软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007344] [CSTR: 32375.14.jos.007344]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

神经网络的增量验证*

刘宗鑫[1,2,3], 迟智名[1,2,3], 赵梦宇[1,2,3], 黄承超[5], 黄小炜[4], 蔡少伟[1,2,3], 张立军[1,2,3], 杨鹏飞[1,2]

[1](基础软件与系统重点实验室 (中国科学院 软件研究所), 北京 100190)

[2](计算机科学国家重点实验室 (中国科学院 软件研究所), 北京 100190)

[3](中国科学院大学, 北京 100049)

[4](University of Liverpool, Liverpool L69 3BX, UK)

[5](中国科学院大学南京学院, 南京 211135)

通信作者: 杨鹏飞, E-mail: ypfbest001@swu.edu.cn

摘　要: 约束求解是验证神经网络的基础方法. 在人工智能安全领域, 为了修复或攻击等目的, 常需要对神经网络的结构和参数进行修改. 面对此类需求, 提出神经网络的增量验证问题, 旨在判断修改后的神经网络是否仍保持安全性质. 针对这类问题, 基于 *Reluplex* 框架提出了一种增量可满足性模理论算法 DeepInc. 该算法利用旧求解过程中关键计算格局的特征, 启发式地检查关键计算格局是否适用于证明修改后的神经网络. 实验结果显示, DeepInc 的效率在大多数情况下都优于 Marabou. 此外, 即使与最先进的验证工具 α, β-CROWN 相比, 对于修改前后均未满足预设安全性质的网络, DeepInc 也实现了显著的加速.

关键词: 可满足性模理论; 深度神经网络; 增量约束求解; 局部鲁棒; 形式化验证

中图法分类号: TP311

中文引用格式: 刘宗鑫, 迟智名, 赵梦宇, 黄承超, 黄小炜, 蔡少伟, 张立军, 杨鹏飞. 神经网络的增量验证. 软件学报. http://www.jos.org.cn/1000-9825/7344.htm

英文引用格式: Liu ZX, Chi ZM, Zhao MY, Huang CC, Huang XW, Cai SW, Zhang LJ, Yang PF. Incremental Verification for Neural Network. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7344.htm

Incremental Verification for Neural Network

LIU Zong-Xin[1,2,3], CHI Zhi-Ming[1,2,3], ZHAO Meng-Yu[1,2,3], HUANG Cheng-Chao[5], HUANG Xiao-Wei[4], CAI Shao-Wei[1,2,3], ZHANG Li-Jun[1,2,3], YANG Peng-Fei[1,2]

[1](Key Laboratory of System Software (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[2](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100049, China)

[4](University of Liverpool, Liverpool L69 3BX, UK)

[5](University of Chinese Academy of Sciences, Nanjing, Nanjing 211135, China)

Abstract: Constraint solving is a fundamental approach for verifying deep neural network (DNN). In the field of AI safety, DNNs often undergo modifications in their structure and parameters for purposes such as repair or attack. In such scenarios, the problem of incremental DNN verification is proposed, which aims to determine whether a safety property still holds after the DNN has been modified. To address this, an incremental satisfiability modulo theory (SMT) algorithm based on the *Reluplex* framework is presented. The proposed algorithm, DeepInc, leverages the key features of the configurations from the previous solving procedure, heuristically checking whether these features can be applied to prove the correctness of the modified DNN. Experimental results demonstrate that DeepInc outperforms Marabou in

* 基金项目: 中国科学院基础研究青年团队计划 (YSBR-040); 中国科学院软件研究所新培育方向项目 (ISCAS-PYFX-202201); 中国科学院软件研究所基础研究项目 (ISCAS-JCZD-202302)
本文由“形式化方法与应用”专题特约编辑陈明帅研究员、田聪教授、熊英飞副教授推荐.
收稿时间: 2024-08-23; 修改时间: 2024-10-14; 采用时间: 2024-11-27; jos 在线出版时间: 2024-12-10

terms of efficiency in most cases. Moreover, for cases where the safety property is violated both before and after modification, DeepInc achieves significantly faster performance, even when compared to the state-of-the-art verifier α, β-CROWN.

Key words: satisfiability module theory; deep neural network (DNN); incremental constraint solving; local robustness; formal verification

过去几年间, 神经网络在计算机视觉, 自然语言处理等多个领域取得了革命性的进展. 然而, 神经网络在安全攸关领域的应用仍受到其可靠性的限制. 特别是, 神经网络面临对抗攻击时极为脆弱, 攻击者可以对输入数据进行微小的修改致使模型做出错误的决策[1]. 这种脆弱性引起了人们对于在自动驾驶, 医疗诊断等安全攸关领域中应用神经网络的担忧.

形式化验证使用数学方法严格证明系统或程序的正确性与可靠性. 在本工作中, 我们聚焦于验证神经网络的安全性质, 即确定在给定输入约束下神经网络的输出是否满足预定义的安全行为. 为实现这一目标, 我们采用约束求解技术, 这是一种在神经网络验证领域中被广泛使用的基础技术. 在进行约束求解时, 神经网络的行为和安全性质的否定被编码成约束. 这些约束通常以实数变量不等式的形式出现. 随后, 使用约束求解器来确定是否存在一组对这些变量的赋值能够满足这些约束. 如果找到了满足约束的赋值, 求解器将返回 SAT, 表示存在违反安全性质的反例 (性质不成立). 反之, 如果没有找到满足约束的赋值, 求解器将返回 UNSAT, 表示不存在违反安全性质的赋值 (性质成立). 2017 年, 针对以线性整流函数 (rectified linear unit, *ReLU*) 作为激活函数的神经网络, Katz 等人[2]和 Ehlers[3]分别提出了的基于可满足性模理论 (satisfiability modulo theories, SMT) 的求解器 *Reluplex* 和 Planet. 此后, Marabou[4]作为 *Reluplex* 的优化版本推出, 展示了更佳的实验性能. 基于抽象解释的方法, 如 AI^2 [5]和 *DeepPoly*[6]等, 能够有效地提供约束求解问题语义的上近似, 因此在进行约束求解时常被用于初始化变量的边界.

增量约束求解是在约束发生微小变化时可以高效求解其可满足性问题的技术. 增量约束求解的核心思想是利用旧求解过程的信息, 即确定在旧求解过程中所做的推断是否可以应用于求解新的约束. 由于约束的变化相对较小, 可以快速检查旧求解过程中的推断是否能够保留. 通常情况下大部分推断能够应用于新的求解过程. 与从头开始解决整个问题相比, 这种方法显著降低了计算开销. 增量约束求解已广泛应用于各种形式方法中, 例如有界模型检验中的增量 SAT 求解[7−9]和组合等效性检查[10]. 该技术还被用于线性规划的敏感性分析[11−15]. 增量约束求解可以有效处理约束的变化, 并提高约束求解算法的整体性能.

神经网络的权重或结构的微小变化催生了对神经网络进行增量验证的需求. 增量验证的一个典型的应用场景是验证神经网络修复[16]后的网络. 神经网络修复的目标是纠正神经网络非预期的行为, 通常情况下, 这需要对神经网络的权重或结构进行微小改动. 在利用现有的验证工具来评估修复效果时, 需要对被微调过的网络进行验证, 因此采用增量约束求解技术显得尤为适宜. 增量约束求解另一个可能的应用是用于神经网络验证中的反例引导的抽象细化 (counterexample-guided abstraction refinement, CEGAR) 框架[17−19], 该框架首先创建神经网络的抽象, 并对抽象后的网络进行验证. 如果验证过程发现假反例, 则迭代地精化抽象. 每个精化步骤都涉及对神经网络的结构和权重进行微小的更改. 例如, Elboher 等人[18]通过合并神经元与其对应权重进行抽象, 通过拆分合并的神经元进行精化. 在这种情况下, 网络的结构和权重都发生改变, 因此可以使用增量约束求解高效的地验证精化后的神经网络. 此外, 增量情况也可能适用于对抗性训练[20]和后门攻击[21]等场景.

本文专注于神经网络验证中的增量约束求解问题. 与传统的增量 SAT 求解或线性规划的灵敏度分析不同, 神经网络增量验证问题中的变化主要涉及权重的定量变化. 换言之, 虽然神经网络增量验证问题中发生变化的约束数量可能很大, 甚至所有约束都可能受到微小扰动的影响, 但权重变化的绝对值之和仍然微小. 这一根本差异使得神经网络的增量验证问题与传统增量约束求解问题存在本质的区别.

鉴于这些挑战, 我们基于 *Reluplex* 框架提出一种针对神经网络增量验证需求的增量约束求解方法. 我们的算法考虑了神经网络权重的定量变化, 并能够高效地处理大量变化的约束. 具体来说, 我们仅考虑神经网络在权重上进行修改, 而结构保持不变的情况. 该情况下, 原始神经网络和修改后的神经网络之间存在一一对应的关系, 这使我们能在修改后的神经网络中模拟旧验证过程中直接得出某一分支验证结果的计算格局. 在模拟中, 我们提取原始计算格局中的关键特征, 包括分支断言, 基础变量集和直接推断出 UNSAT 结果的线性等式的位置. 通过计算断言的合取并利用修改后的神经网络的语义, 我们可以有效地引导搜索树来模拟这些断言. 加强关键线性等式中变

量的数值边界是检查修改后的神经网络中旧 UNSAT 证明是否依然适用的关键, 因此我们采用线性规划并使用线性抽象技术编码不确定的 *ReLU* 关系[22], 这使我们能够得到更紧的数值边界并加速被修改神经网络的证明. 针对旧求解过程中找到反例的情况, 我们启发式地搜索对应的分支及其附近分支, 以在修改后的神经网络中快速找到反例. 通过利用这些技术, 我们的方法能够在增量场景中高效, 准确地验证神经网络的性质.

本工作的主要贡献可以总结如下.

(1) 我们在神经网络验证问题中引入增量约束求解的概念, 并提出基于 *Reluplex* 框架的增量 SMT 算法. 该算法在扰动后的神经网络上模拟旧验证过程中的搜索树上叶子结点处的关键证明, 利用单纯形法的机制快速检查模拟的证明是否能复用, 从而在神经网络权重被更改但结构保持不变时进行高效地增量验证.

(2) 我们实现增量 SMT 求解器 DeepInc, 该求解器基于 SMT 验证工具 Marabou. 实验表明, 在 ACAS Xu 和 MNIST 网络上, DeepInc 分别在 81% 和 57.3% 的验证任务中优于 Marabou. 增量验证求解器 DeepInc 的性能与神经网络的扰动幅度有很大关系, 在扰动幅度不大 (权重值变化不超过 5%) 时, 我们的方法有显著的效率提升. 同时我们分析了增量求解效率较低的情况, 并通过实验表明即使是微小的修改也可能从根本改变原始神经网络的行为, 从而增加增量 SMT 求解的难度.

(3) 我们将 DeepInc 与目前最先进的工具 α, β-CROWN 在困难的验证任务上进行比较. 在寻找反例方面, DeepInc 的性能优于 α, β-CROWN, 表现出强劲的竞争力. 此外, 与同期工作 IVAN[23]的相比, DeepInc 在困难性质上的表现始终优于 IVAN. 这些结果表明, 基于 SMT 的 DeepInc 在求解具有挑战性的验证问题上具有潜在优势.

