软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(5):2307−2324 [doi: 10.13328/j.cnki.jos.006967]

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

基于误差分治的神经网络验证*

董彦松, 刘月浩, 董旭乾, 赵 亮, 田 聪, 于 斌, 段振华

(西安电子科技大学 计算机科学与技术学院, 陕西 西安 710071)

通信作者: 田聪, E-mail: ctian@mail.xidian.edu.cn

摘 要: 随着神经网络技术的快速发展, 其在自动驾驶、智能制造、医疗诊断等安全攸关领域得到了广泛应用, 神经网络的可信保障变得至关重要. 然而, 由于神经网络具有脆弱性, 轻微的扰动经常会导致错误的结果, 因此采用形式化验证的手段来保障神经网络安全可信是非常重要的. 目前神经网络的验证方法主要关注分析的精度, 而易忽略运行效率. 在验证一些复杂网络的安全性质时, 较大规模的状态空间可能会导致验证方法不可行或者无法求解等问题. 为了减少神经网络的状态空间, 提高验证效率, 提出一种基于过近似误差分治的神经网络形式化验证方法. 该方法利用可达性分析技术计算非线性节点的上下界, 并采用一种改进的符号线性松弛方法减少了非线性节点边界计算过程中的过近似误差. 通过计算节点过近似误差的直接和间接影响, 将节点的约束进行细化, 从而将原始验证问题划分为一组子问题, 其混合整数规划 (MILP) 公式具有较少的约束数量. 所提方法已实现为工具 NNVerifier, 并通过实验在经典的 3 个数据集上训练的 4 个基于 *ReLU* 的全连接基准网络进行性质验证和评估. 实验结果表明, NNVerifier 的验证效率比现有的完备验证技术提高了 37.18%.

关键词: 神经网络; 模型抽象; 符号传播; 线性近似; 分治

中图法分类号: TP311

中文引用格式: 董彦松, 刘月浩, 董旭乾, 赵亮, 田聪, 于斌, 段振华. 基于误差分治的神经网络验证. 软件学报, 2024, 35(5): 2307–2324. http://www.jos.org.cn/1000-9825/6967.htm

英文引用格式: Dong YS, Liu YH, Dong XQ, Zhao L, Tian C, Yu B, Duan ZH. Verification for Neural Network Based on Error Divide and Conquer. Ruan Jian Xue Bao/Journal of Software, 2024, 35(5): 2307–2324 (in Chinese). http://www.jos.org.cn/1000-9825/6967.htm

Verification for Neural Network Based on Error Divide and Conquer

DONG Yan-Song, LIU Yue-Hao, DONG Xu-Qian, ZHAO Liang, TIAN Cong, YU Bin, DUAN Zhen-Hua

(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: With the rapid development of neural network technology, neural networks have been widely applied in safety-critical fields such as autonomous driving, intelligent manufacturing, and medical diagnosis. Thus, it is crucial to ensure the trustworthiness of neural networks. However, due to the vulnerability of neural networks, slight perturbation often leads to wrong results. Therefore, it is vital to use formal verification methods to ensure the safety and trustworthiness of neural networks. Current verification methods for neural networks are mainly concerned with the accuracy of the analysis, while apt to ignore operational efficiency. When verifying the safety properties of complex networks, the large-scale state space may lead to problems such as infeasibility or unsolvability. To reduce the state space of neural networks and improve the verification efficiency, this study presents a formal verification method for neural networks based on divide and conquer considering over-approximation errors. The method uses the reachability analysis technique to calculate the upper and lower bounds of nonlinear nodes and uses an improved symbolic linear relaxation method to reduce over-approximation errors during the boundary calculation of nonlinear nodes. The constraints of nodes are refined by calculating the direct and indirect effects of their over-approximation errors. Thereby, the original verification problem is split into a set of sub-problems whose mixed integer linear programming (MILP) formulation has a smaller number of constraints. The method is implemented as a tool named NNVerifier, whose

* 基金项目: 国家重点研发计划 (2018AAA0103202); 国家自然科学基金 (62192734, 62172322)
收稿时间: 2022-11-10; 修改时间: 2023-01-20, 2023-03-23; 采用时间: 2023-05-16; jos 在线出版时间: 2023-09-27
CNKI 网络首发时间: 2023-10-07

properties are verified and evaluated through experiments on four *ReLU*-based fully-connected benchmark networks trained on three classic datasets. The experimental results show that the verification efficiency of the NNVerifier is 37.18% higher than that of the existing complete verification methods.

Key words: neural network (NN); model abstraction; symbolic propagation; linear approximation; divide and conquer

近年来, 神经网络因其在图像分类、对象检测和自然语言处理等方面的出色性能, 在自动驾驶、人脸识别、医学诊断等安全攸关的领域[1]得到了广泛应用. 神经网络为解决软件系统的智能化提供了新的契机并取得了一定进展, 但也带来了全新挑战. 例如神经网络模型非常容易受到对抗攻击而做出错误行为[2−7]. 在计算机系统中, 人眼无法检测到的原始输入上的微小扰动可能会导致深度神经网络预测错误. 这种非鲁棒的行为甚至存在于现实世界的应用中[8−11], 引起了人们对安全攸关人工智能系统应用的担忧. 此外, 投入应用的智能模型通常是由数百万级以上的参数构成. 这些神经网络具有“黑盒”性质, 行为难以理解, 对它们的分析具有挑战性. 安全攸关的系统往往依赖于神经网络做出的关键决策, 因此确保神经网络行为可信和输出符合预期变得至关重要[12]. 传统的神经网络测试和验证方法主要是在输入空间中大量点的集合上评估网络, 并确定输出是否符合预期. 由于输入空间的连续性, 检查所有可能的输入是不可行的[13]. 因此, 即使在大量输入样本上表现良好的网络也可能无法正确泛化到所有情况, 仍可能受到对抗攻击[14,15].

为了解决神经网络的可信性问题, 近年来已有研究开始探索针对神经网络的形式化验证方法[16], 这些方法可以自动证明给定的网络是否满足要求. 总体而言, 这些形式化验证方法分为完备的和不完备的两类. 不完备的方法为网络引入了过度近似的松弛, 因此只能验证输入的一个子集, 例如基于抽象解释的可达性分析方法[17−20], 以及基于线性规划 (LP) 的约束求解方法[21−23]. 完备的方法通常基于混合整数线性规划 (MILP)[24−27]或者可满足性理论 (SAT/SMT) 求解的手段[28,29], 对网络进行精确的编码, 理论上可以找到所有反例并给出验证问题的明确答案. 但是这类方法的主要缺点是状态空间爆炸, 即由 *ReLU* 节点的可能状态生成的搜索空间随着 *ReLU* 节点数量的增加呈指数级增长. 相较于完备的方法, 不完备的方法倾向于更好地扩展, 但是不能提供确定的保证[28]. 此类方法在验证过程中采用凸松弛的方法对神经网络节点进行近似编码或依赖于不精确的抽象域[30], 因此在证明应用程序所需的相关性质的精确性方面依然存在较大的挑战. 目前, 这两类验证方法对小型网络模型有效, 但尚不足以扩展至处理在关键应用 (例如对象检测) 中使用的神经网络的规模. 针对上述问题, 本文提出了一种基于 MILP 的完备的神经网络验证方法, 该方法针对分段线性的激活函数构建精确的约束编码, 基于节点误差分治以提高验证的效率和规模. 在 MILP 的求解过程中, 二进制变量的数量影响问题的计算复杂度. 鉴于可达性分析的方法可在遍历神经网络的过程中获得节点区间, 本文通过自适应的符号线性松弛算法遍历获得节点的上下界, 有效地提高了稳定神经元的数量, 减少了 MILP 编码中的二进制变量. 其次, 由于神经网络的层次化结构导致节点误差随着神经网络的传播而累积和放大, 本文对节点误差的传播误差进行量化计算, 提出了一种新的节点分治策略. 与现有的节点分治方法相比, 显著提高了神经网络的验证效率. 鉴于神经网络模型应用领域的广泛性, 本文旨在对线性修正单元 (*ReLU*) 激活函数的神经网络进行验证. 本文的主要工作包括以下 4 方面.

(1) 利用可达性分析技术计算神经网络中节点的上下界, 通过一组改进的符号线性松弛函数减少计算过程中的过近似误差, 获得更紧密的节点边界, 提高了稳定激活状态的神经元节点数量, 缩减了 MILP 求解过程中的状态空间.

(2) 定义了区间计算过程中引入的过近似误差分析方法, 用于分割网络中的非线性节点. 通过节点分治将原问题转化多个子问题, 获得更紧密的子问题松弛边界, 从而降低验证问题的复杂度.

(3) 在基于 MILP 的优化方法精确的特性上有效地结合了可达性分析方法的便捷, 提高了神经网络的验证效率和规模, 并在此基础上实现了基于误差分析的神经网络验证工具 NNVerifier.

(4) 在 3 个常用的数据集上进行了一系列对比实验, 结果表明 NNVerifier 显著降低了基于 MILP 的神经网络验证方法的状态空间, 性能优于现有的神经网络验证工具.

本文第 1 节介绍神经网络形式化验证的相关方法和研究现状. 第 2 节介绍神经网络的基本概念, 并给出神经网络形式化验证和 MILP 的定义. 第 3 节给出基于误差分治的神经网络验证方法的关键点和实现细节. 第 4 节介绍本文实现的神经网络验证工具 NNVerifier, 并基于此工具开展一系列的对比实验和分析. 第 5 节进行总结以及未来工作展望.

