initial state of D_0 and let $s = \langle \mathcal{B}, T, l, u, \alpha, R \rangle$ denote a state in D in which the ReluSuccess rule was applied (i.e., a predecessor of a SAT leaf).

By Lemma 1, α satisfies T_0. Also, by the guard conditions of the ReluSuccess rule, $l(x_i) \leq \alpha(x_i) \leq u(x_i)$ for all i. By property (ii) of Lemma 3, this implies that $l_0(x_i) \leq \alpha(x_i) \leq u_0(x_i)$ for all i. Consequently, α satisfies every linear inequality in ϕ.

Finally, we observe that by the conditions of the ReluSuccess rule, α satisfies all ReLU constraints of s. From property (i) of Lemma 3, it follows that α also satisfies the ReLU constraints of s_0, which are precisely the ReLU constraints in ϕ. We conclude that α satisfies every constraint in ϕ, and hence ϕ is satisfiable, as needed.

For the unsatisfiable case, it suffices to show that if ϕ is satisfiable then there cannot exist a refutation for it. This is a direct result of Lemma 2: if ϕ is satisfiable, then there exists an assignment α^* that satisfies the initial tableau T_0, and for which all variables are within bounds. Hence, Lemma 2 implies that any derivation tree in any derivation $\mathcal{D}$ from ϕ must have a leaf that is not the distinguished UNSAT leaf. It follows that there cannot exist a refutation for ϕ. □

Claim. The Reluplex calculus is complete.

Proof. Having shown that the Reluplex calculus is sound, it suffices to show a strategy for deriving a witness or a refutation for every ϕ within a finite number of steps. As mentioned in Section 3, one such strategy involves two steps: (i) Eagerly apply the ReluSplit rule until it no longer applies; and (ii) For every leaf of the resulting derivation tree, apply the simplex rules Pivot_1, Pivot_2, Update, and $\mathsf{Failure}$, and the Reluplex rule $\mathsf{ReluSuccess}$, in a way that guarantees a `SAT` or an `UNSAT` configuration is reached within a finite number of steps.

Let D denote the derivation tree obtained after step (i). In every leaf s of D, all ReLU connections have been eliminated, meaning that the variable bounds force each ReLU connection to be either active or inactive. This means that every such s can be regarded as a pure simplex problem, and that any solution to that simplex problem is guaranteed to satisfy also the ReLU constraints in s.

The existence of a terminating simplex strategy for deciding the satisfiability of each leaf of D follows from the completeness of the simplex calculus [34]. One such widely used strategy is *Bland's Rule* [34]. We observe that although the simplex $\mathsf{Success}$ rule does not exist in Reluplex, it can be directly substituted with the $\mathsf{ReluSuccess}$ rule. This is so because, having applied the $\mathsf{ReluSplit}$ rule to completion, any assignment that satisfies the variable bounds in s also satisfies the ReLU constraints in s.

It follows that for every ϕ we can produce a witness or a refutation, as needed. □

III A Reluplex Strategy that Guarantees Termination

As discussed in Section 6, our strategy for applying the Reluplex rules was to repeatedly fix any out-of-bounds violations first (using the original simplex rules), and only afterwards to correct any violated ReLU constraints using the Update_b, Update_f and $\mathsf{PivotForRelu}$ rules. If correcting a violated ReLU constraint introduced new out-of-bounds violations, these were again fixed using the simplex rules, and so on.

As mentioned above, there exist well known strategies for applying the simplex rules in a way that guarantees that within a finite number of steps, either all variables become assigned to values within their bounds, or the $\mathsf{Failure}$ rule is applicable (and is applied) [34]. By using such a strategy for fixing out-of-bounds violations, and by splitting on a ReLU pair whenever the Update_b, Update_f or $\mathsf{PivotForRelu}$ rules are applied to it more some fixed number of times, termination is guaranteed.

IV Under-Approximations

Under-approximation can be integrated into the Reluplex algorithm in a straightforward manner. Consider a variable x with lower and upper bounds $l(x)$ and $u(x)$, respectively. Since we are searching for feasible solutions for which $x \in [l(x), u(x)]$, an under-approximation can be obtained by restricting this range, and only considering feasible solutions for which $x \in [l(x) + \epsilon, u(x) - \epsilon]$ for some small $\epsilon > 0$.

Applying under-approximations can be particularly useful when it effectively eliminates a ReLU constraint (consequently reducing the potential number of case splits needed). Specifically, observe a ReLU pair $x^f = \mathrm{ReLU}(x^b)$ for which we have $l(x^b) \geq -\epsilon$ for a very small positive ϵ. We can under-approximate this range and instead set $l(x^b) = 0$; and, as previously discussed, we can then fix the ReLU pair to the active state. Symmetrical measures can be employed when learning a very small upper bound for x^f, in this case leading to the ReLU pair being fixed in the inactive state.