1 基础知识

在本节中, 我们将回顾深度神经网络验证的基础定义.

深度神经网络由一系列层组成, 从输入层开始, 经过若干隐藏层, 最终到达输出层. 每一层中都存在若干神经元, 每个神经元代表一个实数变量. 除输入层外, 神经元的值是通过前一层的神经元的函数变换获得的. 我们将神经网络建模为函数 $f:\mathbb{R}^m\to\mathbb{R}^n$, 该函数是层与层之间函数的复合. 这些函数包括仿射函数和非线性激活函数. 仿射函数的形式为 $y=Wx+b$, 其中 W 和 b 分别是常数实数矩阵和向量, 称为权重和偏置. 在本工作中, 我们只考虑 *ReLU* 激活函数 $ReLU(x)=\max(0,x)$, 其中 $x\in\mathbb{R}$. 对于 *ReLU* 关系 $x_j=ReLU(x_i)$, 如果 x_i 的所有可能的取值均为非负/非正, 我们称神经元 x_i 被确定激活/非激活. 如果 x_i 既不能被确定激活也不是被确定非激活, 我们称其为不确定状态.

在形式验证中, 安全性质意味着不发生任何不期望的事件. 在神经网络验证的背景下, 安全性质通常要求神经网络对给定区域内的所有输入都应表现出正确的行为. 我们可以对安全性在进行如下定义.

定义 1 (神经网络的安全性质). 神经网络的安全性质是一个三元组 (f,X,P), 其中 $f:\mathbb{R}^m\to\mathbb{R}^n$ 是神经网络, $X\subseteq\mathbb{R}^m$ 和 $P\subseteq\mathbb{R}^n$ 分别是输入和输出的集合. 当且仅当对所有 $x\in X$, 有 $f(x)\in P$ 时, 性质 (f,X,P) 成立. 违反性质 P 的输入 $x^*\in X$ 满足 $f(x^*)\notin P$, 被称为性质 (f,X,P) 的反例. 神经网络验证问题是判断给定的性质 (f,X,P) 是否成立.

局部鲁棒性也是安全性质的一种. 对于分类神经网络 $f:\mathbb{R}^m\to\mathbb{R}^n$, 定义 $C_f(x)=\arg\max_{1\leqslant i\leqslant n}f(x)_i$ 为 f 的分类. 如果对于输入 x_0 的邻域 $B(x_0)$ 中的所有 x, 都有 $C_f(x)=C_f(x_0)$, 则称分类神经网络 f 在给定邻域中是局部鲁棒的. 该性质可以写成安全性质 (f,X,P), 其中 $X=B(x_0)$ 且 $P=\{y\in\mathbb{R}^n\mid\arg\max_i y_i=C_f(x_0)\}$.

约束求解是神经网络验证的基本方式. 给定一个安全性质 (f,X,P), 我们将输入集合 X, 构成神经网络 f 的所有函数, 以及性质 P 的否定编码为等式或不等式, 并确定这些约束的合取是否可以被满足. 如果这些约束的合取可以被满足, 则必定存在违反该性质的反例; 否则, 该安全性质成立.

2 适用于神经网络验证的增量可满足模理论

在本节中, 我们将介绍神经网络验证的增量 SMT 求解问题, 并通过一个示例展示我们的算法.

2.1 *Reluplex* 算法

Reluplex[2]是一个基于 SMT 的神经网络验证算法. *Reluplex* 求解过程中, 会对当前的计算格局进行管理. 当计

算格局未终止时, 它维护线性等式、*ReLU* 关系、数值边界和所有变量的当前赋值. 形式化地, *Reluplex* 计算格局可以定义如下.

定义 2 (*Reluplex* 计算格局). 在给定变量集 $\mathcal{X}$ 上的 *Reluplex* 计算格局 C 是 SAT, UNSAT, 或一个六元组 $(\mathcal{B},T,R,l,u,\alpha)$, 其中,

- $\mathcal{B}\subseteq\mathcal{X}$ 是基础变量 (basic variable) 的集合;
- T 是解表 (tableau), 对于每个 $x_i\in\mathcal{B}$, 存储一个形式为 $x_i=\sum_{x_j\notin\mathcal{B}}a_{ij}x_j$ 的约束;
- $R\subseteq\mathcal{X}\times\mathcal{X}$ 是 *ReLU* 对的集合;
- $l,u:\mathcal{X}\to\mathbb{R}$ 分别是每个变量的下界和上界;
- $\alpha:\mathcal{X}\to\mathbb{R}$ 是每个变量的赋值.

给定性质 (f,X,P), 我们引入松弛变量来建立解表, 并调用 *DeepPoly*[6]计算变量的数值边界来初始化计算格局. *DeepPoly* 是一种基于抽象解释计算神经元在给定输入约束下的上下界的高效算法. 该算法对每个神经元维护一组线性的符号上下界和数值上下界. 其中符号上下界是通过对神经元进行抽象 (三角形松弛) 得到的神经元输出范围的线性边界; 数值上下界由符号上下界具体化得到. 具体化是一个迭代过程, 需要不断将当前符号上下界表达式中的变量替换为前一层的变量, 直到得到当前符号上下界关于输入的表达式, 并带入输入的上下界得到数值上下界.

当求解过程处于计算格局 $(\mathcal{B},T,R,l,u,\alpha)$ 时, *Reluplex* 会通过不断调整变量的赋值 α 以匹配解表 T 和 *ReLU* 关系 R 进行局部搜索. 如果当前计算格局 $(\mathcal{B},T,R,l,u,\alpha)$ 中的赋值能够使所有约束被满足, 即:

- 对于解表 T 中的任何线性等式 $x_i=\sum_{x_j\notin\mathcal{B}}a_{ij}x_j$, 都有 $\alpha(x_i)=\sum_{x_j\notin\mathcal{B}}a_{ij}\alpha(x_j)$;
- 对于任何对 $(x_i,x_j)\in R$, $\alpha(x_j)=ReLU(\alpha(x_i))$;
- 对于 $\mathcal{X}$ 中的任何 x, $l(x)\leqslant\alpha(x)\leqslant u(x)$;

则意味着已经找到一个反例, *Reluplex* 算法返回 SAT.

如果存在解表 T 中的线性等式 $x_i=\sum_{x_j\notin\mathcal{B}}a_{ij}x_j$ 满足以下之一:

$$l(x_i)>\sum_{x_j\notin\mathcal{B}}\max(a_{ij},0)\cdot u(x_j)+\sum_{x_j\notin\mathcal{B}}\min(a_{ij},0)\cdot l(x_j),$$

$$u(x_i)<\sum_{x_j\notin\mathcal{B}}\max(a_{ij},0)\cdot l(x_j)+\sum_{x_j\notin\mathcal{B}}\min(a_{ij},0)\cdot u(x_j),$$

则在此计算格局中不可能有可以满足约束的赋值, 我们为此计算格局的相应分支标记 UNSAT.

如果经过一定数量的局部搜索后可能仍无法确定当前计算格局究竟是 SAT 还是 UNSAT, 我们需要停止局部搜索, 并选择一个不确定的神经元 x_j, 将其数值边界分割为 $[0,u(x_j)]$ 和 $[l(x_j),0]$. 随后向当前计算格局中添加断言, 即添加形式为 $x_j\geqslant 0$ 或 $x_j\leqslant 0$ 的约束, 并进行新的局部搜索. 当所有分支都被标记为 UNSAT 时, *Reluplex* 返回 UNSAT.

我们用图 1 中的小型神经网络来说明 *Reluplex* 算法, 该网络的输入层包括两个神经元 x_1 和 x_2, 输出为 $y\in\mathbb{R}$. 该网络的行为可以表示为仿射变换 $x_3=0.2x_1-0.7x_2-0.1$, $x_4=0.8x_1-0.8x_2$, $y=0.4x_5+0.6x_6$ 与 *ReLU* 关系 $x_5=ReLU(x_3)$, $x_6=ReLU(x_4)$ 的组合.

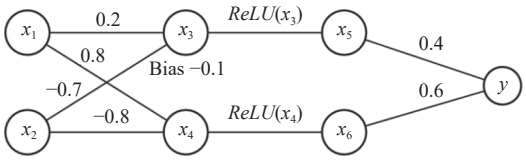


图 1 小型神经网络 f

考虑对性质 (f,X,P) 进行约束求解, 其中输入约束为 $X=[-1,1]\times[-1,1]$, 性质 $P=\{y\mid y<0.3\}$. 编码约束时需要将输入约束 ($-1\leqslant x_1\leqslant 1$, $-1\leqslant x_2\leqslant 1$), 性质 P 的否定 ($y\geqslant 0.3$) 以及神经网络 f 的行为合取.

求解上述约束时, 首先初始化其 *Reluplex* 计算格局. 神经网络 f 在 X 上的 *DeepPoly* 抽象如图 2 所示. 其中, *ReLU* 关系集为 $R=\{(x_3,x_5),(x_4,x_6)\}$. 初始解表根据 f 中的仿射函数 (即 $x_3=0.2x_1-0.7x_2-0.1$, $x_4=0.8x_1-0.8x_2$ 和 $y=0.4x_5+0.6x_6$), 以及来自 *ReLU* 关系的不等式 (即 $x_5\geqslant x_3$ 和 $x_6\geqslant x_4$) 构建. 首先为两个不等式引入两个松弛变量 x_7 和 x_8, 即 $x_5-x_3-x_7=0$ 和 $x_6-x_4-x_8=0$. 从 *DeepPoly* 抽象中可以得到 x_7 和 x_8 的数值边界分别是 [0, 1] 和 [0, 1.6]. 然后为其他 5 个线性等式分别添加一个松弛变量来替代常量, $x_9=-x_3+0.2x_1-0.7x_2$, $x_{10}=0.8x_1-0.8x_2-x_4$, $x_{11}=0.4x_5+0.6x_6-y$, $x_{12}=x_5-x_3-x_7$, 和 $x_{13}=x_6-x_4-x_8$, 其中 $x_9=0.1$ 和 $x_{10},x_{11},x_{12},x_{13}=0$. 对于以上每个方程, 我们启发式地选择一个变量作为基础变量, 并获得初始基础变量集 $\mathcal{B}_0=\{x_3,x_4,y,x_7,x_8\}$. 初始表 T_0 如下:

$$\begin{cases} x_3=0.2x_1-0.7x_2-x_9 \\ x_4=0.8x_1-0.8x_2-x_{10} \\ y=0.4x_5+0.6x_6-x_{11} \\ x_7=x_5-x_3-x_{12}=-0.2x_1+0.7x_2+x_5+x_9-x_{12} \\ x_8=x_6-x_4-x_{13}=-0.8x_1+0.8x_2+x_6+x_{10}-x_{13} \end{cases}.$$

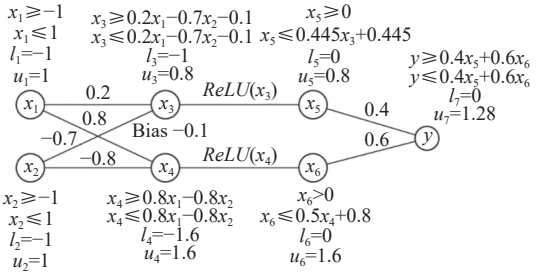


图 2　神经网络 f 的 *DeepPoly* 抽象

初始数值边界和赋值如表 1, 加粗的赋值表示其不满足数值边界或 *ReLU* 关系.