1 神经网络验证相关工作

神经网络的形式化验证包括完备和不完备的方法[16]. 完备的方法对网络进行精确编码, 理论上可以给出验证问题的明确结果, 但需要较高的计算资源. 与完备的验证方法中求解全局最优值的方法不同, 可靠但不完备的验证方法通过引入凸松弛, 计算神经网络验证问题最优解的保证下界. 这类方法尽管可以扩展至更大的网络, 但是可能会给出虚假反例, 因此不能保证给出确定的验证结果.

完备的方法通常基于混合整数线性规划 (MILP)、可满足性理论 (SMT/SAT) 或者区间精化 (input refinement, IR) 等方式对神经网络进行严格的约束编码. 基于 MILP[24,25,31−33]的方法通过编码具有整数约束的分段线性激活函数, 并利用有效的求解器来求解. 基于 SMT 的方法[28,29,34]最初在一个宽松的网络上进行推理, 并将其与分支定界细化相结合, 以迭代地消除高估. 例如, 可以通过扩展 Simplex 算法[28,34]以支持松弛的 *ReLU* 约束或通过直接放松激活函数[29]来获得松弛网络. 基于 SMT 和 MILP 的方法都在产生的问题中对整个网络进行编码. 基于区间精化的方法[35−38]采用结合输入细化的符号传播计算节点边界, 确定网络中非线性节点所处的激活状态. 同时这些输出边界作为 LP 求解器中的约束, 可以降低验证过程由非线性节点引入的复杂度. 完备方法主要针对是大规模非凸优化问题, 这类问题难以实现有效的求解. 相较而言, 不完备的方法通过引入凸松弛找到最优解的保证下界. 基于抽象解释的方法[17−19,39]通过使用抽象域 (区间、Zonotopes、多面体等) 来近似分析输入到输出层的约束关系. 基于线性近似的验证方法[20−23]通过用线性函数近似非线性的神经元, 并给出神经网络输出相对精确的近似范围. 基于对偶松弛的方法[40−44]通过引入对偶变量编码神经网络节点的权重的方式放宽原始优化的约束. 虽然不完备的方法实现了较为高效的性能, 但是验证过程中对 *ReLU* 函数的近似降低了验证结果的精确性. 即使是优化调整的凸松弛边界也可能会产生虚假反例, 因此该类方法不能保证得到确定的结果. 此外, 尽管凸松弛可以在多项式时间求解, 从而在一定程度上提高方法的可扩展性, 但仍不足以处理实际应用中的大规模神经网络.

本文的研究改进了基于 MILP 编码[25,32,33]的神经网络验证算法, 具有在相同约束数量条件下更加严格的非线性约束, 降低了预求解过程的复杂度, 并提高了完备的验证方法的可扩展性. 首先, 采用了改进的符号线性松弛技术, 来提高神经网络中非线性神经元边界的紧密程度并降低神经网络中非线性节点的数量, 进而提高整个神经网络神经元的稳定性. 与已有的符号区间传播算法 (SIP)[45]相比, 本方法采用了改进的线性松弛方式. 对于不能确定的非线性节点, 通过两种不同的下界约束来获得更紧密的边界, 这相较于相关研究中[46]的符号线性松弛方法误差更小. 尽管文献 [29] 中提出的三角松弛给出了 *ReLU* 节点的最紧密凸近似, 但是该方法使用区间传播估计节点的上下界, 且约束数量随着节点数量呈现指数增长, 与本文中的方法相比限制了可验证的网络规模. 其次, 本文提出了一种基于自适应的节点分治策略, 根据节点的过近似误差选择 *ReLU* 节点进行分割, 该策略在计算过近似导致的直接误差的同时计算传播效应带来的间接误差. 通过拆分对整个网络后续分析影响最大的非线性节点, 将验证问题分解为多个子问题, 以获得更紧密的松弛边界. 类似的分治方法在基于 SAT 和 MILP 的神经网络验证[29,33]方法中被委托给底层的求解器, 但未提供 *ReLU* 节点的选择机制. 文献 [36,47] 通过依赖分析和统计节点的影响实现对于 *ReLU* 节点的分割, 但是节点边界不够紧密, 只能减少一部分非线性节点. 本方法充分利用神经网络层次化结构特征, 通过降低节点的过近似误差直接消除一部分二进制变量, 缩减子问题的求解空间. 此外, 通过构建更紧密的松弛方法计算输出节点的边界可达集进而证明部分有界安全性, 对于未解决的子问题采用并行求解的方式. 实验结果表明, 本方法求解效率相较于已有的 MILP 方法更高.

2 基础知识

本节简要介绍前馈神经网络验证相关定义和基本概念, 包括符号区间传播和 MILP 相关概念, 并给出了本文使用的记号.

2.1 前馈神经网络

前馈神经网络 (FFNN) 由一个输入层、一个或多个隐藏层和一个输出层组成, 每一层由多个神经元组成. 如图 1 所示, 一个 L 层的 FFNN 存在向量空间上的映射关系 $f: R^{s_0} \to R^{s_L}$, 其中 s_0 为输入向量的维度, s_L 为输出向量的维度. 层与层之间也存在映射关系: $f_1: R^{s_0} \to R^{s_1}, \ldots, f_L: R^{s_{L-1}} \to R^{s_L}$. 因此神经网络也表示为:

$$f = f_L \circ f_{L-1} \circ \ldots \circ f_1 \tag{1}$$

其中, $\circ$ 表示函数复合. 相邻层神经元通过权重 W 连接, 每个神经元都有一个偏置值 b, 第 i 个隐藏层 z_i 的输入通过前一层 z_{i-1} 输出的仿射变换和非线性激活函数的复合获得. 第 i 层的具体函数表示为:

$$z_i = f_i(z_{i-1}) = \sigma_i(W_i z_{i-1} + b_i) \tag{2}$$

其中, σ 为任意非线性激活函数. 常见的激活函数包括 *ReLU*、*Sigmoid*、tanh 等类型, 其表达形式如公式 (3)–公式 (5) 所示.

$$ReLU(\hat{z}) = \max(\hat{z}, 0) \tag{3}$$

$$Sigmoid(\hat{z}) = \frac{1}{1+\mathrm{e}^{-\hat{z}}} \tag{4}$$

$$\tanh(\hat{z}) = 1 - \frac{2\mathrm{e}^{-\hat{z}}}{\mathrm{e}^{\hat{z}}+\mathrm{e}^{-\hat{z}}} \tag{5}$$

其中, *ReLU* 的使用最为广泛. 非线性函数的存在使得神经网络可以逼近任意函数[48], 这使得神经网络功能更加复杂的同时, 也为验证工作增加了难度.

本文旨在对使用 *ReLU* 激活函数的神经网络进行验证. *ReLU* 是一个分段线性函数, 如图 2 所示. 当节点的输入下界大于 0 时, 称其处于激活态; 当节点的输入上界小于等于 0 时, 称其处于非激活态. 激活态和非激活态都是处于线性区域, 均称为稳定态. 反之, 当节点的输入区间处于非线性区域时, 称其处于不稳定状态. 处于稳定态的 *ReLU* 节点可以直接通过线性计算进行传播, 不稳定的节点则不能直接线性传播. 因此不稳定的节点越少, 验证的复杂度越低.

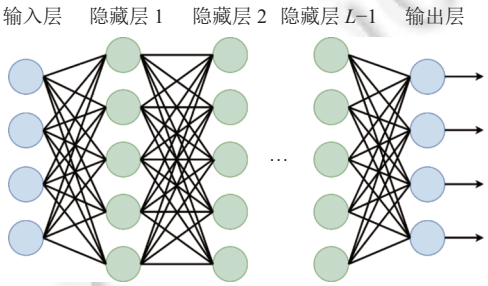


图 1 前馈神经网络结构图

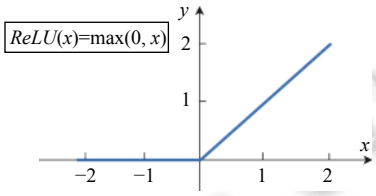


图 2 *ReLU* 函数图像

2.2 神经网络验证问题的形式化描述

给定一个前馈神经网络 f、输入集 $\mathcal{X} \subset R^{s_0}$、输出集 $\mathcal{Y} \subset R^{s_L}$, 验证的一般问题 $(f, \mathcal{X}, \mathcal{Y})$ 是确定神经网络 f 是否满足性质 $\forall x \in \mathcal{X} : f(x) \in \mathcal{Y}$, 即当输入 x 被允许在输入集合 $\mathcal{X}$ 内变化时, 神经网络在输入 x 上的输出 y 是否满足输出约束集 $\mathcal{Y}$.

具体而言, 目前的验证问题可以描述为局部鲁棒性问题, 其核心思想是验证对神经网络的输入 x_0 在给定的半径 ϵ 进行微小扰动后其输出是否受影响. 以一般的分类神经网络的验证为例, 需要保证对于给定的输入周围所有的样本都会被分类到同一标签. 假设目标标签为 tar, 则输入输出约束的形式化描述如公式 (6) 所示.

$$\begin{cases} \mathcal{X} = \{x : \|x - x_0\|_p \leqslant \epsilon\} \\ \mathcal{Y} = \{y : y_{tar} > y_c, \forall c \neq tar\} \end{cases} \tag{6}$$

其中, ϵ 是输入空间中最大允许扰动, 衡量干扰的距离度量可以是任何范式 p. 由此, 验证问题被转化为判断网络在给定的输入集下, 所有输入产生的输出是否都在给定的输出约束中. 其形式化描述如公式 (7) 所示.

$$\forall x \in \mathcal{X} \Rightarrow y = f(x) \in \mathcal{Y} \tag{7}$$

若输入约束集 $\mathcal{X}$ 内的所有输入的输出都在约束集 $\mathcal{Y}$ 中, 则待验证问题是安全的; 换言之, 需要判断的是违背约束的情形是否可达, 即是否存在一个反例 $x \in \mathcal{X}$ 的输出 $y \notin \mathcal{Y}$. 若存在, 则表示网络不安全.