Any feasible solution that is found using this kind of under-approximation will be a feasible solution for the original problem. However, if we determine that the under-approximated problem is infeasible, the original may yet be feasible.

V Encoding ReLUs for SMT and LP Solvers

我们演示了ReLU节点的编码，我们将其用于使用SMT和LP解算器进行评估。 令y = ReLU (x) 。 在我们测试的所有SMT求解器都使用的SMTLIB格式中，ReLU使用if-then-else构造进行编码：(assert(= y(ite(>= x 0) x 0)))

We demonstrate the encoding of ReLU nodes that we used for the evaluation conducted using SMT and LP solvers. Let $y = \mathrm{ReLU}(x)$. In the SMTLIB format, used by all SMT solvers that we tested, ReLUs were encoded using an if-then-else construct:

```
(assert (= y (ite (>= x 0) x 0)))
```

以LP格式使用混合整数编程进行编码。使用Gurobi的内置布尔类型，我们为每个ReLU连接定义了一对布尔变量bon和boff，并使用它们对连接的两种可能状态进行编码。 假设M是一个非常大的正常数，我们使用以下断言：

In LP format this was encoded using mixed integer programming. Using Gurobi's built-in Boolean type, we defined for every ReLU connection a pair of Boolean variables, $\mathrm{b_{on}}$ and $\mathrm{b_{off}}$, and used them to encode the two possible states of the connection. Taking M to be a very large positive constant, we used the following assertions:

$$
\begin{aligned}
&\mathtt{b_{on}} + \mathtt{b_{off}} = 1\\
&\mathtt{y} >= 0\\
&\mathtt{x} - \mathtt{y} - \mathtt{M{*}b_{off}} <= 0\\
&\mathtt{x} - \mathtt{y} + \mathtt{M{*}b_{off}} >= 0\\
&\mathtt{y} - \mathtt{M{*}b_{on}} <= 0\\
&\mathtt{x} - \mathtt{M{*}b_{on}} <= 0
\end{aligned}
$$

当bon = 1和boff = 0时，ReLU连接处于活动状态； 否则，当bon = 0且boff = 1时，它处于非活动状态。

When $\mathrm{b_{on}}= 1$ and $\mathrm{b_{off}}= 0$, the ReLU connection is in the active state; and otherwise, when $\mathrm{b_{on}}= 0$ and $\mathrm{b_{off}}= 1$, it is in the inactive state.

在活动情况下，由于boff = 0，因此第三和第四方程式暗示x = y（观察到y总是非负值）。 M非常大，可以看作是∞；因此，由于bon = 1，因此最后两个方程仅表示x，y≤∞，因此对解无限制。

In the active case, because $\mathrm{b_{off}} = 0$ the third and fourth equations imply that $x = y$ (observe that y is always non-negative). M is very large, and can be regarded as ∞; hence, because $\mathrm{b_{on}}= 1$, the last two equations merely imply that $x, y \leq \infty$, and so pose no restriction on the solution.

在非活动情况下，bon = 0，因此后两个方程强制y = 0且x≤0。在这种情况下，boff = 1，因此第三个方程和第四个方程对解没有任何限制。

In the inactive case, $\mathrm{b_{on}} = 0$, and so the last two equations force $y = 0$ and $x \leq 0$. In this case $\mathrm{b_{off}}= 1$ and so the third and fourth equations pose no restriction on the solution.

VI Formal Definitions for Properties $\phi_1,\ldots,\phi_{10}$

The units for the ACAS Xu DNNs' inputs are:

– ρ: feet.
– θ, ψ: radians.
– $v_{\text{own}}, v_{\text{int}}$: feet per second.
– τ: seconds.

逆时针

θ and ψ are measured counter clockwise, and are always in the range $[-\pi, \pi]$.

In line with the discussion in Section 5, the family of 45 ACAS Xu DNNs are indexed according to the previous action a_{prev} and time until loss of vertical separation τ. The possible values are for these two indices are:

1. a_{prev}: [Clear-of-Conflict, weak left, weak right, strong left, strong right].
2. τ: $[0, 1, 5, 10, 20, 40, 60, 80, 100]$.

We use $N_{x,y}$ to denote the network trained for the x-th value of a_{prev} and y-th value of τ. For example, $N_{2,3}$ is the network trained for the case where $a_{\text{prev}} =$ weak left and $\tau = 5$. Using this notation, we now give the formal definition of each of the properties $\phi_1, \ldots, \phi_{10}$ that we tested.

Property ϕ_1.