表 1　初始数值边界和赋值

边界与赋值	x_1	x_2	x_3	x_4	x_5	x_6	y	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
l	−1	−1	−1	−1.6	0	0	0.3	0	0	0.1	0	0	0	0
u	1	1	0.8	1.6	0.8	1.6	1.28	1	1.6	0.1	0	0	0	0
α	−1	−1	0.4	0	0.4	0	**0**	**−0.4**	0	0.1	0	0	0	0

注意, y 和 x_7 的赋值不满足其数值边界, 并且它们都是当前的基础变量. 我们选择其中一个 (如 x_7), 进行交换 (pivot). 此时 x_7 的当前赋值小于其下界约束, 而对 x_5 的赋值进行修改可能修复这一冲突, 因为 x_5 并且未达到其上界. 交换 x_7 和 x_5 后可以得到 $x_5=0.2x_1-0.7x_2+x_7-x_9+x_{12}$. 随后将其他方程中出现的 x_5 替换为当前的表达式 $0.2x_1-0.7x_2+x_7-x_9+x_{12}$. 此时基础变量集变为 $\mathcal{B}_1=\{x_3,x_4,x_5,y,x_8\}$, 解表变为 T_1:

$$\begin{cases} x_3=0.2x_1-0.7x_2-x_9 \\ x_4=0.8x_1-0.8x_2-x_{10} \\ x_5=0.2x_1-0.7x_2+x_7-x_9+x_{12} \\ y=0.08x_1-0.28x_2+0.6x_6+0.4x_7-0.4x_9-x_{11}+0.4x_{12} \\ x_8=-0.8x_1+0.8x_2+x_6+x_{10}-x_{13} \end{cases}.$$

此时 x_7 是非基础变量, 我们将其赋值更改为 0 以满足其数值边界, 同时基础变量 x_5 和 y 的赋值根据 T_1 进行相应地变化. 上述操作完成后当前赋值如表 2.

表 2　修改 x_7 后的边界与赋值

边界与赋值	x_1	x_2	x_3	x_4	x_5	x_6	y	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
l	−1	−1	−1	−1.6	0	0	0.3	0	0	0.1	0	0	0	0
u	1	1	0.8	1.6	0.8	1.6	1.28	1	1.6	0.1	0	0	0	0
α	−1	−1	0.4	0	0.4	0	**0.16**	0	0	0.1	0	0	0	0

现在 y 的赋值仍然违反其数值边界, 因此我们选择与非基础变量 x_6 交换, 并调整其赋值为 0.3. 类似地, 我们可以得到基础变量集 $\mathcal{B}_2=\{x_3,x_4,x_5,x_6,x_8\}$、表 T_2 和相应地赋值 (如表 3).

$$\begin{cases} x_3=0.2x_1-0.7x_2-x_9 \\ x_4=0.8x_1-0.8x_2-x_{10} \\ x_5=0.2x_1-0.7x_2+x_7-x_9+x_{12} \\ x_6=-0.13x_1+0.47x_2+1.67y-0.67x_7+0.4x_7-0.67x_9+1.67x_{11}-0.67x_{12} \\ x_8=-0.93x_1+1.27x_2+1.67y-0.67x_7+0.67x_9+x_{10}+1.67x_{11}-0.67x_{12}-x_{13} \end{cases}.$$

表 3 修改 x_6 后的边界与赋值

边界与赋值	x_1	x_2	x_3	x_4	x_5	x_6	y	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
l	−1	−1	−1	−1.6	0	0	0.3	0	0	0.1	0	0	0	0
u	1	1	0.8	1.6	0.8	1.6	1.28	1	1.6	0.1	0	0	0	0
α	−1	−1	0.4	**0**	0.4	**0.233**	0.3	0	0.233	0.1	0	0	0	0

这组赋值不违反任何数值边界, 且 T_2 中所有线性等式都被满足, 但 *ReLU* 关系 $x_6=ReLU(x_4)$ 被违反. 如果在这一步我们用 y 替换 x_6, 我们将再次获得 $\mathcal{B}_1$, T_1 和上一步相同的冲突赋值. 假设现在达到局部搜索的阈值, 我们必须选择一个不确定的 *ReLU* 神经元进行分割. 这里一个启发式的选择是分割 x_4, 因为在 x_4 的局部搜索中已有一次冲突, 但 x_3 没发生过冲突. 在断言 $x_4\leqslant 0$ 的非激活分支中, 我们首先将 x_4 的上界重置为 0, 并在这一断言下再次运行 *DeepPoly*, 以获得新的数值边界和赋值 (如表 4).

表 4 分割 *ReLU* 节点后的边界与赋值

边界与赋值	x_1	x_2	x_3	x_4	x_5	x_6	y	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
l	−1	−1	−1	−1.6	0	0	0.3	0	0	0.1	0	0	0	0
u	1	1	0.8	0	0.8	0	0.32	1	1.6	0.1	0	0	0	0
α	−1	−1	0.4	0	0.4	0	**0.16**	0	0	0.1	0	0	0	0

现在变量 y 再次违反其数值边界, 我们将其与一个非基础变量交换. 经过多步局部搜索后, 我们无法找到满足所有约束的赋值, 因此必须再次进行分割, 断言 $x_3\leqslant 0$ 或 $x_3\geqslant 0$. 在这两个分支中, 问题都简化为一个线性规划问题, 并且这两个分支都是不可行的. 我们仍然有未求解的分支 $x_4\geqslant 0$. 我们以类似的方式对断言 $x_4\geqslant 0$ 进行 *DeepPoly* 算界, 检查赋值是否有效, 并在违反约束的基础变量中进行交换. 经过多步局部搜索后, 我们再次需要对 x_3 进行分割. 在 $x_3\leqslant 0$ 的分支中, 我们找到一个满足所有约束的赋值, 其中的反例是 $(0.675,0.05)^{\mathrm{T}}$, 其输出是 $y=0.3$. 这意味着该组输入违反了性质 $P: y<0.3$. 由于已找到一个真实的反例, *Reluplex* 立即输出 SAT, 最后 $x_3\geqslant 0$ 对应的分支不会被探索. 上述求解过程对应的搜索树 $\mathcal{T}=(V,E,r,L)$ 如图 3 所示, 其中 $V=\{r,v_1,v_2,v_3,v_4,v_5,v_6\}$, 且 r 是根; $E=\{(r,v_1),(r,v_2),(v_1,v_3),(v_1,v_4),(v_2,v_5),(v_2,v_6)\}$; $V\cup E$ 上的标记函数 L 在图 3 中标出. 在这一求解过程中, 叶节点 v_3 和 v_4 是 UNSAT 叶节点, v_5 是 SAT 叶节点, v_6 是一个标记为空序列 ε 的叶节点. 在节点 v_5, 我们已找到一个真正的反例, 因此最后的叶节点 v_6 在求解过程中不会被求解, 其标签应为 ε.

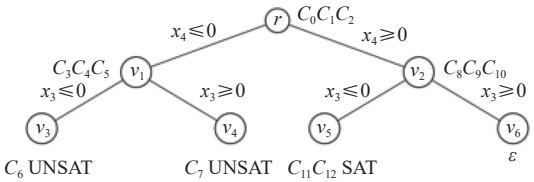


图 3 性质 (f,X,P) 的求解过程

2.2 增量 SMT 求解问题

如前文所述, *Reluplex* 求解实际上是通过添加断言来将不确定 *ReLU* 神经元行为限定为绝对激活或非激活来进行深度优先搜索, 因此, 我们可以将 *Reluplex* 求解过程形式化为一个带标记的二叉树, 并称其为搜索树.

定义 3 (搜索树). 搜索树是一个标记的二叉树 $\mathcal{T}=(V,E,r,L)$, 可以用来表示 *Reluplex* 求解过程, 其中,

- V 是节点集;
- $E\subseteq V\times V$ 是边的集合;
- $r\in V$ 是根;
- L 是一个标记函数, 它给每个节点 $v\in V$ 标记一组有限的计算格局序列, 并给每个边 $e\in E$ 标记一组形式为 $x_j\geqslant 0$ 或 $x_j\leqslant 0$ 的断言, 其中 x_j 是一个不确定的神经元.

如果 *Reluplex* 求解的输出是 UNSAT, 则其搜索树中所有叶节点的标记都有 $C_1,\ldots,C_n$, UNSAT 的形式, 我们将这样的叶节点称为 UNSAT 叶节点. 如果输出是 SAT, 则存在一个其标签形式为 $C_1,\ldots,C_n$, SAT 的叶节点, 我们称之为 SAT 叶节点; 在这种情况下, 其他叶节点要么是 UNSAT 叶节点, 要么被标记为空序列 ε. 对于非空的计算格局序列 ℓ, 我们使用 $\ell\downarrow$ 表示 ℓ 中最后一个不是 SAT 或 UNSAT 的计算格局. 对于节点 $v\in V$, 我们使用 $Assert(v)$ 表示从根 r 到节点 v 的边上的断言集合. 注意节点 v 的语义是初始计算格局与 $Assert(v)$ 中的断言的合取的交集. 为方便叙述, 我们将约束 $\bigwedge_{Q\in Assert(v)} Q$ 简写为 $Assert(v)$.

综上所述, 增量约束求解用于神经网络增量验证的目的是利用原始神经网络 f 的验证过程, 高效地验证修改后的神经网络 f' 上的性质. 本工作中基于 *Reluplex* 框架考虑增量 SMT 求解, 并只针对修改后的神经网络 f' 与原始神经网络 f 在结构上完全相同, 仅在仿射函数中的权重和偏置上有所不同的情况. 下面我们正式陈述要求解的神经网络增量验证问题:

定义 4 (神经网络增量验证问题). 给定安全性质 (f,X,P) 的求解程序 $\mathcal{T}=(V,E,r,L)$, 我们确定性质 (f',X,P) 是否成立, 其中神经网络 f' 仅在权重和偏置上与 f 不同.

2.3 整体框架

图 4 为神经网络的增量 SMT 求解框架. 对于神经网络的增量求解问题, 最重要的假定是修改后的神经网络 f' 与原始神经网络 f 只存在微小的差异, 因此 f' 的验证过程中的大多数分支很可能与验证 f 时类似. 这启发我们可以直接定位到 $\mathcal{T}$ 中叶节点的计算格局, 特别是能够最后一个直接推断出 UNSAT 或 SAT 的计算格局, 并检查该计算格局中的做出的推断是否仍适用于修改后的神经网络 f'.