2.3 *ReLU* 函数的 MILP 表示

神经网络的验证问题可以通过优化的方法求出给定断言 (公式 (7)) 为假 (例如, $g(x)=y_c-y_{tar}<0$) 的可行解, 因此神经网络中的非线性激活函数被编码为优化问题的约束. 优化问题是指对于一组特定的参数或变量, 如何从所有满足限制条件的可行解中找出最优解. 优化问题的一般形式如公式 (8) 所示.

$$\text{minimize}: g(x)\ \ \text{s.t.}\ C(x) \tag{8}$$

其中, $g(x)$ 为目标函数, $C(x)$ 是对目标函数中的变量添加的约束条件, 包括不等式约束和等式约束.

根据 *ReLU* 函数组成的神经网络 f 的特性, 通过引入一个二进制变量将 *ReLU* 节点编码为 4 个线性约束, 验证问题可表达为求解混合整数线性规划 MILP 的形式[24], 如公式 (9) 所示.

$$\begin{cases} \min_{x_0,\cdots,x_L} c^{\mathrm{T}}x_L+c_0 \\ \text{s.t.}\begin{cases} x_0\in\mathcal{X} \\ z_i=W_ix_{i-1}+b_i \\ x_i^j\geqslant z_i^j, x_i^j\leqslant z_i^j-l_i^j\cdot\left(1-\delta_i^j\right) \\ x_i^j\leqslant u_i^j\cdot\delta_i^j, x_i^j\geqslant 0, \delta_i^j\in\{0,1\} \end{cases} \end{cases} \tag{9}$$

其中, 层数 $1\leqslant i\leqslant L$, 每层的节点 $1\leqslant j\leqslant s_i$. l_i^j 和 u_i^j 为该不稳定节点激活前的上下界. 根据二进制变量 δ_j^i 的取值可以精确地表示一个 *ReLU* 节点: 当 $\delta_j^i=0$ 时, 表示节点处于非激活态; 当 $\delta_j^i=1$ 时, 表示节点处于激活态. 若目标函数 $c^{\mathrm{T}}x_L+c_0<0$ 有解, 则表示网络不安全.

由于引入了二进制变量, MILP 问题通常是 NP 问题, 求解复杂度随着二进制变量数量的增加而呈指数级增长. MILP 问题的求解效率取决于二进制变量的数量以及线性松弛的紧密度, 即不稳定节点的数量和边界的紧密度. 通过符号区间传播, 可以改善不稳定节点的边界, 减少二进制变量, 进而提高求解效率. 若在实数和整数变量线性约束下的目标函数存在满足 MILP 的可行解, 则表明神经网络存在反例. MILP 算法是对于 *ReLU* 函数的严格描述, 因此基于 MILP 的验证算法是完备的, 即在求解时间充分的前提下, 一定会返回准确的验证结果. 实际情况下, 会设置一个时限以避免在状态空间爆炸时耗费过多的时间.

2.4 符号区间传播

可达性分析的方法通过计算网络输出边界从而对网络进行验证, 可以获取中间节点的上下界以提高 MILP 的求解效率. 区间算数是最简单的区间传播技术, 也称为朴素的区间传播[45], 具体的传播方式如公式 (10) 所示.

$$\begin{cases} \left[\hat{l}_i,\hat{u}_i\right]=W_i\otimes[l_{i-1},u_{i-1}]+[b_i,b_i] \\ [l_i,u_i]=\left[\sigma\left(\hat{l}_i\right),\sigma(\hat{u}_i)\right] \end{cases} \tag{10}$$

其中, $\hat{l}_i$ 和 $\hat{u}_i$ 为经过第 i 层激活函数前的上下界, l_i 和 u_i 为第 i 层输出的上下界, $\otimes$ 表示矩阵乘法, b_i 表示节点的偏置, $\sigma(\cdot)$ 表示激活函数. 该方法可以简单地逐层计算出每个神经元的取值范围, 但是会忽略节点取值范围对输入变量的依赖. 如图 3(a) 所示, 由于节点的极值不会同时取到. 这导致该方法存在较大的高估误差.

符号传播算法[45]解决了后置隐藏层中神经元的取值区间对输入向量没有依赖的问题. 该方法传播的是当前神经元相对于输入向量的上下界表达式 Eq, 第 1 层神经元表达式为:

$$Eq_{\text{up}}(x)=[W_1]+b_1 \qquad Eq_{\text{low}}(x)=[W_1]+b_1 \tag{11}$$

中间线性层神经元的传播方式与区间算数一致.

$$\begin{cases} Eq_{\text{up}}(x)=[W_i]_+Eq_{\text{up}}^{\text{pre}}+[W_i]_-Eq_{\text{low}}^{\text{pre}}+b_i \\ Eq_{\text{low}}(x)=[W_i]_+Eq_{\text{low}}^{\text{pre}}+[W_i]_-Eq_{\text{up}}^{\text{pre}}+b_i \end{cases} \tag{12}$$

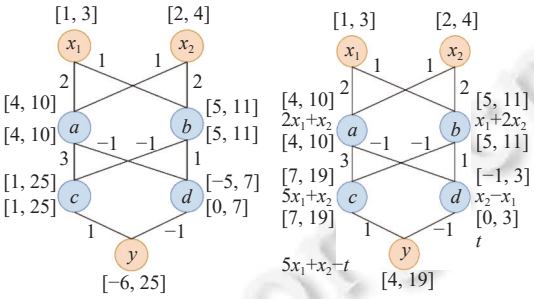
当表达式的传播经过 *ReLU* 激活函数时, 需要对表达式进行具体化操作, 即将输入向量 x 的上下界 $[l,u]$ 代入当前神经元的上下界表达式, 分别计算出具体上界和具体下界, 根据具体上下界将当前神经元边界具体化为实数或者继续以表达式传播, 如图 3(b) 所示.

经过上述过程可以获得节点的上下界 $\left[\underline{Eq_{\text{low}}(x)},\overline{Eq_{\text{up}}(x)}\right]$, 符号区间传播相比于朴素区间算术, 在计算边界的紧密程度上更接近理论上的最紧密的边界, 公式 (13) 成立.

$$\left[\underline{f(x)},\overline{f(x)}\right]\subseteq\left[\underline{Eq_{\text{low}}(x)},\overline{Eq_{\text{up}}(x)}\right]\subseteq\left[\underline{F(x)},\overline{F(x)}\right] \tag{13}$$

其中, $\underline{F(x)}$ 和 $\overline{F(x)}$ 表示的是通过区间传播计算获得的神经网络输出节点的上下界.

然而如图 3(b) 所示, 在 *ReLU* 类型的节点 d 处, 符号表达式 Eq 具体化时, 会在非线性的边界丢失部分依赖关系, 导致产生过近似误差.



(a) 朴素区间传播 (b) 符号区间传播

图 3 神经网络节点边界计算流程图

3 基于节点误差的分治方法

针对神经网络的验证问题, 本文提出基于节点误差的分治方法, 其主要流程如图 4 所示.

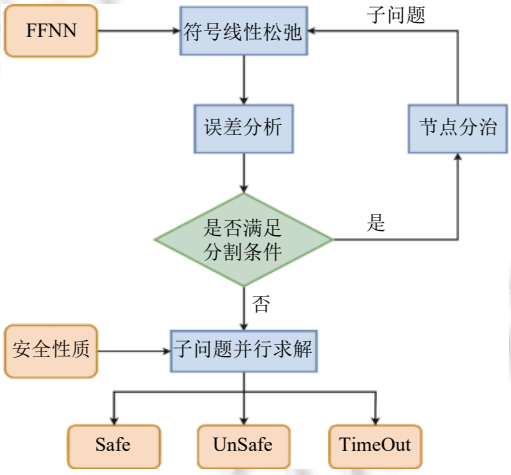


图 4 基于节点误差的分治方法流程图

总体而言, 本方法采用分治策略递归地将验证问题一分为二, 通过将 *ReLU* 节点由最初的不稳定态分割为稳定的激活态 $(0, u]$ 和非激活态 $[l, 0]$, 从而创建两个单独的子问题分支. 子问题具有更紧密的边界和较少的完整性约束节点, 其 MILP 公式的求解较原问题容易, 且可以并行求解以提高效率. 由于子问题的数量随着不稳定节点的数量呈指数增长, 如何选择待分割的 *ReLU* 节点至关重要. 本方法利用 *ReLU* 节点松弛过程中的近似误差来选取可以减少 *ReLU* 完整约束编码的节点, 进而降低子问题的求解复杂度.