如果入侵者距离远且比自身慢得多，则COC警告的分数将始终低于某个固定阈值。

– Description: If the intruder is distant and is significantly slower than the ownship, the score of a COC advisory will always be below a certain fixed threshold.
– Tested on: all 45 networks.
– Input constraints: $\rho \geq 55947.691$, $v_{\text{own}} \geq 1145$, $v_{\text{int}} \leq 60$.
– Desired output property: the score for COC is at most 1500. 阈值 1500

Property ϕ_2.

如果入侵者距离远且比自身慢得多，则COC警告的分数将永远不会最高。

– Description: If the intruder is distant and is significantly slower than the ownship, the score of a COC advisory will never be maximal.
– Tested on: $N_{x,y}$ for all $x \geq 2$ and for all y.
– Input constraints: $\rho \geq 55947.691$, $v_{\text{own}} \geq 1145$, $v_{\text{int}} \leq 60$.
– Desired output property: the score for COC is not the maximal score.

Property ϕ_3.

如果入侵者直行向前，并且正朝着自己的方向移动，那么COC的分数将不会是最小的。

– Description: If the intruder is directly ahead and is moving towards the ownship, the score for COC will not be minimal.
– Tested on: all networks except $N_{1,7}$, $N_{1,8}$, and $N_{1,9}$.
– Input constraints: $1500 \leq \rho \leq 1800$, $-0.06 \leq \theta \leq 0.06$, $\psi \geq 3.10$, $v_{\text{own}} \geq 980$, $v_{\text{int}} \geq 960$.
– Desired output property: the score for COC is not the minimal score.

Property ϕ_4.

- Description: If the intruder is directly ahead and is moving away from the ownship but at a lower speed than that of the ownship, the score for COC will not be minimal.
- Tested on: all networks except $N_{1,7}$, $N_{1,8}$, and $N_{1,9}$.
- Input constraints: $1500 \leq \rho \leq 1800$, $-0.06 \leq \theta \leq 0.06$, $\psi = 0$, $v_{\text{own}} \geq 1000$, $700 \leq v_{\text{int}} \leq 800$.
- Desired output property: the score for COC is not the minimal score.

Property ϕ_5.

- Description: If the intruder is near and approaching from the left, the network advises "strong right".
- Tested on: $N_{1,1}$.
- Input constraints: $250 \leq \rho \leq 400$, $0.2 \leq \theta \leq 0.4$, $-3.141592 \leq \psi \leq -3.141592 + 0.005$, $100 \leq v_{\text{own}} \leq 400$, $0 \leq v_{\text{int}} \leq 400$.
- Desired output property: the score for "strong right" is the minimal score.

Property ϕ_6.

- Description: If the intruder is sufficiently far away, the network advises COC.
- Tested on: $N_{1,1}$.
- Input constraints: $12000 \leq \rho \leq 62000$, $(0.7 \leq \theta \leq 3.141592) \vee (-3.141592 \leq \theta \leq -0.7)$, $-3.141592 \leq \psi \leq -3.141592 + 0.005$, $100 \leq v_{\text{own}} \leq 1200$, $0 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for COC is the minimal score.

Property ϕ_7.

- Description: If vertical separation is large, the network will never advise a strong turn.
- Tested on: $N_{1,9}$.
- Input constraints: $0 \leq \rho \leq 60760$, $-3.141592 \leq \theta \leq 3.141592$, $-3.141592 \leq \psi \leq 3.141592$, $100 \leq v_{\text{own}} \leq 1200$, $0 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the scores for "strong right" and "strong left" are never the minimal scores.

Property ϕ_8.

- Description: For a large vertical separation and a previous "weak left" advisory, the network will either output COC or continue advising "weak left".
- Tested on: $N_{2,9}$.
- Input constraints: $0 \leq \rho \leq 60760$, $-3.141592 \leq \theta \leq -0.75 \cdot 3.141592$, $-0.1 \leq \psi \leq 0.1$, $600 \leq v_{\text{own}} \leq 1200$, $600 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for "weak left" is minimal or the score for COC is minimal.

Property ϕ_9.

- Description: Even if the previous advisory was "weak right", the presence of a nearby intruder will cause the network to output a "strong left" advisory instead.
- Tested on: $N_{3,3}$.
- Input constraints: $2000 \leq \rho \leq 7000$, $-0.4 \leq \theta \leq -0.14$, $-3.141592 \leq \psi \leq -3.141592 + 0.01$, $100 \leq v_{\text{own}} \leq 150$, $0 \leq v_{\text{int}} \leq 150$.
- Desired output property: the score for "strong left" is minimal.

Property ϕ_{10}.

- Description: For a far away intruder, the network advises COC.
- Tested on: $N_{4,5}$.
- Input constraints: $36000 \leq \rho \leq 60760$, $0.7 \leq \theta \leq 3.141592$, $-3.141592 \leq \psi \leq -3.141592 + 0.01$, $900 \leq v_{\text{own}} \leq 1200$, $600 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for COC is minimal.