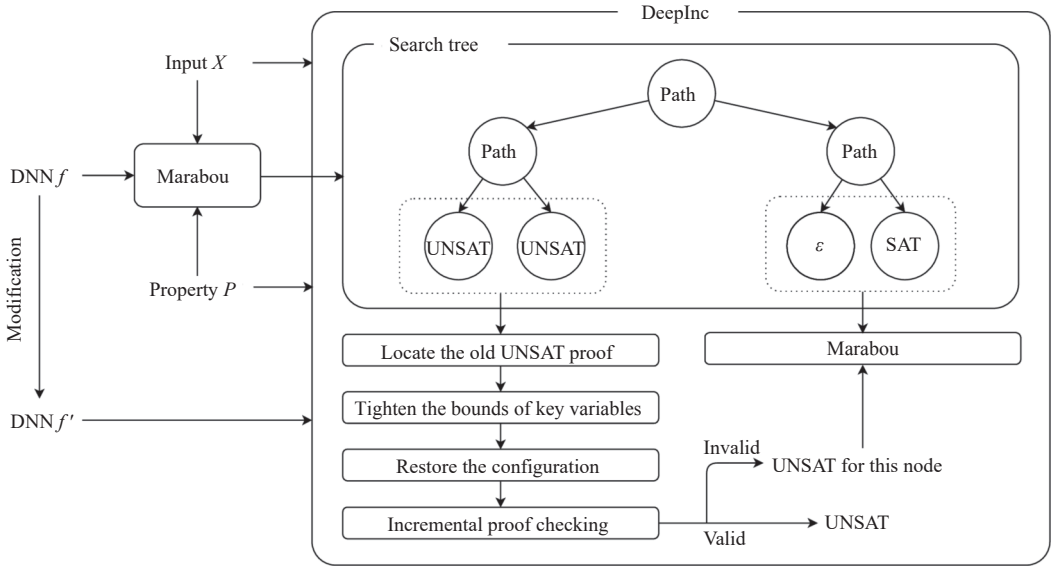


图 4 增量 SMT 求解框架

对于 $\mathcal{T}$ 中的叶节点 v, 我们尝试提取计算格局 $L(v)\downarrow$ 的关键特征, 并将这些特征添加到修改后的神经网络 f' 的初始计算格局 C_0 中. 一个关键特征是 $Assert(v)$, 即从 r 到 v 的路径上断言的合取. 我们可以通过将 $Assert(v)$ 添加

到初始计算格局 C_0 的方式维护 $Assert(v)$ 中涉及的不确定神经元的所有行为. 另一个重要的特征是计算格局 $L(v)\downarrow$ 中的基础变量的集合, 特别是当 v 是一个 UNSAT 叶节点时. 我们用 $(L(v)\downarrow)^*$ 表示在数值边界上存在矛盾的线性等式. UNSAT 的推理严重依赖于 $(L(v)\downarrow)^*$, 而线性等式 $(L(v)\downarrow)^*$ 来源于当前基础变量集下的线性等式解表. 因此, 可以根据这组基础变量集合在 f 的计算格局中模拟线性等式 $(L(v)\downarrow)^*$.

在 f' 的验证过程中, 继承旧验证过程中的 UNSAT 叶节点 v 的关键特征并形成模拟计算格局后, 我们不难观察到 $(L(v)\downarrow)^*$ 中变量的数值边界的精度与 UNSAT 证明的有效性高度相关. 数值边界越精确, 推断出 UNSAT 的可能性就越大. 注意, $L(v)$ 中的计算格局的语义等同于初始计算格局 C_0 和断言 $Assert(v)$ 的合取, 因此我们直接通过 f' 的行为和 $Assert(v)$ 计算数值边界, 而不是通过 $L(v)$ 中的计算格局.

如果旧的验证结果是 UNSAT, 我们增量地检查每个 UNSAT 叶节点上旧的 UNSAT 推理是否仍然适用于 f'. 对于无法立即证明的叶节点 v, 我们将计算格局设置为 $L(v)\downarrow$ 并使用 *Reluplex* 继续求解. 如果旧的验证结果是 SAT, 我们首先定位在 SAT 叶节点, 并检查这个分支中是否有反例. 如果没有反例, 则按从最近到最远的顺序遍历标记为 ε 的叶节点. 若仍没有反例, 则按照上述对 UNSAT 叶节点的处理方式对剩余的 UNSAT 叶节点进行增量验证.

算法 1 展示的是增量验证的主算法, 其中输入是安全性质 (f,X,P) 的 *Reluplex* 求解过程 $\mathcal{T}=(V,E,r,L)$ 和修改后的神经网络 f'. 首先初始化 f' 的计算格局并计算该初始计算格局的数值上下界 U. 计算上下界时通常会调用可靠 (sound) 但不完备 (complete) 的方法, 在本工作中我们选择抽象解释方法 *DeepPoly*. 由于 *DeepPoly* 是可靠的, 如果 *DeepPoly* 成功验证出性质, 算法可以直接返回 UNSAT.

算法 1. 适用于深度神经网络验证的增量 SMT 求解算法.

输入: *Reluplex* 对于安全性质 (f,X,P) 的求解过程 $\mathcal{T}=(V,E,r,L)$ 和修改后的深度神经网络 f';
输出: 如果安全性质 (f',X,P) 不成立, 返回 SAT; 否则返回 UNSAT.

```
1. function Verify( f',X,P,T )
2.    U ← DeepPoly(f',X)
3.    if U ∧ ¬P = ⊥ then
4.       return UNSAT
5.    C_0 ← Initialize(f',X,P,U)
6.    for e ∈ E do
7.       if L(e) ∪ U = ∅ then
8.          T ← T\{e}
9.    if 存在 SAT 叶节点 v_SAT ∈ V then
10.      for 叶节点 v 的标记为 SAT 或 ε do
11.         C ← L(v)↓                          ▷ C = (B,T,R,l,u,α)
12.         l,u ← DeepPoly(f',X ∧ Assert(v))
13.         T ← GaussElimination(C_0,B)
14.         if Reluplex(C) = SAT then
15.            return SAT
16.   for UNSAT 叶节点 v ∈ V do
17.      if Solve(f',T,v) = SAT then
18.         return SAT
19. return UNSAT
```

初始化计算格局后, 我们能推断出 $\mathcal{T}$ 中的一些边是否可以被剪枝. 如果一个边 $e\in E$ 的标签 $L(e)$ 与 *DeepPoly* 计算的结果矛盾, 即 $L(e)\cap DeepPoly(f',X)=\varnothing$, 则从 $\mathcal{T}$ 中删除边 e 及其下的子树, 我们用 $\mathcal{T}\backslash\{e\}$ 表示该剪枝过程.

如果 f 的验证结果是 SAT, 则有唯一的 SAT 叶节点并且可能有一些标记为 ε 的叶节点, 因为 SAT 意味着找到了一组对应反例的赋值, 一旦找到这样一组赋值, 验证会立即终止, 所以可能有一些分支未被验证. 在这种情况下, 我们总是首先对 SAT 叶节点尝实验证, 然后是标记为 ε 的叶节点 (直接调用 *Reluplex* 求解), 最后是 UNSAT 叶节点. 这是因为叶节点的验证结果多数情况下不会因为权重的微小修改而改变, 所以旧 UNSAT 叶节点存在反例的可能性也不会比未探索过的叶节点大.

对于标记为 ε 的叶节点, 我们认为即使原始 SAT 叶节点对应的计算格局在 f' 中的求解结果是 UNSAT, 靠近 SAT 叶节点的叶节点也比其他叶节点在 $\mathcal{T}$ 中更有可能存在反例, 具体来说, 我们通过对称差集 $Assert(v)$ 和 $Assert(v')$ 的基数来衡量标记二叉树 $\mathcal{T}$ 中两个节点 v 和 v' 之间的距离. 在处理标记为 ε 的叶节点之前, 我们计算 SAT 叶节点与它们之间的距离, 并按照这个距离从最小到最大的顺序排序作为求解顺序. 对于 UNSAT 节点 v, 我们通过检查 $\mathcal{T}$ 中的 UNSAT 证明是否仍然适用于修改后的神经网络 f' 进行增量求解, 算法 2 的 $Solve(f',\mathcal{T},v)$ 对该过程进行了详细描述.

算法 2. 针对 UNSAT 节点 v 的函数 $Solve(f',\mathcal{T},v)$.

输入: *Reluplex* 对于安全性质 (f,X,P) 的求解过程 $\mathcal{T}=(V,E,r,L)$, 修改后的深度神经网络 f' 和 USNAT 节点 $v\in V$;
输出: 如果安全性质 $(f',X\wedge Assert(v),P)$ 不成立, 返回 SAT; 否则返回 UNSAT.

1. **function** $Solve(f',\mathcal{T},v)$
2. $\quad C\leftarrow L(v)\downarrow$ ▷ $C=(\mathcal{B},T,R,l,u,\alpha)$
3. $\quad l,u\leftarrow DeepPoly(f',X\wedge Assert(v))$
4. $\quad$ **if** $LP(f'\wedge X\wedge\neg P\wedge Assert(v))$ 不可满足 **then**
5. $\quad\quad$ **return** UNSAT
6. $\quad$ **for** $x_i\in Var((L(v)\downarrow)^*)$ **do**
7. $\quad\quad (l_i,u_i)\leftarrow LP(f'\wedge X\wedge\neg P\wedge Assert(v),x_i)$
8. $\quad\quad \mathcal{T}\leftarrow\mathcal{T}\backslash\{e\}$
9. $\quad T\leftarrow GaussElimination(C_0,\mathcal{B})$
10. $\quad$ **if** $((L(v)\downarrow)^*(T),l,u)$ 推断出 UNSAT **then**
11. $\quad\quad$ **return** UNSAT
12. $\quad$ **else return** $Reluplex(C)$

算法 2 中, 首先获取 $\mathcal{T}$ 中推断出 UNSAT 的计算格局 $L(v)\downarrow$. 根据该计算格局中的基础变量, 我们可以使用高斯消元法将 f' 的初始计算格局 C_0 的表转换为具有相同基础变量的形式. 如前文所述, 对于旧求解过程中推断出 UNSAT 的线性等式 $(L(v)\downarrow)^*$, 如果其中的变量的数值边界越紧, 那么立即推断出 UNSAT 的可能性就越大. 但在 f' 中模拟 $(L(v)\downarrow)^*$ 时, 由 *DeepPoly* 得到的数值边界往往不够紧, 因为 *DeepPoly* 不允许反向分析导致 $Assert(v)$ 中的断言 $x_j\geqslant 0$ (或 $x_j\leqslant 0$) 不能细化同一层或前一层神经元的边界. 为了获得更紧的数值上下界, 我们调用非迭代版本的 DeepSRGR[22], 该版本结合线性规划和线性近似来收紧边界. 我们首先优化 $\min x_i$ 或 $\max x_i$ 以获取 $x_i\in Var((L(v)\downarrow)^*)$ 更紧的边界, 这里 $Var((L(v)\downarrow)^*)$ 表示出现在 $(L(v)\downarrow)^*$ 的变量集合. 随后检查当前解表 T 中模拟 $(L(v)\downarrow)^*$ 产生的方程在当前的数值界下是否能推断出 UNSAT. 如果不能证明 UNSAT, 我们基于当前的数值边界对 v 调用 *Reluplex* 求解.

接下来我们展示如何使用 DeepInc 增量验证修改后的神经网络. 在图 5 和图 6 中, 我们分别展示了从原始神经网络 f 在图 1 的基础上修改得到的两个神经网络 f' 和 f''. 我们首先增量验证性质 (f',X,P). 由于 f 的求解程序中有 SAT 节点 v_5, 所以首先在 f' 中对 v_5 进行增量求解. 我们采用 v_5 在第 1 次验证中的断言 $Assert(v_5)=x_4\geqslant 0\wedge x_3\leqslant 0$ 并初始化 f' 在此约束下的计算格局. 经过一次交换操作后, 找到了一个反例 $(0.714,0.204)^{\mathrm{T}}$, 所以算法返回 SAT. 由于 f' 与 f 仅略有不同, 增量 SMT 求解使我们能够快速找到 f' 的反例. 两个反例的计算格局中断言的神经元激活模式完全相同, 从而避免了对其他节点的大量局部搜索.