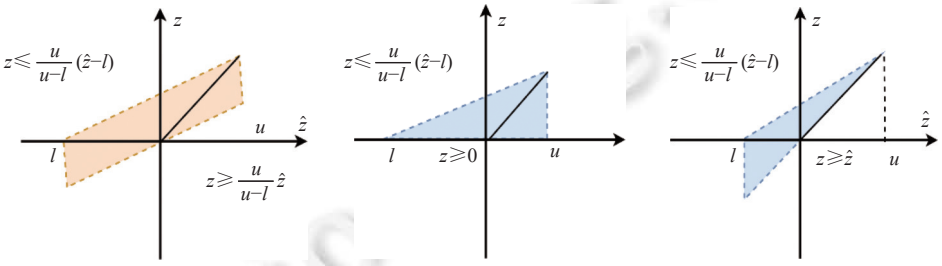
该流程首先采用一种改进的符号线性松弛方法来计算非线性节点的上下界. 根据节点边界的变化, 分别采用两种不同的下界近似函数来松弛该节点的符号传播公式 Eq_{low}. 其次, 分析节点松弛过程中引入的直接误差和对后续传播过程中的间接误差的影响, 构建节点误差矩阵. 随后, 选择误差最大的 *ReLU* 节点进行分割, 将原始问题分

割为子问题, 同时设定分割条件包括分割次数的阈值以确保整个分治流程终止. 最后, 对子问题进行并行求解. 当且仅当所有子问题都满足待验证的安全性质时, 原问题满足该安全性质.

3.1 改进的符号线性松弛方法

基于 MILP 的优化求解方法, 通过引入二进制变量将 *ReLU* 节点编码为 4 个线性约束, 导致求解的状态空间过于复杂. 为了提高求解效率, 本文通过改进的符号线性松弛这种可达性分析方法准确地计算网络中节点的可达区间, 减少编码过程中的二进制变量, 缩减验证问题的状态空间.

符号线性松弛可以看作符号传播的直接扩展, 通过引入线性松弛对非线性的激活函数进行近似, 可以使非线性函数也能以一组符号或者线性方程的方式进行传播. 该方法使得神经网络中的 *ReLU* 函数在传播过程中保持符号依赖关系, 解决了符号区间传播过程中具体化操作引起的输入依赖丢失问题. 如图 5 所示, 符号线性松弛算法可以将传播过程中的不稳定节点的非线性关系松弛为线性关系. 通过在符号区间传播过程中跟踪节点的输入依赖, 在网络输出上计算更严格的边界, 降低了可达性分析过程中的误差.



(a) 平行松弛 (b) 零近似 (c) 恒等近似

图 5 *ReLU* 节点的符号线性松弛方式

具体而言, 符号线性松弛分为两步. 第 1 步进行逐层的线性传播, 传播过程由公式 (14) 给出.

$$\begin{cases}\widehat{Eq}_{\text{low}}^{i}=[W_i]_{+}Eq_{\text{low}}^{i-1}+[W_i]_{-}Eq_{\text{up}}^{i-1}+b_i\\ \widehat{Eq}_{\text{up}}^{i}=[W_i]_{+}Eq_{\text{up}}^{i-1}+[W_i]_{-}Eq_{\text{low}}^{i-1}+b_i\end{cases}\tag{14}$$

第 2 步为对非线性激活函数 *ReLU* 的线性松弛, 由公式 (15) 给出.

$$\begin{cases}Eq_{\text{low}}^{i}=\lambda_l\widehat{Eq}_{\text{low}}^{i}+\mu_l\\ Eq_{\text{up}}^{i}=\lambda_u\widehat{Eq}_{\text{up}}^{i}+\mu_u\end{cases}\tag{15}$$

其中, λ_l 和 λ_u 分别对应下界表达式和上界表达式进行线性近似后的系数, μ_l 和 μ_u 分别为对应下界和上界进行线性近似后的常数. 传统的线性近似采用的是图 5(a) 所示的平行松弛, 而改进后的方法根据节点边界正负两部分的大小不同, 其具体的近似符号公式不同. 如图 5(b) 和图 5(c) 所示, 两种近似方法中上界表达式相同, 即由上界约束产生的误差相等. 而由两种下界约束引起的误差分别为 $u^2/2$ 和 $l^2/2$. 因此, 两种近似方法的选择策略如下.

(1) 当 $abs(l)\geqslant u$ 时, 下界表达式选择零近似, 上下界表示如公式 (16) 所示:

$$\begin{cases}Eq_{\text{up}}=\dfrac{u}{u-l}\left(\widehat{Eq}_{\text{up}}-l\right)\\ Eq_{\text{low}}=0\end{cases}\tag{16}$$

(2) 当 $abs(l)<u$ 时, 下界表达式选择恒等近似, 上下界表示如公式 (17) 所示:

$$\begin{cases}Eq_{\text{up}}=\dfrac{u}{u-l}\left(\widehat{Eq}_{\text{up}}-l\right)\\ Eq_{\text{low}}=\widehat{Eq}_{\text{low}}\end{cases}\tag{17}$$

得到非线性节点的松弛函数之后, 层次化遍历神经网络隐藏层的所有节点. 至此所有节点均为符号表达, 可以通过导出网络中所有节点对输入的依赖关系, 获取所有节点的上下界. 符号线性松弛避免了具体化时丢失输入依赖的问题, 使用上下线性松弛减少了过度估计. 该方法还可用于计算输出节点的边界可达集来证明部分子问题的

边界安全性.

3.2 非线性节点的误差分析

对于大多数网络, 在给定的输入空间 $\mathcal{X}$, 只有一部分中间 *ReLU* 节点在非线性区域 ($l<0<u$) 中运行. 通过符号线性松弛的节点可以根据输入区间计算该节点的上下界. 由于在松弛过程中引入了高估的误差, 它们也被称为过近似节点. 由于在隐藏层的非线性节点引入的过近似误差会随着神经网络的逐层传播计算过程而放大, 多个节点的误差相互叠加会对网络的输出造成较大影响. 因此, 本文使用误差计算函数 $\xi: N \to \mathbb{R}$ 估计误差的大小, 其中 N 是 FFNN 中所有隐藏节点的集合. 该函数同时考虑了直接误差 $\xi_{\text{dir}}: N \to \mathbb{R}$ 和间接误差 $\xi_{\text{indir}}: N \to \mathbb{R}$, 即 $\xi = \xi_{\text{dir}} + \xi_{\text{indir}}$. 接下来具体给出这两种误差的定义.

定义 1. 直接误差. 对于隐藏层的非线性节点, 其直接误差是指在线性松弛过程中其上下界被松弛到线性约束构成的闭合区间内产生的过近似误差, 计算方式如公式 (18) 所示:

$$\xi_{dir} = \begin{cases} \dfrac{u^2 - u \times l}{2}, u \leqslant abs(l) \\ \dfrac{l^2 - u \times l}{2}, u > abs(l) \end{cases} \tag{18}$$

其中, l 和 u 代表了节点的近似上下界. 当 $u \leqslant abs(l)$ 时, 近似方式为零近似, 直接误差为图 5(a) 所示的阴影面积. 当 $u > abs(l)$ 时, 近似方式为恒等近似, 直接误差即为图 5(b) 所示的阴影的面积.

由于神经网络中节点的传播过程复杂, 节点误差随着层次化结构进行传播, 导致不同层次节点之间的误差相互影响, 仅通过直接误差来选择节点是不够合理的. 本方法引入间接误差, 用来模拟逐层传播带来的近似影响.

定义 2. 间接误差. 间接误差代表了过近似节点对后续层的所有节点带来的影响, 计算方式如公式 (19) 所示.

$$\xi_{\text{indir}} = \frac{\xi_{\text{dir}}}{pow(2, i) + 1} \tag{19}$$

其中, ξ_{dir} 是节点的引入的直接误差, i 是当前层的层数. $pow(2, i) + 1$ 则考虑到了间接影响, 即 i 越小误差传播的层数会越多, 影响也就越大.

根据 $\xi = \xi_{\text{dir}} + \xi_{\text{indir}}$, 可以得到每个节点对网络带来的误差影响. 最大的 ξ 表明该节点给神经网络带来了最多的误差, 对输出影响最大, 进而对该节点进行分割处理是有效的.

3.3 基于节点误差的分治方法

分治是一种搜索与迭代的方法, 选择不同的变量进行分治, 将可行解空间反复地分割为越来越小的子集. 本文的方法通过分治将给定的验证问题划分为一组验证子问题, 然后分别将子问题编码为可以并行求解的 MILP 公式. 由于子问题是通过对 *ReLU* 节点的状态进行分割获得的, 因此如果所有子问题满足, 则原始问题满足. 分治过程主要包括递归的拆分不稳定 *ReLU* 节点, 通过在解析树中创建两个单独的分支来实现的: 在第 1 个分支中, 节点的输入被限制为负值, 而在第 2 个分支中被限制为正值. 随着子问题节点的稳定化, 对其进行符号传播将得到更紧密的边界, 甚至可将更多的不稳定节点转化为稳定节点. 这样可减少 MILP 约束的上下界或者直接减少二进制变量的数量, 使得 MILP 公式更容易求解, 其过程如图 6 所示.

显然, 为了获得更为简单的子问题, 可以充分地执行这种分治策略, 子问题中求解的节点上下界将更逼近理论值. 其结果是, 由于子问题的数量在不稳定节点的分裂数量上呈指数增长, 重新进行符号传播的计算代价高昂. 例如, 对含有 n 个节点的网络中的所有节点进行分割会产生 2^n 个子问题, 该方法会转为精确的可达性分析方法. 然而, n 个节点的网络的精确可达性分析复杂度都是 $O(2^n)$, 会导致方法不具备扩展性. 因此, 需要在误差和计算之间进行均衡, 仅对网络中的部分节点进行分割处理. 本文按照节点误差的大小依次对节点进行分割, 这样可以最大程度上降低网络的整体误差, 获得更加紧密的边界, 缩减求解空间.