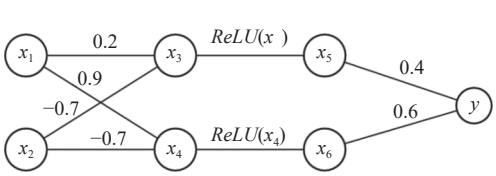


图 5 修改后的神经网络 f'

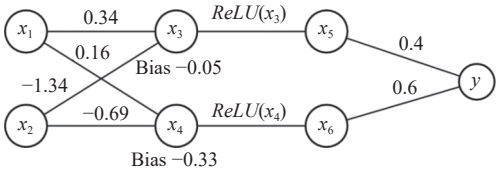


图 6 修改后的神经网络 f''

神经网络 f'' 的权重与原始神经网络 f 差异很大. 以类似方式, 我们增量求解 SAT 叶节点 v_5 . 但在该网络中, v_5 的求解结果变为 UNSAT, 因此我们必须探索其他节点. 首先处理标签为空的节点 v_6 , 检查是否有反例. 这里启发式地首先处理 v_6 而不是两个 UNSAT 叶节点 v_3 和 v_5 , 因为我们认为标记为 ε 的叶节点更有可能找到修改后神经网络的反例. 在这个示例中, v_6 的 *Reluplex* 求解结果依然为 UNSAT, 所以我们必须返回到 UNSAT 叶节点 v_3 和 v_5 . 对于 UNSAT 叶节点的增量求解是不同的, 因为我们可以利用旧的 UNSAT 证明. 对于这里的 UNSAT 叶节点 v_3 , 我们有一个证明 $C_6 \to \text{UNSAT}$. 接下来通过 f'' 在 $Assert(v_3)$ 限制下的计算格局来模拟 C_6 . 我们提取 C_6 中基础变量的集合 $\mathcal{B}_6$, 并将 f'' 的初始表转换为以 $\mathcal{B}_6$ 为基础变量的形式. 此外, $C_6 \to \text{UNSAT}$ 源于表中的一个线性等式 C_6^* , 因此我们关注当前表中的这个线性等式. 注意, 更紧的数值边界有助于证明这个 UNSAT, 因此我们求解一个以此线性等式中的变量为优化对象的线性规划. 通过紧化边界, UNSAT 立即从 C_6^* 推断出来. 类似的程序也在 UNSAT 叶节点 v_4 上进行, 并且它也有一个立即的 UNSAT 证明. 现在树 $\mathcal{T}$ 中的所有叶节点都有验证为 UNSAT, 因此我们的增量 SMT 求解返回 UNSAT, 即性质 (f'', X, P) 成立. 当修改较小时, 我们可以可能继承旧求解程序中相应叶节点的验证结果, 从而避免大量的局部搜索和不必要的 UNSAT 叶节点的求解. 然而, 当修改很大, 以至于本质上改变了神经网络的行为时, 许多验证结果可能无法继承, 可能还需要更多的局部搜索. 直观上, 我们的增量 SMT 求解是否更有效取决于修改的大小以及修改是否从本质上改变神经网络的行为.

我们在定理 1 中总结了算法 1 的可靠性和完备性.

定理 1. 算法 1 是可靠且完备的, 即算法 1 返回 UNSAT 当且仅当性质 (f', X, P) 成立.

证明: 首先证明对于 UNSAT 叶节点 v , 函数 $Solve(f', \mathcal{T}, v)$ 返回 UNSAT 当且仅当 $(f', X \wedge Assert(v), P)$ 成立. 在算法 2 的第 8 行中, 当前计算格局 C 的语义恰好是 $f' \wedge X \wedge Assert(v) \wedge \neg P$, 因为 X , $\neg P$ 以及断言 $Assert(v)$ 已在 $L(v)$ 中编码, 第 8 行的解表编码了 f' 中的仿射变换且 *ReLU* 关系 R 保持不变. 数值边界 l 和 u 是 $f' \wedge X \wedge Assert(v) \wedge \neg P$ 的上近似, 因此它们不违反此语义. 如果在算法 2 的第 10 行, $Solve(f', \mathcal{T}, v)$ 返回 UNSAT, 则 C 得到一个 UNSAT 证明, 这意味着性质 $(f', X \wedge Assert(v), P)$ 成立. 如果在算法 2 的第 11 行, $Solve(f', \mathcal{T}, v)$ 返回 UNSAT, 由于 *Reluplex* 是完备的且可靠的, 性质 $(f', X \wedge Assert(v), P)$ 也成立. 现在我们假设 $(f', X \wedge Assert(v), P)$ 成立, 则无论是在第 10 行迅速给出证明, 还是在第 11 行通过 *Reluplex* 求解, $Solve(f', \mathcal{T}, v)$ 必然输出 UNSAT. 对于那些非 UNSAT 的叶节点 v , 同样在算法 1 的第 10 行, 当前计算格局 C 的语义是 $f' \wedge X \wedge Assert(v) \wedge \neg P$. $Reluplex(C)$ 返回 UNSAT 当且仅当 $(f', X \wedge Assert(v), P)$ 成立. 因此, 我们得出算法 1 返回 UNSAT 当且仅当对所有叶节点 v , $(f', X \wedge Assert(v), P)$ 成立. 根据求解过程建, 自然地有 $\vee_{v\text{是叶节点}} Assert(v) = \top$, 因此对所有叶节点 v , $(f', X \wedge Assert(v), P)$ 成立当且仅当 (f', X, P) 成立.

算法 1 也适用于输入约束 X 和性质 P 被修改的情况. 这在我们使用基于 SMT 的验证工具通过二分搜索计算最大鲁棒性半径时非常有用.

值得注意的是, 我们的算法与 IVAN[23]所采用的技术从不同角度对神经网络增量验证问题进行了优化. 我们增量验证算法的关键是在 SMT 求解器角度快速重用首次验证过程的信息验证修改后的神经网络, 为增量验证提取关键信息和在叶节点计算数值边界是该方法最大的挑战; IVAN 则专注于启发式地构建增量求解的最优搜索策略; 此外, IVAN 中虽然提到对叶子节点的重用, 但其对叶子节点的利用更加直接, 即直接添加验证中的分支信息后调用现成的求解器. 相比之下, 我们的方法从 SMT 求解器的角度, 更细粒度的利用 UNSAT 的验证格局 (即基础变量表, 导致冲突的变量等) 进一步地加速 UNSAT 的证明.

3 实验评估

本节将对增量 SMT 求解器 DeepInc 进行实验评估. 所有实验均在配备 AMD R9 5900HS@3.00 GHz (8 核心) 和 16 GB RAM 的 Ubuntu 20.04 笔记本电脑上进行, 最多同时使用 16 个子进程.

● 网络: 我们使用 ACAS Xu 数据集和 MNIST 数据集上训练出的网络进行评估. ACAS Xu 是无人机空中碰撞避免系统[2]的缩写, 该系统使用神经网络为飞机做出决策以避免飞机在空中发生碰撞. 我们使用的网络包含 6 个隐藏层, 每层有 50 个神经元, 并且所有隐藏神经元都使用 *ReLU* 激活函数. MNIST 数据集包含 60 000 张训练图像和 10 000 张测试图像, 每张图像均为 28×28 像素的手写数字. 我们从 MNIST 数据集训练出的网络中选择 2×256, 4×256, 6×256 这 3 种类型的全连接网络进行测试.

● 性质: 对于 ACAS Xu 数据集训练出的网络, 我们选择文献 [2] 的附录 VI 中提供的前 4 个安全性质, 即性质 φ_1、φ_2、φ_3 和 φ_4. 另外我们还生成了 3 个局部鲁棒性性质, 其 L_∞ 半径分别为 0.05、0.075 和 0.01. 除这些性质外, 我们还通过在原始神经网络上进行二分搜索找到最大鲁棒性半径, 生成了 24 个接近鲁棒边界的局部鲁棒性性质, 验证这些性质非常具有挑战性. 我们基于这些性质来比较 DeepInc、α, β-CROWN 和 IVAN 的验证效率. 由于这 3 种方法都是可靠且完备的, 因此验证这些困难性质的性能真实地显示了各个工具的极限能力. 对 MNIST 上训练出的网络, 我们从测试集选择一些图片, 并将图片分别在 0.05 和 0.1 的扰动下鲁棒作为性质.

● 网络参数的扰动: 我们对神经网络权重的采用两种随机修改方法. 第 1 种修改方法对所有权重进行修改. 且所有权重在变化率 γ 内变化. 具体而言, 权重 w 可以修改为区间 $I(w,\gamma) = [(1-\gamma)w, (1+\gamma)w]$ 中的一个数字. 修改后的权重是根据权重变化率 γ 在 $I(w,\gamma)$ 上进行均匀分布的随机采样获得的. 在我们的实验中, 权重变化率 γ 设置为 0.001、0.01、0.03 和 0.05. 第 2 种修改方法是部分权重修改, 即只修改原始神经网络的部分权重. 在我们的实验中, 改变的权重百分比设为 10%、30% 和 50%, 并且随机选择要改变的权重. 在部分修改中, 我们还对改变的权重施加了权重变化率的限制, 实验中为 0.01、0.03 和 0.05. 对于实际的修改, 我们使用神经网络修复工具 CARE[16]、PRDNN[24]和 VeRe[25]来修复 ACAS Xu 网络中 Marabou 发现的不安全行为. 这里我们不仅考虑了不同程度和不同形式随机生成的扰动, 还考虑了修复产生的扰动, 这能体现出我们的工具在各种扰动下的性能.

我们将 DeepInc 与 Marabou 进行比较, 以展示我们的增量验证算法的有效性和效率, 此外还将 DeepInc 与 α, β-CROWN 和 IVAN 进行比较, 以对比这些工具在增量验证场景下的性能.

3.1 总体效率

我们在随机修改后的神经网络上运行 Marabou 和 DeepInc, 其中 DeepInc 还有 Marabou 给出的原始神经网络的验证过程作为输入. 各个工具验证单个问题的超时时间 (timeout, TO) 设置为 20 000 s. 实验结果显示在图 7、图 8 和图 9 中, 每个子图都对比了两种工具在不同权重扰动下验证对应数据集全部安全性质/局部鲁棒性性质的运行时间, 验证时间为超时或少于 1 s 的情况未在图中显示. 在所有验证问题中, 原始神经网络及其所有修改后的神经网络的对于同一性质的验证结果都是相同的.

总体效率方面, 在 ACAS Xu 数据集上, DeepInc 在 81% 的 ACAS Xu 验证问题中胜过 Marabou, Marabou 的总运行时间是 DeepInc 的 2.003 倍. DeepInc 和 Marabou 在同一个验证问题上都有一个超时结果. 从图 7 中可以看出, 在 4 个安全性质上 DeepInc 显示出更多的效率优势, 有 11 个验证问题比 Marabou 快 100 倍以上, 但它们在局部鲁棒性验证上的表现相对接近. 这可能是因为安全性质输入范围更大, 分支数更多, 能够继承的验证信息更多, 有利于我们的算法发挥作用. 从图 7 中可以看到, 在权重变化率在 0.01 和 0.05 时, DeepInc 在绝大多数问题上都有明显的加速. 在权重变化率为较大的 0.05 时, 多数验证任务性能相当, 但仍有少数验证任务 (x 轴上的验证任务) 有着极大的加速. 在 UNSAT 验证问题 (验证结果为 UNSAT 的验证问题) 和 SAT 验证问题 (验证结果为 SAT 的验证问题) 上分别加速 0.61 倍和 41.01 倍, 表明 DeepInc 在增量搜索反例方面表现突出, 这是由于权重变化不明显时, 修改后的网络的反例往往存在于原始网络反例所在分支的附近, 我们的方法因此能够快速找到反例. 具体来说, 当权重变化率为 0.001、0.01、0.03 和 0.05 时, DeepInc 的加速比分别为 1.25、0.6、0.51 和 0.11, 这表明随着修改的增大, 增量求解的优势有所下降. 这种现象与我们的增量验证算法设计直觉一致.

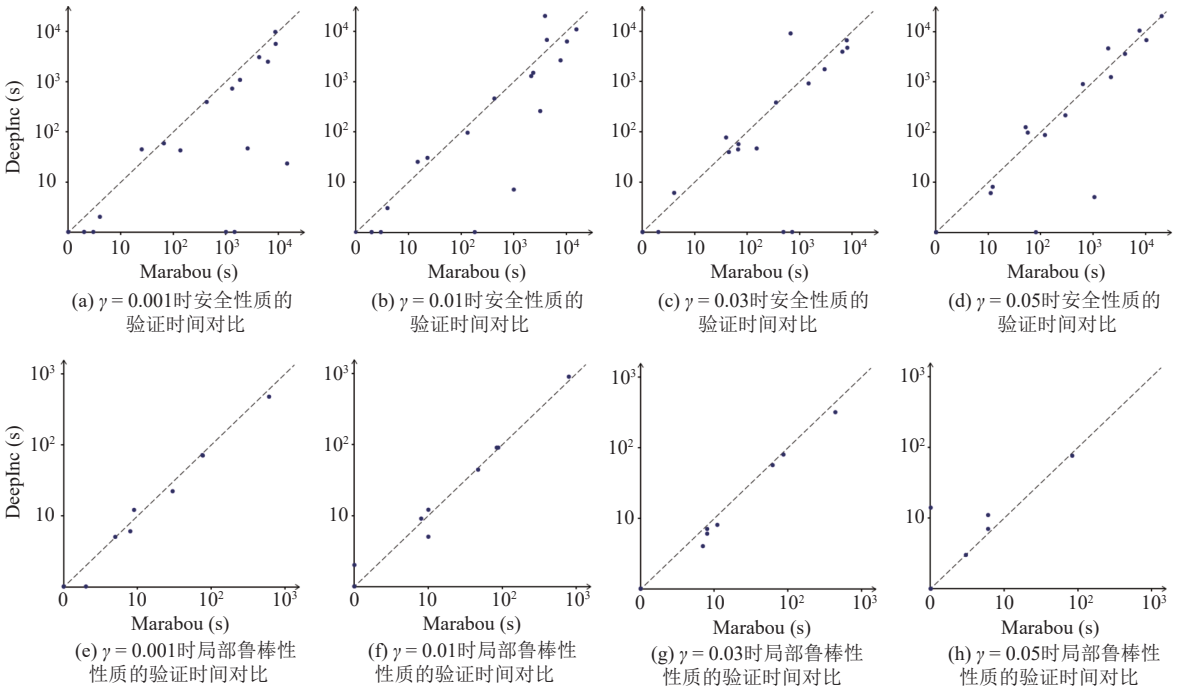


图 7 DeepInc 与 Marabou 在所有权重都变化的 ACAS Xu 网络上验证安全性质和局部鲁棒性的运行时间比较

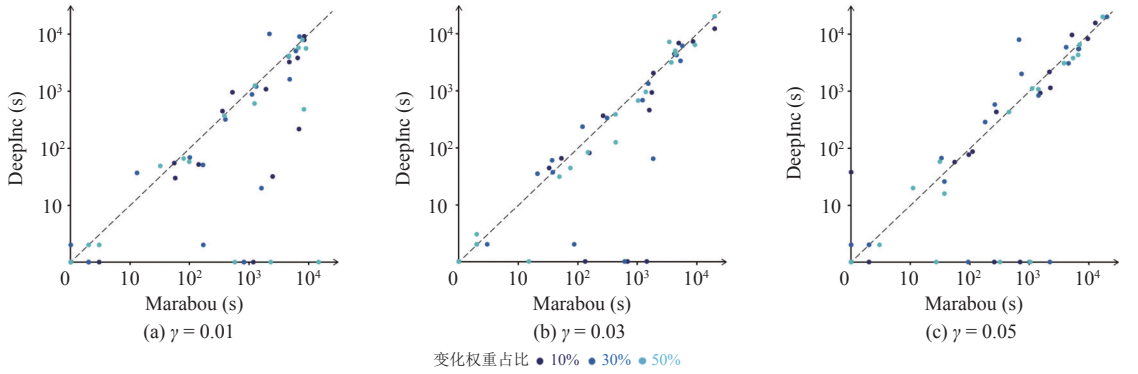


图 8 DeepInc 与 Marabou 在部分权重变化的 ACAS Xu 网络上验证安全性质的运行时间比较

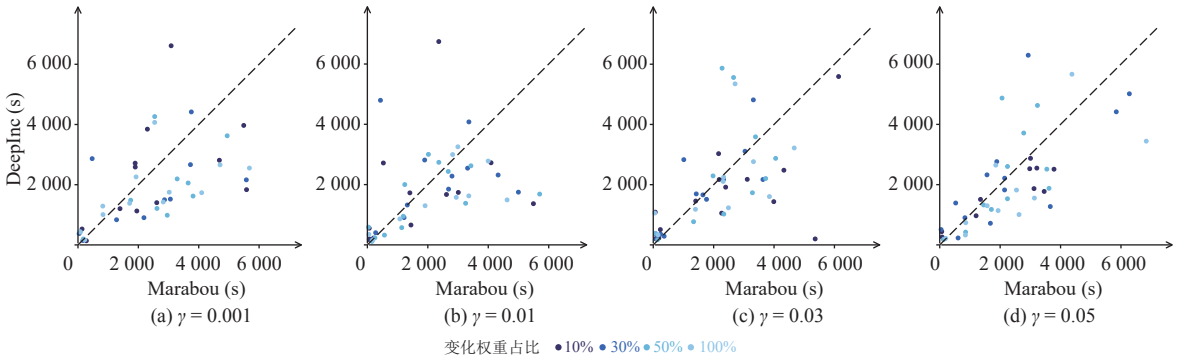


图 9 DeepInc 与 Marabou 在部分权重变化的 MNIST 神经网络上验证局部鲁棒性性质的运行时间

如图 9 所示, 在验证 MNIST 数据集的任务中, 我们在 57.3% 的任务中实现了加速, 最高加速 26.94 倍, 比 Marabou 多解决了 3 个验证任务. 这里我们并未使用对数坐标轴, 因为能验证出来的性质多数集中在 10 000 s 以内. 随着权重变化率的提高, 加速效果略有下降, 这是因为随着扰动大小的增加, 能够复用的证明减少. 我们同样测试了在性质发生变化 (扰动半径分别增加 10%、50% 和 100%) 下的结果, DeepInc 仅在 40% 的任务上实现了加速, 但总体时间仍然比 Marabou 缩短了 1.44 h. 扰动半径的增加一定会改变神经元的上下界, 因此能够继承的证明大大减少, 这也是性质发生改变时, 我们的方法的加速效果有所减弱的主要原因.

3.2 不同修改规模下的性能分析

我们针对 DeepInc 在不同修改规模下的性能进行了一系列测试. 一个重要的指标是在增量模式下直接证明出 UNSAT (即不运行算法 2 的第 11 行) 的 UNSAT 叶节点的百分比. 对于所有权重变化的 4 个权重变化率 0.001、0.01、0.03 和 0.05, 这个百分比分别是 89.01%、85.77%、9.43% 和 85.84%. 对于部分权重变化的实验, 这个百分比是 88.45%. 这表明, 当权重变化率 0.001–0.05 逐渐变化时, 我们增量求解的核心性能并没有显著下降. 从图 7 和图 8 可以看出, 随着权重变化率的增加, DeepInc 的效率优势趋于减小, 尽管仍有少数验证问题中 DeepInc 快了 10 倍以上. 这种现象是合理的, 因为较大的修改不利于增量验证.

除了随机生成的修改, 我们还在神经网络修复的实际修改上对 DeepInc 进行评估. 图 10 显示了 DeepInc 和 Marabou 在由 CARE、PRDNN 和 Vere 修复的神经网络上验证 4 个安全性质的时间, 图 10 中每根柱为一个验证问题, 柱高为不同工具验证该问题的耗时, 较少的耗时会覆盖较多的耗时, 如果两个工具均超时 (TO) 或验证时间均为 0 s (小于 0.1 s 即视为 0 s) 则不在图中使用柱形展示, 仅在右下角统计超时或 0 s 的验证问题的数量. 总体而言, DeepInc 的平均运行时间比 Marabou 快了 596.7 s, 在超过 70% 的验证问题中 DeepInc 更高效.

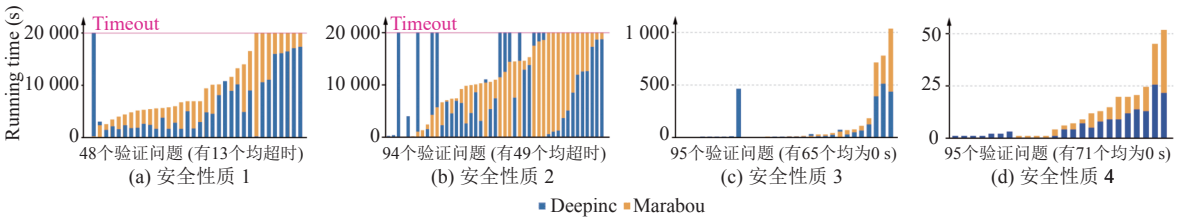


图 10 DeepInc 和 Marabou 在由 CARE、PRDNN 和 Vere 修复的神经网络上验证 4 个安全性质的运行时间

3.3 低效率验证问题分析

我们进一步探索了 DeepInc 效率较低的验证问题. 对于验证结果为 SAT 的验证问题, 通过将新网络中的 *ReLU* 激活模式与 $Assert(v_{\mathrm{SAT}})$ 中的断言进行匹配来检查修改后神经网络中找到的反例是否位于原始 SAT 叶节点 v_{SAT} 的分支上. 可以发现, 所有低效问题中的反例都不在 v_{SAT} 所在的分支上, 这意味着 $Reluplex(v_{\mathrm{SAT}})$ 的输出都是 UNSAT. 反例出现在另外的分支中意味着我们的方法可能在许多输出 UNSAT 的分支中进行了大量的局部搜索后才能发现反例. 对于 DeepInc 输出 SAT 速度比 Marabou 快的验证问题, 超过 90% 的反例位于旧验证程序中 SAT 叶节点的同一分支. 对于 UNSAT 验证问题, 观察到类似本质原因并不容易, 因此我们统计了 DeepInc 效率较低的验证问题中, 在增量验证下证明出 UNSAT 的 UNSAT 叶节点的百分比, 分别为 86.20%、85.09%、65.64%、84.43% 和 83.41%. 与第 3.2 节中的平均比例相比, 其中 4 个略有下降, 另一个 (权重变化率为 0.03) 下降更加显著. 该现象也出现在神经网络修复生成的修改中. 在 DeepInc 效率较低的验证问题中, 超过 90% 的实例修复成功, 即验证结果在修复后从 SAT 变为 UNSAT, 而在 DeepInc 更高效的验证问题中, 这个百分比是 20%. 在 DeepInc 效率较低的验证问题中, 特别是对于那些验证结果为 SAT 的验证问题, 神经网络的修改显著改变了神经网络的行为, 以至于增量模式下的验证大多失败.