具体而言, 根据第 4.2 节提出的误差函数分析方法, 构建节点误差矩阵 $E = \left(\xi_i^j\right) \in R^{s_{\max} \times L}$ 旨在识别分治候选节点, 进而启发式地选择影响最大的 *ReLU* 节点进行分割. 节点选择的合适与否, 会直接影响验证子问题的效率. 分治过程中的子问题树与相关节点的激活和非激活状态相关联. 对于一个被拆分的非线性节点 $\left(node_i^j, [l, u]\right)$, 根据其

上下界将节点分为激活态 ($node_{\text{active}}, (0,u]$) 和非激活态 ($node_{\text{inactive}}, [l,0]$) 两部分.

对于分支后的子问题 $(f,\mathcal{X},\mathcal{Y})'$, 重新执行符号线性松弛过程, 计算拆分该节点之后影响的所有 *ReLU* 节点的边界. 之后相应节点的边界得到收紧, 线性松弛也得到改善, 可以确定子问题中更多 *ReLU* 节点的激活状态, 因此子问题的求解状态空间小于原问题.

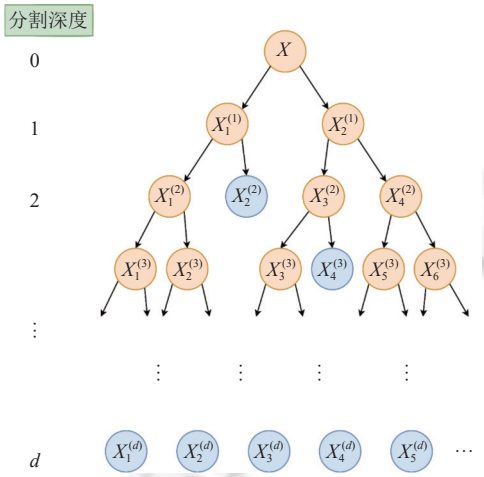


图 6 基于节点误差的分治过程

给定选择用于拆分的 *ReLU* 节点, 子问题 $(f,\mathcal{X},\mathcal{Y})'$ 的 MILP 公式作为公式 (9) 给出, 但根据以下条件替换为分治节点 $node_i^j$ 的 *ReLU* 约束.

(1) 节点状态稳定为激活, 即 $node_{\text{status}} = \text{active}$, 节点区间设置为 $(0,u]$, 约束为 $\delta_i^j = 1$.

(2) 节点状态稳定为非激活, 即 $node_{\text{status}} = \text{inactive}$, 节点区间设置为 $[l,0]$, 约束为 $\delta_i^j = 0$.

基于 MILP 的验证过程是完备的, 即在理论上可以评估所有网络的安全性. 为了减少计算过程的需要被探索的 *ReLU* 节点数量, 将 *ReLU* 节点分割获得与原问题等价的两个分支, 子分支的约束依然采用 MILP 公式, 因此迭代过程中的子问题求解依然是完备的. 在实际计算中, 为了避免在状态空间爆炸时耗费过多的时间, 算法设置了一个时限. 这一过程具体展示如算法 1.

算法 1. 基于误差的节点分治算法.

输入: 验证的问题 $(f,\mathcal{X},\mathcal{Y})$, 分割深度 *count*, 验证时限 *time*;
输出: Safe/UnSafe/TimeOut.

1. 初始化验证问题队列 *queue*=([*f*, 0]);
2. 初始化不可分割子问题集合 *sub_problem*=[];
3. 初始化返回结果 *result* 为 Safe;
4. **while** 问题队列 *queue* 非空:
5. 从队首取验证问题 [*q*, *d*]=*queue.pop*();
6. **if** 当前问题的分割次数 *d* 小于分割深度 *count*;
7. 对问题 *q* 进行符号线性松弛;
8. 计算问题 *q* 的节点误差矩阵 *error_matrix*;
9. 根据误差矩阵 *error_matrix* 选择分割节点 *node*;
10. 将节点 *node* 分割, 生成两个子问题 *q*1, *q*2;
11. 将子问题 *q*1, *q*2 分割次数 *d*+1 添加至队列 *queue*;

12. **else**
13. 将 q 添加至不可分割子问题集合 *sub_problem*;
14. **for** *sub_problem* 中的每一个子问题 q:
15. 对子问题 q 进行 MILP 编码;
16. 将子问题 q 加入并行求解器求解;
17. **if** 求解器返回子问题 q 的可行解:
18. 将 *result* 赋值为 UnSafe, 并终止循环;
19. **if** 程序执行时间超过验证时间限制 *time*:
20. 将 *result* 赋值为 TimeOut, 并终止循环;
21. *sub_problem* 队列为空, 且无可行解返回:
22. 返回验证结果 *result*;

首先, 第 1–3 行进行变量初始化, 将初始验证问题放入验证队列 *queue*, 不可分割子问题集合 *sub_problem* 置为空, 验证结果 *result* 默认为 Safe. 第 4 行启动循环, 对验证问题不断进行分割, 在确定问题队列为空后循环结束. 第 6–13 行进行判断, 如果当前问题 q 的分割次数未达到上限 *count*, 首先第 5–7 行从队列中取出一个验证问题 q, 利用改进的符号线性松弛算法对 q 进行边界计算并根据误差计算公式 (18), 公式 (19) 计算每个节点的误差, 形成误差矩阵 *error_matrix*, 然后根据误差矩阵, 对误差最大的节点 *node* 进行分割, 从而将 q 分解为 $q1$ 和 $q2$ 两个子问题并添加至验证队列. 如果当前问题 q 的分割次数 d 已达到上限 *count*, 则将其添加至不可分割子问题集合 *sub_problem*. 第 14–20 行对 *sub_problem* 中的所有不可分割子问题执行循环, 对各子问题进行 MILP 编码后加入并行求解器. 如果有子问题返回可行解, 表示验证结果为 UnSafe, 则将 *result* 赋值为 UnSafe 并终止循环. 如果所有子程序均无可行解, 表示验证结果为 Safe. 如果程序执行时间超过时限 *time*, 则表示在给定的时间求解器无法给出验证结果, 将 *result* 赋值为 TimeOut 并终止循环. 循环结束后返回验证结果 *result*, 其中 Safe 表示原问题安全, UnSafe 表示原问题不安全, TimeOut 则表示在给定时限内无法做出准确的判断.

4 实 验

作为基于节点误差分析的分治方法的实现, 本文开发了工具 NNVerifier, 用于验证神经网络模型是否符合安全性质.

4.1 工具设计

本文在基于 MILP 方法的基础上, 提出了利用非线性节点的近似误差进行节点分割的方法. 该方法有效地降低了神经网络层间传递带来的误差放大效应, 缩减了 MILP 问题的求解空间. 同时, 方法采用了分治策略将原始验证问题分割为多个子问题, 提高了基于 MILP 的神经网络验证的效率. 作为该方法的实现, 工具 NNVerifier 可以利用多线程进行并行计算. 该工具的架构如图 7 所示.

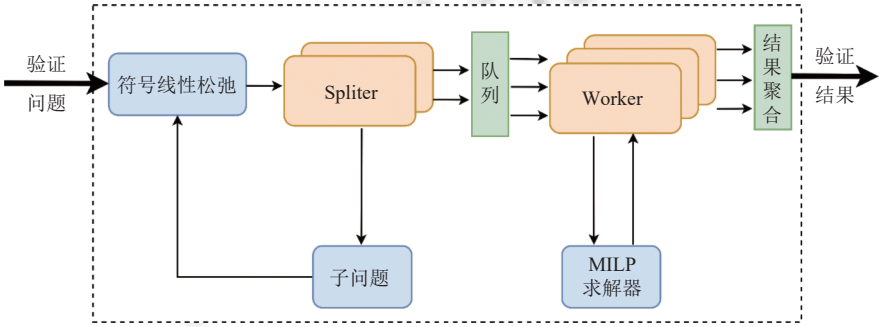


图 7 NNVerifier 架构图

具体而言, NNVerifier 将神经网络验证问题的形式化规范作为输入, 并通过输出安全 (Safe) 或不安全 (UnSafe) 来确定神经网络是否具有鲁棒性, 即是否符合给定的安全约束. 此外, 输出还可能为超时 (TimeOut), 表明方法的运行达到了预设的时限, 但尚未得到确定的结果. 在 NNVerifier 框架中, 改进的符号线性松弛模块首先完成节点的上下界计算过程, 构建误差分析矩阵. 其次, 节点分割模块基于误差矩阵将待拆分的节点分为两个子问题, 重复执行符号线性松弛模块, 有效地降低节点的整体误差. 根据输出节点的上下界, 一部分的子问题可以提前得到验证, 返回 Safe 的结果. 分割后得到的待验证的子问题放入阻塞队列, 等待工作线程将其取出并放入 MILP 求解器. MILP 求解阶段首先对子问题进行编码, 其中稳定状态的 *ReLU* 节点就被当做线性节点, 不稳定的节点基于 MILP 公式被完整编码. 最终将返回的结果聚合. 当某个子问题返回 UnSafe 时表示找到反例, 则原问题存在反例, 说明神经网络不满足给定的安全性质, 验证过程提前终止. 若所有子问题的结果都是 Safe, 表示原问题不存在反例, 说明神经网络满足安全性约束. 整个工具通过多线程技术来实现并行求解, 结合了基于线性松弛的符号传播方法的扩展性和基于 MILP 的优化方法的完备性.