3.4 与 α, β-CROWN 和 IVAN 的比较

我们使用 α, β-CROWN[26]在 VNN-COMP 22 的 ACAS Xu 任务中的参数设置. 该工具在易于验证的性质上显示出显著的效率优势. 为了探索基于 SMT 方法的本质优势, 我们在接近鲁棒性边界的性质上对它们进行比较. 验

证接近边界的性质的性能在评估基于 SMT 的验证工具时至关重要. 为了公平比较, 我们没有启用类似 PGD[27]的攻击方法, 并在单进程环境中部署 DeepInc、Marabou、α, β-CROWN 和 IVAN 的实验. α, β-CROWN 是连续多年 VNN-COMP 比赛的冠军, 所以与 α, β-CROWN 的比较有更助于展示 DeepInc 在特定验证问题上的效率. 由于 α, β-CROWN 本身不支持增量验证, 因此我们直接使用其验证修改后的网络. 而 DeepInc 与 IVAN 则先对原始网络进行验证生成搜索树, 再利用搜索树验证修改后的网络. 此外, 我们构建的这些具有挑战性的性质将有助于对神经网络验证工具在极其困难的性质上的性能评估的研究. 神经网络验证问题的难度不仅与神经网络的大小有关, 还与被验证的特定性质有关. 对于大型卷积或残差网络, 如果性质远离鲁棒性边界, 验证可能很容易. 对于接近鲁棒性边界的性质, 即使是非常小的全连接网络, 如 ACAS Xu, 也可能为当前最先进求解器带来挑战.

图 11 为上述配置下的实验结果. 图 11 中每根柱为一个验证问题, 柱高为不同工具验证该问题的耗时, 较少的耗时会覆盖较多的耗时. IVAN 在所有验证问题上都超时. α, β-CROWN 的验证效率在具有 SAT 结果的验证问题上不如 Marabou 和 DeepInc, 但在具有 UNSAT 结果的验证问题上更好. 对于 UNSAT 验证问题, 工具 α, β-CROWN 在分支定界过程中不断收紧神经元边界, 并在约束的下界满足 UNSAT 条件时立即返回结果. 然而, 对于 SAT 验证问题, 这个下界在分支定界过程中可能永远不会满足 SAT 条件, 因此 α, β-CROWN 会持续执行搜索过程直到找到一个反例, 从而导致性能较差.

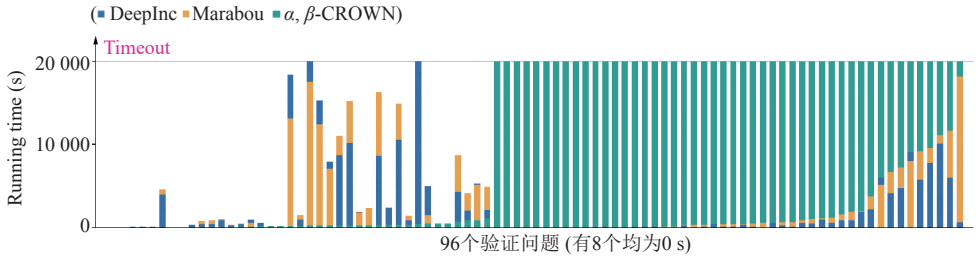


图 11 DeepInc、Marabou 和 α, β-CROWN 在接近鲁棒性边界的性质上的运行时间

这些结果表明, DeepInc 在寻找反例方面比 α, β-CROWN 更高效, 而 α, β-CROWN 在可靠地证明性质方面更为高效. IVAN 的表现不如其他工具, 这可能是因为其底层求解器 RefineZono[28]在困难性质上不够高效. 我们直接使用 IVAN 对修改后的网络及其鲁棒性进行验证, 发现无一例外全部超时. 这意味着我们验证接近鲁棒性边界的性质对于 IVAN 的底层求解器确实具备非常大的挑战.

4 相关工作

文献 [29] 首次提出神经网络验证问题及方法. 目前, 神经网络验证的方法主要包括约束求解[30−33], 抽象解释[5,6,28,34−36], 逐层穷举搜索[37], 全局优化[38−40], 函数近似[41], 归约为博弈问题[42,43], 利用星集抽象[44−46], CEGAR[17−19,22]和 PAC 学习[47−50]等方式. 对于本文考虑的增量情况, SMT 方法是最适合增量设计的约束求解方法, 因为在增量 SMT 求解中已有许多成熟的技术.

关于神经网络增量验证的同期工作是 IVAN[23]. 与我们的方法一样, IVAN 考虑了在神经网络验证中的分支定界 (branch and bound) 过程, 将验证程序抽象为一个二叉树. 不同之处在于 IVAN 的重点不在于复用叶节点的底层证明, 而在于如何从旧的验证程序中获得启发, 并在新的验证程序中制定更好的分支策略. 而我们的方法从底层 SMT 求解的角度出发, 侧重于提取计算格局中的关键特征, 并重用这些信息加速证明过程. 另一个相关工作是 FANC[51], 在其中生成证明模板以快速将证明从旧的转移到新的证明. FANC 是可靠但不完备的, 因为它没有考虑验证中的分支定界过程.

我们方法的有效性依赖于修改前后的网络的相似性, 与当前研究主要关注的网络表征相似性[52−54]与结果相似性[55−57]不同, 这里的相似更多地体现在相同约束下神经元之间的约束关系以及上下边界的相似性, 找到一个合适

的指标度量这种相似性也是一个有意义的问题.

差异验证[58−60]涉及比较两个不同的神经网络模型或同一模型的两个不同版本的行为. 其目标是检查模型是否对相同的输入产生相似的输出, 而增量验证通过重用以前的结果和知识优化验证过程. 有可能将特定的增量验证问题简化为差异验证问题.

5 总 结

在本文中, 我们提出神经网络验证的增量约束求解问题, 并基于 *Reluplex* 框架提出一个增量 SMT 求解算法. 我们将算法实现为增量 SMT 求解器 DeepInc. 实验结果表明 DeepInc 在大多数权重被修改验证问题中更高效, 并且特定场景下与目前最先进的求解器 α, β-CROWN 性能相当. 未来我们将考虑神经网络的结构变化, 以适应基于 CEGAR 框架的验证方法. 另一个挑战性的工作是考虑将 CDCL 框架与增量 SMT 求解结合. 此外, 结合 IVAN 和 DeepInc 的算法法并探究这两种方法如何结合以及选择哪种底层求解器能够获得最佳性能也是一个有价值的研究方向.

References:

[1] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow IJ, Fergus R. Intriguing properties of neural networks. In: Proc. of the 2nd Int'l Conf. on Learning Representations. Banff: OpenReview.net, 2014.

[2] Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ. Reluplex: An efficient SMT solver for verifying deep neural networks. In: Majumdar R, Kunčak V, eds. Computer Aided Verification. Cham: Springer, 2017. 97–117. [doi: 10.1007/978-3-319-63387-9_5]

[3] Ehlers R. Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza D, Narayan Kumar K, eds. Automated Technology for Verification and Analysis. Cham: Springer, 2017. 269–286. [doi: 10.1007/978-3-319-68167-2_19]

[4] Katz G, Huang DA, Ibeling D, Julian K, Lazarus C, Lim R, Shah P, Thakoor S, Wu HZ, Zeljić A, Dill DL, Kochenderfer MJ, Barrett C. The Marabou framework for verification and analysis of deep neural networks. In: Dillig I, Tasiran S, eds. Computer Aided Verification. Cham: Springer, 2019. 443–452. [doi: 10.1007/978-3-030-25540-4_26]

[5] Gehr T, Mirman M, Drachsler-Cohen D, Tsankov P, Chaudhuri S, Vechev M. AI^2: Safety and robustness certification of neural networks with abstract interpretation. In: Proc. of the 2018 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2018. 3–18. [doi: 10.1109/SP.2018.00058]

[6] Singh G, Gehr T, Püschel M, Vechev M. An abstract domain for certifying neural networks. Proc. of the ACM on Programming Languages, 2019, 3(POPL): 41. [doi: 10.1145/3290354]

[7] Eén N, Sörensson N. Temporal induction by incremental sat solving. Electronic Notes in Theoretical Computer Science, 2003, 89(4): 543–560. [doi: 10.1016/S1571-0661(05)82542-3]

[8] Shtrichman O. Pruning techniques for the sat-based bounded model checking problem. In: Proc. of the 11th IFIP WG 10.5 Advanced Research Working Conf. Scotland: Springer, 2001. 58–70. [doi: 10.1007/3-540-44798-9_4]

[9] Schrammel P, Kroening D, Brain M, Martins R, Teige T, Bienmüller T. Successful use of incremental BMC in the automotive industry. In: Núñez M, Güdemann M, eds. Formal Methods for Industrial Critical Systems. Cham: Springer, 2015. 62–77. [doi: 10.1007/978-3-319-19458-5_5]

[10] Disch S, Scholl C. Combinational equivalence checking using incremental sat solving, output ordering, and resets. In: Proc. of the 2007 Asia and South Pacific Design Automation Conf. Yokohama: IEEE, 2007. 938–943. [doi: 10.1109/ASPDAC.2007.358110]

[11] Ravi N, Wendell RE. The tolerance approach to sensitivity analysis in network linear programming. Networks, 1988, 18(3): 159–171. [doi: 10.1002/net.3230180303]

[12] Wondolowski Jr FR. A generalization of Wendell's tolerance approach to sensitivity analysis in linear programming. Decision Sciences, 1991, 22(4): 792–811. [doi: 10.1111/j.1540-5915.1991.tb00365.x]

[13] Filippi C. A fresh view on the tolerance approach to sensitivity analysis in linear programming. European Journal of Operational Research, 2005, 167(1): 1–19. [doi: 10.1016/j.ejor.2004.01.050]

[14] Borgonovo E, Buzzard GT, Wendell RE. A global tolerance approach to sensitivity analysis in linear programming. European Journal of Operational Research, 2018, 267(1): 321–337. [doi: 10.1016/j.ejor.2017.11.034]

[15] Wagner HM. Global sensitivity analysis. Operations Research, 1995, 43(6): 948–969. [doi: 10.1287/opre.43.6.948]

[16] Sun B, Sun J, Pham LH, Shi J. Causality-based neural network repair. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 338–349. [doi: 10.1145/3510003.3510080]

[17] Ashok P, Hashemi V, Křetínský J, Mohr S. DeepAbstract: Neural network abstraction for accelerating verification. In: Proc. of the 18th Int'l Symp. Hanoi: Springer, 2020. 92–107. [doi: 10.1007/978-3-030-59152-6_5]

[18] Elboher YY, Gottschlich J, Katz G. An abstraction-based framework for neural network verification. In: Lahiri S K, Wang C, eds. Computer Aided Verification. Cham: Springer, 2020. 43–65. [doi: 10.1007/978-3-030-53288-8_3]

[19] Ostrovsky M, Barrett C, Katz G. An abstraction-refinement approach to verifying convolutional neural networks. In: Bouajjani A, Holík L, Wu ZL, eds. Automated Technology for Verification and Analysis. Cham: Springer, 2022. 391–396. [doi: 10.1007/978-3-031-19992-9_25]

[20] Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V. Domain-adversarial training of neural networks. In: Csurka G, ed. Domain Adaptation in Computer Vision Applications. Cham: Springer, 2017. 189–209. [doi: 10.1007/978-3-319-58347-1_10]

[21] Liu YQ, Ma SQ, Aafer Y, Lee WC, Zhai J, Wang WH, Zhang XY. Trojaning attack on neural networks. In: Proc. of the 25th Annual Network and Distributed System Security Symp. San Diego: The Internet Society, 2018.