虽然基于误差的分治可以提高验证效率, 但对于已经很容易使用分支定界解决的 MILP 问题, 进一步分割节点可能会增加验证时间. 在实现过程中本文采用 MILP 程序中常用的节点吞吐量[47,49]作为评估指标, 并将其参数化为分支临界值. 在工具验证过程使用分支临界值来确定是否启动节点分割程序 (算法 1), 为了避免基于误差的分治算法对简单问题分割导致的验证时间的增加, 该参数依据问题的复杂程度进行选择.

4.2 测试集

为了评估 NNVerifier 的性能, 本文在常用的 3 个数据集上训练的 4 个模型组成的测试集上进行了实验. 表 1 给出了这些模型的具体结构.

表 1 验证模型的结构和性质

数据集	模型	节点数	结构	性质
ACAS Xu	ACAS Xu	300	<5, 50×6, 5>	186
MNIST	MNIST2	512	<784, 256, 256, 10>	50
	MNIST4	1 024	<784, 256, 256, 256, 256, 10>	50
CIFAR	CIFAR	2 048	<3072, 1024, 512, 512, 10>	100

ACAS Xu[50]是由 45 个激活函数为 *ReLU* 的前馈神经网络组成的集合. 它们是机载防撞系统的一部分, 用于处理传感器接收的数据, 为无人驾驶飞机的水平转向决策提供建议. 这些网络在验证领域被广泛使用[28,33,34,45]. 每个网络有 5 个输入和 5 个输出, 以及 300 个 *ReLU* 节点排列在 6 层中, 其中每层 50 个神经元. 本文根据文献 [28] 的安全规范验证网络, 包括在所有 45 个网络上检查的 4 个性质和在单个网络上检查的 6 个性质. 因此, 共验证了 186 个安全性质.

MNIST[51]是一个包含手写数字 0–9 的图像数据集, 图片格式为 28×28×1 像素的灰度图像. 本文使用在 VNN-COMP2020 中提出的两个全连接前馈神经网络: MNIST2 和 MNIST4. 这两个网络的隐藏层分别为 2 层和 4 层. 每个网络的每层都有 256 个 *ReLU* 节点. 本文针对 25 个正确分类的图像及 0.02 和 0.05 的扰动半径来验证网络的局部对抗鲁棒性. 因此, 总共包含 100 个验证性质.

CIFAR[52]是一个包含飞机、汽车、鸟类、猫等 10 种对象的图片数据集, 图像格式为 32×32×3 的彩色图像. Botoeva 等人[33]在 CIFAR 数据集上训练了一个激活函数为 *ReLU* 的全连接前馈神经网络. 该网络具有 3 层隐藏层, 第 1 层包含 1 024 个神经元, 第 2 层和第 3 层各包含 512 个神经元. 基于该 CIFAR 网络, 本文针对 CIFAR 测试集中随机选择的 100 张正确分类图像和 0.05 的扰动半径验证该网络的局部对抗鲁棒性. 因此, 在该数据集上总共包含 100 个验证性质.

4.3 实验环境及配置

所有实验均在配备 64 GB 内存的 64 位 Ubuntu 18.04 平台上进行, CPU 为 Intel Core i7-7700, 核心数为 4. 软

件方面选择使用 Python 3.8 来实现各神经网络模型. 基于 MILP 方法实现的工具 (Venus[33]、NNVerifier) 依赖于 Gurobi 9.0 作为 MILP 后端求解器.

每个验证问题都设置了 30 min 的时限. NNVerifier 在验证 MNIST2 网络时, 分割深度为 2, 分支临界值为 2 000; 验证 MNIST4 网络时, 分割深度为 3, 分支临界值为 200; 验证 ACAS Xu 网络时, 分割深度为 2, 分支临界值为 1 000; 验证 CIFAR 网络时, 分割深度为 3, 分支临界值为 600; 其余工具的参数均设置为其推荐值.

4.4 实验结果及分析

本文的实验主要由 4 部分组成. 第 1 部分通过将开发的工具 NNVerifier 与 5 个完备的神经网络验证工具 Venus[33], Neurify[46], Nnenum[27], PeregriNN[53]和 Marabou[34]在不同的数据集上进行对比, 评估 NNVerifier 工具的性能. 第 2 部分通过与不同的符号区间传播算法对比, 测试改进的符号线性松弛方法对工具带来的性能提升. 第 3 部分通过对比不同的近似方式, 评估本文采用的近似方法对非线性节点误差的影响, 验证方法的可扩展性. 第 4 部分对 NNVerifier 中关键参数的选取进行实验和分析.

4.4.1 工具性能对比

本节对比 NNVerifier 和 5 个完备的面向 *ReLU* 的神经网络验证工具, 分别是基于依赖分割实现的 Venus, 基于符号线性松弛实现的 Neurify 工具, 基于抽象精化和几何路径枚举方法实现的工具 Nnenum, 基于松弛惩罚机制实现的工具 PeregriNN, 基于 SMT 实现的工具 Marabou. 6 种工具在 ACAS Xu、MNIST2、MNIST4 以及 CIFAR 模型上的实验结果如表 2 所示. 其中, *ver* 代表了可以验证的问题数量, *t* 则表示验证所需的平均时间, 以 s 为单位. 加粗的数据表示在同一数据集下最优的数据.

表 2 神经网络验证工具验证数量和时间对比

Model	Radius	NNVerifier		Venus		Neurify		Nnenum		PeregriNN		Marabou	
		ver	*t* (s)	*ver*	*t* (s)	*ver*	*t* (s)	*ver*	*t* (s)	*ver*	*t* (s)	*ver*	*t* (s)
ACAS Xu	—	186	16.99	182	89.26	179	80.30	**186**	**1.76**	**186**	**1.76**	141	496.95
MNIST2	0.02	**25**	**0.32**	25	0.56	22	280.33	25	4.88	24	74.21	24	78.23
	0.05	25	7.62	**25**	**4.03**	20	413.33	25	63.11	20	362.47	10	1 153.36
MNIST4	0.02	**25**	**14.72**	24	73.00	23	149.27	21	291.59	19	450.12	21	320.20
	0.05	**22**	**1 142.22**	5	1 457.64	3	1 584.29	5	1 523.14	3	1 584.74	2	1 659.50
CIFAR	0.05	**99**	64.25	97	66.32	97	**56.76**	47	1 207.67	87	349.41	98	460.68
总计		**382**	**100.28**	358	159.62	344	210.60	309	435.65	339	251.43	296	566.79

从表 2 可以看出, 在验证 MNIST2 网络时, NNVerifier 和 Venus 两个工具表现出了相当的性能, 但是, 在验证 MNIST4 网络时, 相比于其他 5 个工具, NNVerifier 平均验证数量更多并且平均验证时间更短. 在验证 CIFAR 网络时, NNVerifier 的验证时间仅多于 Neurify, 但验证问题数量最多. 此外, Nnenum 和 PeregriNN 在 ACAS Xu 网络上实现了较好的性能, 但是随着网络规模的增加, Nnenum 和 PeregriNN 验证的问题数量和时间大幅下降. 因此, 在相对较为简单的验证要求下, NNVerifier 的性能与已有工具相近; 而随着问题的复杂度增加, NNVerifier 展现出了优势. 这是因为基于节点误差的分治方法通过将原问题分割为子问题, 使得非线性节点的边界更紧密, 从而降低验证问题的状态空间和复杂度, 提高了验证效率. 最终, NNVerifier 以更短的平均验证时间验证了更多的问题数量. 特别是相比于基于输入分割和依赖剪枝的验证工具 Venus, NNVerifier 在可验证的安全性质数量和验证用时方面有着明显优势. 本文提出的方法多验证了 24 个安全性质, 平均解决时间减少了 37.18%.

此外, 将上述工具在 ACAS Xu、MNIST2、MNIST4 和 CIFAR 网络上验证性质的数量和需要的验证时间进行整合, 得到的结果如图 8 所示, 其中横坐标为时间, 纵坐标为验证性质的数量. 可以观察出, 在相同时间内, NNVerifier 可验证性质的数量超出 Venus、Neurify、Nnenum、PeregriNN 和 Marabou. 实验表明, 相比于基于输入域分割的 Venus 方法, 基于节点误差分析的分割方法可以更有效地缩减问题的状态空间, 具有更高的扩展性.

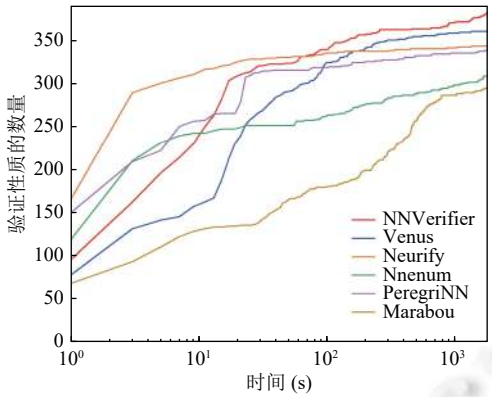


图 8　验证性质数量和运行时间关系图

4.4.2　优化方法的性能分析

为了展示文中所提出的改进的符号线性松弛方法 (MILP_ISLR) 和基于节点误差的分治方法 (MILP_EDC) 两部分带来的性能增益, 在验证难度中等的 CIFAR 网络上以 0.05 的扰动半径设计了 4 组实验, 分别使用 NNVerifier 工具、基于区间算术和上下界编码实现的基准 MILP 工具 (Pure_MILP), 以及在此基础上分别增加了 ISLR 和 EDC 优化的方法. 实验中除对比的方法不同外, 其余设置保持一致. 4 种方法的对比结果如图 9 和表 3 所示.

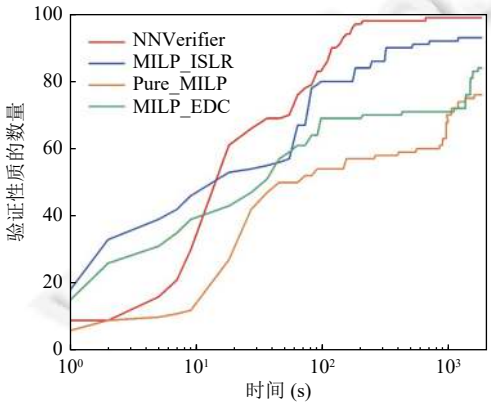


图 9　不同优化算法验证性质数量和运行时间关系图

表 3　不同优化算法验证性能表

Verifier	*ver*	*t* (s)	↓ (%)
Pure_MILP	76	439.78	—
MILP_EDC	84	218.78	50.25
MILP_ISLR	93	124.45	71.70
NNVerifier	99	64.25	85.39

图 9 展示的是不同方法在不同时间内验证的性质个数. 表 3 中的列↓表示和基础的 MILP 方法相比, 加入不同优化方法后平均减少的验证时间. 结果表明, 本文提出基于改进线性松弛的符号传播和基于节点误差的分支定界方法均提高了神经网络验证算法的效率. 其中, 对于 CIFAR 网络, 基于线性松弛的符号传播方法可以将平均验证时间降低 50.25%, 基于节点误差的分支定界方法可以将平均验证时间降低 71.70%. 采用两者相结合的 NNVerifier 将平均验证时间降低了 85.39%, 提高了基于 MILP 方法的可扩展性.

4.4.3　线性松弛方法的性能分析

为了评估本文提出的改进的符号线性松弛方法的性能, 设计 4 组对比实验, 分别采用文献 [46] 提出的符号线性松弛、基于零近似改进的符号线性松弛、基于恒等近似改进的符号线性松弛, 以及本方法提出的根据区间选择零近似和恒等近似相结合的符号线性松弛. 4 组实验均基于 NNVerifier 工具实现, 仅选用的近似方法不同, 其他参数均保持一致. 为了显示相同结构网络随着层数传播误差带来的影响, 选择 MNIST2 网络和 MNIST4 网络作为实验对象. 实验结果如表 4 所示, 其中 *t* 表示花费的平均时间, range 表示输出节点的平均范围. range 越小, 则表明输出范围越紧密, 即直接误差越小.

表 4　4 种线性松弛方法的对比数据

Model	Radius	平行松弛		零近似		恒等近似		零近似 & 恒等近似	
		t (s)	range	t (s)	range	t (s)	range	t (s)	range
MNIST2	0.02	0.06	0.58	0.05	0.44	0.05	15.18	0.05	**0.29**
	0.05	0.05	12.53	0.05	8.33	0.05	52.30	0.05	**6.61**
MNIST4	0.02	0.01	4.31	0.01	**0.85**	0.01	156.74	0.01	0.86
	0.05	0.01	1663.13	0.01	310.6	0.01	2001.40	0.01	**251.87**

由表 4 可以观察出, 当遍历同一网络时, 几种近似方法花费时间基本一致. 这是因为基于线性松弛的符号传播将复杂的非线性函数近似为线性函数, 大大降低了问题复杂度. 此外, 近似方法不同, 得到的最终输出节点的平均范围相差甚远. 虽然在扰动半径为 0.02 的 MNIST4 网络上, 零近似方法与恒等近似结合零近似方法得到的节点的输出范围基本一致, 但是整体上零近似结合恒等近似的方法比其他近似方法得到的节点输出更加紧密. 更紧密的边界则意味着可以验证更大规模网络的性质. 即使问题无法通过输出节点的可达性分析直接得到验证, 也会减少后续的 MILP 求解的状态空间, 提高求解效率.

4.5　参数影响分析

为了研究分治节点的数量和对验证带来的增益之间的关系, 使它们达到平衡, 本文对分割深度和启动分支的临界值参数进行了对比实验.

4.5.1　分割深度的选择

不同的分割深度限制了原问题分割产生的子问题数量, 因此我们针对不同的分割参数对验证时间和数量进行了实验和分析. 该实验基于 4 个网络的全部 386 个性质进行验证, 除分割深度作为参数发生变化外, 其他参数保持一致. 图 10 展示了分割深度和验证问题数量以及求解时间的关系. 横坐标为分割深度, 左侧纵坐标为可验证问题的数量, 右侧纵坐标为验证时间.

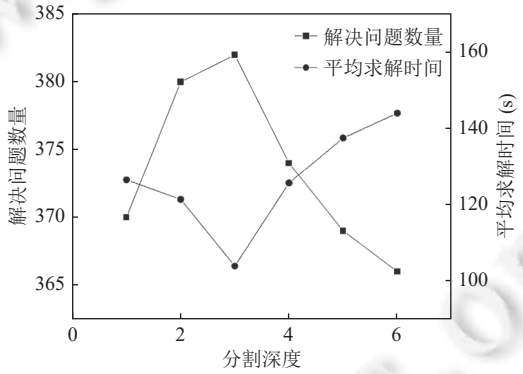


图 10　分割深度与验证问题数量和时间关系图

从图 10 可以看出, 随着分割深度的增加, 可验证问题的数量呈现先上升后下降的趋势, 而平均验证时间则呈现先下降后上升的趋势. 前一部分的时间和验证问题数量的变化表明本文提出的分割方法可以有效地提高验证效率. 而之后随着分割深度的增加, 继续执行分割使得误差较小的节点被分割, 导致单个问题的求解时间没有减少. 在线程等计算资源固定的情况下, 形成的子问题数量更多, 导致整体验证时间增加, 不再提供验证性能的增益.

4.5.2　分支临界值的选择

本文进一步对启动节点分支的临界值进行分析. 为了避免对简单问题分割导致验证时间的增加, 通过设定分支临界值来控制分支过程的启动, 该参数通过评估 MILP 问题的节点吞吐量获得. 由于分支临界值对于复杂问题的影响较为明显, 因此该部分实验选择基于 CIFAR 和 MNIST4 的网络进行, 扰动半径均选取 0.05. 除了参数分支临界值不一致, 其余参数在对应网络中分别保持一致. 图 11 和图 12 分别展示了 NNVerifier 在 CIFAR 网络和 MNIST4 网络上的实验结果.

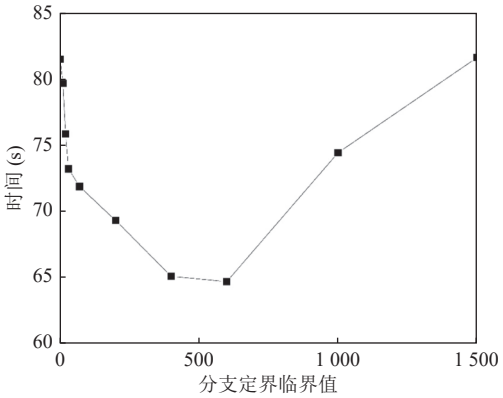


图 11 CIFAR 网络分支临界值与时间关系图

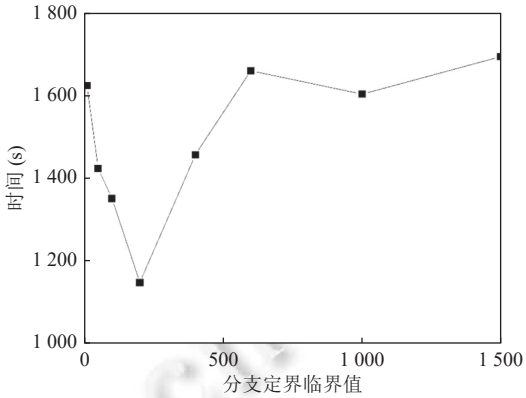


图 12 MINST4 网络分支临界值与时间关系图

随着分支临界值的增加, 所需时间也呈现了先下降后上升的趋势, 与分割深度的结果类似. 该参数过小时会导致一些相对误差较小的节点被分割, 相对简单的子问题被分割成多个, 增加了额外的开销. 该参数过大时, 对于难以解决的问题, 会导致验证问题的状态空间复杂, 从而增加求解器运行时间. 由于网络层数的增加会使得节点的过近似误差被叠加, 因此在选择该参数时应考虑网络的复杂程度. 具体而言, 对于深层神经网络应取低值, 而对于浅层神经网络网络应取高值. 例如, NNVerifier 在 CIFAR 网络上分支临界值取 600 时表现最佳; 在 MNIST4 网络上分支临界值取 200 时表现最佳.

综上所述, 通过与当前主流工具进行对比实验, 本文方法实现的验证工具 NNVerifier 展示出了更高的效率和更好的可扩展性. 一系列的消融实验也分析了基于节点误差的分治方法中主要策略和参数对性能变化带来的影响. 此外, 本文方法虽然在包含使用 *Sigmoid* 或 tanh 激活函数的网络被限制, 但是在由线性操作 (如全连接、卷积和平均池化) 和分段线性操作 (如 *ReLU* 和最大池化) 组成的网络上可以实现较好的扩展.