[22] Yang PF, Li RJ, Li JL, Huang CC, Wang JY, Sun J, Xue B, Zhang LJ. Improving neural network verification through spurious region guided refinement. In: Groote J F, Larsen K G, eds. Tools and Algorithms for the Construction and Analysis of Systems. Cham: Springer, 2021. 389–408. [doi: 10.1007/978-3-030-72016-2_21]

[23] Ugare S, Banerjee D, Misailovic S, Singh G. Incremental verification of neural networks. Proc. of the ACM on Programming Languages, 2023, 7(PLDI): 185. [doi: 10.1145/3591299]

[24] Sotoudeh M, Thakur AV. Provable repair of deep neural networks. In: Proc. of the 42nd ACM SIGPLAN Int'l Conf. on Programming Language Design and Implementation. ACM, 2021. 588–603. [doi: 10.1145/3453483.3454064]

[25] Ma JN, Yang PF, Wang JY, Sun YC, Huang CC, Wang Z. VeRe: Verification guided synthesis for repairing deep neural networks. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon Portugal: ACM, 2024. 8. [doi: 10.1145/3597503.3623332]

[26] Xu KS, Zhang H, Wang SQ, Wang YH, Jana S, Lin X, Hsieh CJ. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In: Proc. of the 9th Int'l Conf. on Learning Representations. OpenReview.net, 2021.

[27] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. In: Proc. of the 6th Int'l Conf. on Learning Representations. Vancouver: OpenReview.net, 2018.

[28] Singh G, Gehr T, Mirman M, Püschel M, Vechev M. Fast and effective robustness certification. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 10825–10836.

[29] Pulina L, Tacchella A. An abstraction-refinement approach to verification of artificial neural networks. In: Touili T, Cook B, Jackson P, eds. Computer Aided Verification. Berlin, Heidelberg: Springer, 2010. 243–257. [doi: 10.1007/978-3-642-14295-6_24]

[30] Lomuscio A, Maganti L. An approach to reachability analysis for feed-forward ReLU neural networks. arXiv: 1706.07351, 2017.

[31] Narodytska N, Kasiviswanathan S, Ryzhyk L, Sagiv M, Walsh T. Verifying properties of binarized deep neural networks. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 6615–6624. [doi: 10.1609/aaai.v32i1.12206]

[32] Bunel R, Turkaslan I, Torr PHS, Kumar MP, Lu JY, Kohli P. Branch and bound for piecewise linear neural network verification. The Journal of Machine Learning Research, 2020, 21(1): 42.

[33] Lin W, Yang ZF, Chen X, Zhao QY, Li XK, Liu ZM, He JF. Robustness verification of classification deep neural networks via linear programming. In: Proc. of the 2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). Long Beach: IEEE, 2019. 11410–11419. [doi: 10.1109/CVPR.2019.01168]

[34] Müller C, Serre F, Singh G, Püschel M, Vechev MT. Scaling polyhedral neural network verification on GPUs. In: Proc. of the 4th Conf. on Machine Learning and Systems. MLSys, 2021.

[35] Singh G, Ganvir R, Püschel M, Vechev M. Beyond the single neuron convex barrier for neural network certification. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 1352.

[36] Li JL, Liu JC, Yang P, Chen LQ, Huang XW, Zhang LJ. Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification. In: Proc. of the 26th Int'l Symp. Porto: Springer, 2019. 296–319. [doi: 10.1007/978-3-030-32304-2_15]

[37] Huang XW, Kwiatkowska M, Wang S, Wu M. Safety verification of deep neural networks. In: Majumdar R, Kunčak V, eds. Computer Aided Verification. Cham: Springer, 2017. 3–29. [doi: 10.1007/978-3-319-63387-9_1]

[38] Ruan WJ, Huang XW, Kwiatkowska M. Reachability analysis of deep neural networks with provable guarantees. In: Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence. Stockholm: AAAI Press, 2018. 2651–2659.

[39] Dutta S, Jha S, Sankaranarayanan S, Tiwari A. Output range analysis for deep feedforward neural networks. In: Dutle A, Muñoz C,

Narkawicz A, eds. NASA Formal Methods. Cham: Springer Int'l Publishing, 2018. 121–138. [doi: 10.1007/978-3-319-77935-5_9]

[40] Ruan WJ, Wu M, Sun YC, Huang XW, Kroening D, Kwiatkowska M. Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: IJCAI, 2019. 5944–5952.

[41] Weng TW, Zhang H, Chen HG, Song Z, Hsieh CJ, Daniel L, Boning DS, Dhillon IS. Towards fast computation of certified robustness for ReLU networks. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm, 2018. 5273–5282.

[42] Wicker M, Huang XW, Kwiatkowska M. Feature-guided black-box safety testing of deep neural networks. In: Beyer D, Huisman M, eds. Tools and Algorithms for the Construction and Analysis of Systems. Cham: Springer, 2018. 408–426. [doi: 10.1007/978-3-319-89960-2_22]

[43] Wu M, Wicker M, Ruan WJ, Huang XW, Kwiatkowska M. A game-based approximate verification of deep neural networks with provable guarantees. Theoretical Computer Science, 2020, 807: 298–329. [doi: 10.1016/j.tcs.2019.05.046]

[44] Tran HD, Lopez DM, Musau P, Yang XD, Nguyen LV, Xiang WM, Johnson TT. Star-based reachability analysis of deep neural networks. In: Ter Beek MH, McIver A, Oliveira JN, eds. Formal Methods—The Next 30 Years. Cham: Springer, 2019. 670–686. [doi: 10.1007/978-3-030-30942-8_39]

[45] Tran HD, Bak S, Xiang WM, Johnson TT. Verification of deep convolutional neural networks using imagestars. In: Lahiri SK, Wang C, eds. Computer Aided Verification. Cham: Springer, 2020. 18–42. [doi: 10.1007/978-3-030-53288-8_2]

[46] Yang PF, Li JL, Liu JC, Huang CC, Li RJ, Chen LQ, Huang XW, Zhang LJ. Enhancing robustness verification for deep neural networks via symbolic propagation. Formal Aspects of Computing, 2021, 33(3): 407–435. [doi: 10.1007/s00165-021-00548-1]

[47] Li RJ, Yang PF, Huang CC, Sun YC, Xue B, Zhang LJ. Towards practical robustness analysis for dnns based on PAC-model learning. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 2189–2201. [doi: 10.1145/3510003.3510143]

[48] Baluta T, Chua ZL, Meel KS, Saxena P. Scalable quantitative verification for deep neural networks. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering. Madrid: IEEE, 2021. 312–323. [doi: 10.1109/ICSE43902.2021.00039]

[49] Cardelli L, Kwiatkowska M, Laurenti L, Paoletti N, Patane A, Wicker M. Statistical guarantees for the robustness of Bayesian neural networks. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: IJCAI, 2019. 5693–5700.

[50] Mangal R, Nori AV, Orso A. Robustness of neural networks: A probabilistic and practical approach. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). Montreal: IEEE, 2019. 93–96. [doi: 10.1109/ICSE-NIER.2019.00032]

[51] Ugare S, Singh G, Misailovic S. Proof transfer for fast certification of multiple approximate neural networks. Proc. of the ACM on Programming Languages, 2022, 6(OOPSLA1): 75. [doi: 10.1145/3527319]

[52] Ding F, Denain JS, Steinhardt J. Grounding representation similarity with statistical testing. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2024. 120.

[53] Hamilton WL, Leskovec J, Jurafsky D. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In: Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing. Austin: Association for Computational Linguistics, 2016. 2116–2121. [doi: 10.18653/v1/D16-1229]

[54] Shahbazi M, Shirali A, Aghajan H, Nili H. Using distance on the riemannian manifold to compare representations in brain and in models. NeuroImage, 2021, 239: 118271. [doi: 10.1016/j.neuroimage.2021.118271]

[55] Madani O, Pennock DM, Flake GW. Co-validation: Using model disagreement on unlabeled data to validate classification algorithms. In: Proc. of the 18th Int'l Conf. on Neural Information Processing Systems. Vancouver: MIT Press, 2004. 873–880.

[56] Hsu H, Calmon FP. Rashomon capacity: A metric for predictive multiplicity in classification. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 2101.

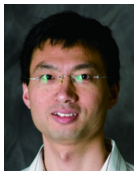
[57] Li YC, Zhang ZQ, Liu BY, Yang ZY, Liu YX. ModelDiff: Testing-based DNN similarity comparison for model reuse detection. In: Proc. of the 30th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. New York: ACM, 2021. 139–151. [doi: 10.1145/3460319.3464816]

[58] Paulsen B, Wang JB, Wang JW, Wang C. NEURODIFF: Scalable differential verification of neural networks using fine-grained approximation. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: IEEE, 2020. 784–796.

[59] Paulsen B, Wang JB, Wang C. ReLUDiff: Differential verification of deep neural networks. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering. Seoul: IEEE, 2020. 714–726.

[60] Mohammadinejad S, Paulsen B, Deshmukh JV, Wang C. DiffRNN: Differential verification of recurrent neural networks. In: Proc. of the 19th Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Paris: Springer, 2021. 117–134. [doi: 10.1007/978-3-030-85037-1_8]

刘宗鑫(1999－), 男, 博士生, CCF 学生会员, 主要研究领域为形式化方法, 神经网络验证.



黄小炜(1979－), 男, 博士, 教授, 博士生导师, 主要研究领域为人工智能安全与验证, AI 可解释性, 形式化方法.

迟智名(1998－), 男, 博士生, CCF 学生会员, 主要研究领域为概率模型检验, 人工智能安全.

蔡少伟(1986－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为约束求解, 形式化方法.

赵梦宇(1999－), 男, 博士生, 主要研究领域为约束求解, 形式化方法.

张立军(1979－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为概率模型检测, 协议验证, 学习算法, 自动驾驶系统验证.

黄承超(1992－), 男, 博士, 主要研究领域为符号与代数计算, 自动推理, 形式化方法等基础理论与算法及其应用.

杨鹏飞(1993－), 男, 博士, CCF 专业会员, 主要研究领域为人工智能安全, 概率模型检验.