5 结 论

以神经网络为代表的人工智能技术是软件智能合成的重要组成部分. 为保障智能模型的可信性, 本文提出了一种基于误差分治的神经网络验证方法. 该方法基于 MILP 公式扩展, 采用改进的符号线性松弛来计算给定输入范围内网络输出的严格近似函数, 并使用过近似节点的误差约束分割策略将原验证问题划分为等价的一系列子问题, 有效地缩减了神经网络验证过程中的状态空间. 作为该方法的实现, 开发了一个高效且可扩展的神经网络验证工具 NNVerifier, 用于验证给定的神经网络安全性质并提供具体的反例. 广泛的实验结果表明, 本文方法的性能优于现有的神经网络形式化验证系统.

在未来的工作中, 计划进一步研究更加高效且可扩展的神经网络的安全验证方法, 针对其他类型激活函数, 研究如何使用类似的上下界优化算法和节点分割技术处理包含 *Sigmoid* 和 tanh 等激活函数的网络, 用于扩展可验证的网络结构, 提高验证方法的性能以帮助验证规模较大、复杂度较高的神经网络.

References:

[1] Pouyanfar S, Sadiq S, Yan YL, Tian HM, Tao YD, Reyes MP, Shyu ML, Chen SC, Iyengar SS. A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys, 2018, 51(5): 92. [doi: 10.1145/3234150]

[2] Biggio B, Corona I, Maiorca D, Nelson B, Šrndić N, Laskov P, Giacinto G, Roli F. Evasion attacks against machine learning at test time. In: Proc. of the 2013 Joint European Conf. on Machine Learning and Knowledge Discovery in Databases. Prague: Springer, 2013. 387–402. [doi: 10.1007/978-3-642-40994-3_25]

[3] 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: ICLR, 2014.

[4] Li MH, Jiang PP, Wang Q, Shen C, Li Q. Adversarial attacks and defenses for deep learning models. Journal of Computer Research and Development, 2021, 58(5): 909–926 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2021.20200920]

[5] Li XJ, Wu GW, Yao L, Zhang WZ, Zhang B. Progress and future challenges of security attacks and defense mechanisms in machine

learning. Ruan Jian Xue Bao/Journal of Software, 2021, 32(2): 406–423 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6147.htm [doi: 10.13328/j.cnki.jos.006147]

[6] Liu XM, Zhang ZH, Yang CY, Zhang YC, Shen C, Zhou YD, Guan XH. Adversarial technology of text content on online social networks. Chinese Journal of Computers, 2022, 45(8): 1571–1597 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2022.01571]

[7] Zhang SS, Zuo X, Liu JW. The problem of the adversarial examples in deep learning. Chinese Journal of Computers, 2019, 42(8): 1886–1904 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2019.01886]

[8] Athalye A, Engstrom L, Ilyas A, Kwok K. Synthesizing robust adversarial examples. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: ICML, 2018. 284–293.

[9] Eykholt K, Evtimov I, Fernandes E, Li B, Rahmati A, Xiao CW, Prakash A, Kohno T, Song D. Robust physical-world attacks on deep learning visual classification. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 1625–1634. [doi: 10.1109/CVPR.2018.00175]

[10] Yu ZF, Yan Q, Zhou Y. A survey on adversarial machine learning for cyberspace defense. Acta Automatica Sinica, 2022, 48(7): 1625–1649 (in Chinese with English abstract). [doi: 10.16383/j.aas.c210089]

[11] Yuan L, Li XM, Pan ZX, Sun JM, Xiao L. Review of adversarial examples for object detection. Journal of Image and Graphics, 2022, 27(10): 2873–2896 (in Chinese with English abstract). [doi: 10.11834/jig.210209]

[12] Yang MF, Gu B, Duan ZH, Jin Z, Zhan NJ, Dong YW, Tian C, Li G, Dong XG, Li XF. Intelligent program synthesis framework and key scientific problems for embedded software. Chinese Space Science and Technology, 2022, 42(4): 1–7 (in Chinese with English abstract). [doi: 10.16708/j.cnki.1000-758X.2022.0046]

[13] Wang Z, Yan M, Liu S, Chen JJ, Zhang DD, Wu Z, Chen X. Survey on testing of deep neural networks. Ruan Jian Xue Bao/Journal of Software, 2020, 31(5): 1255–1275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]

[14] Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A. The limitations of deep learning in adversarial settings. In: Proc. of the 2016 IEEE European Symp. on Security and Privacy (EuroS&P). Saarbruecken: IEEE, 2016. 372–387. [doi: 10.1109/EuroSP.2016.36]

[15] Li ZT, Sun JB, Yang KW, Xiong DH. A review of adversarial robustness evaluation for image classification. Journal of Computer Research and Development, 2022, 59(10): 2164–2189 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.20220507]

[16] Liu CL, Arnon T, Lazarus C, Strong C, Barrett C, Kochenderfer MJ. Algorithms for verifying deep neural networks. Foundations and Trends® in Optimization, 2021, 4(3–4): 244–404. [doi: 10.1561/2400000035]

[17] Gehr T, Mirman M, Drachsler-Cohen D, Tsankov P, Chaudhuri S, Vechev M. Ai2: 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]

[18] Mirman M, Gehr T, Vechev MT. Differentiable abstract interpretation for provably robust neural networks. In: Proc. of the 35th Int'l Conf. on Machine Learning. 2018. 3575–3583.

[19] 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.

[20] Tjandraatmadja C, Anderson R, Huchette J, Ma W, Patel K, Vielma JP. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. 2020. 21675–12686.

[21] 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. 2018. 5273–5282.

[22] 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]

[23] Tran HD, Bak S, Xiang WM, Johnson TT. Verification of deep convolutional neural networks using imagestars. In: Proc. of the 32nd Int'l Conf. on Computer Aided Verification. Los Angeles: Springer, 2020. 18–42. [doi: 10.1007/978-3-030-53288-8_2]

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

[25] Tjeng V, Xiao KY, Tedrake R. Evaluating robustness of neural networks with mixed integer programming. In: Proc. of the 7th Int'l Conf. on Learning Representations. 2019.

[26] Bastani O, Ioannou Y, Lampropoulos L, Vytiniotis D, Nori AV, Criminisi A. Measuring neural net robustness with constraints. In: Proc. of the 30th Int'l Conf. on Neural Information Processing Systems. Barcelona: Curran Associates Inc., 2016. 2621–2629.

[27] Bak S, Tran HD, Hobbs K, Johnson TT. Improved geometric path enumeration for verifying ReLU neural networks. In: Proc. of the 32nd

Int'l Conf. on Computer Aided Verification. Los Angeles: Springer, 2020. 66–96. [doi: 10.1007/978-3-030-53288-8_4]

[28] Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ. Reluplex: An efficient SMT solver for verifying deep neural networks. In: Proc. of the 29th Int'l Conf. on Computer Aided Verification. Heidelberg: Springer, 2017. 97–117. [doi: 10.1007/978-3-319-63387-9_5]

[29] Ehlers R. Formal verification of piece-wise linear feed-forward neural networks. In: Proc. of the 15th Int'l Symp. on Automated Technology for Verification and Analysis. Pune: Springer, 2017. 269–286. [doi: 10.1007/978-3-319-68167-2_19]

[30] Ghorbal K, Goubault E, Putot S. The zonotope abstract domain Taylor1+. In: Proc. of the 21st Int'l Conf. on Computer Aided Verification. Grenoble: Springer, 2009. 627–633. [doi: 10.1007/978-3-642-02658-4_47]

[31] Akintunde M, Lomuscio A, Maganti L, Pirovano E. Reachability analysis for neural agent-environment systems. In: Proc. of the 16th Int'l Conf. on Principles of Knowledge Representation and Reasoning. Tempe: AAAI, 2018. 184–193.

[32] Anderson R, Huchette J, Ma W, Tjandraatmadja C, Vielma JP. Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming, 2020, 183(1): 3–39. [doi: 10.1007/s10107-020-01474-5]

[33] Botoeva E, Kouvaros P, Kronqvist J, Lomuscio A, Misener R. Efficient verification of ReLU-based neural networks via dependency analysis. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence. New York: AAAI, 2020. 3291–3299. [doi: 10.1609/aaai.v34i04.5729]

[34] 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: Proc. of the 31st Int'l Conf. on Computer Aided Verification. New York: Springer, 2019. 443–452. [doi: 10.1007/978-3-030-25540-4_26]

[35] Henriksen P, Lomuscio AR. Efficient neural network verification via adaptive refinement and adversarial search. In: Proc. of the 24th ECAI European Conf. on Artificial Intelligence. Santiago de Compostela: IOS Press, 2020. 2513–2520.

[36] Henriksen P, Lomuscio A. DEEPSPLIT: An efficient splitting method for neural network verification via indirect effect analysis. In: Proc. of the 30th Int'l Joint Conf. on Artificial Intelligence. 2021. 2549–2555.

[37] Wang SQ, Zhang H, Xu KD, Lin X, Jana S, Hsieh CJ, Kolter JZ. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. 2021. 29909–29921.

[38] Hashemi V, Kouvaros P, Lomuscio A. OSIP: Tightened bound propagation for the verification of ReLU neural networks. In: Proc. of the 19th Int'l Conf. on Software Engineering and Formal Methods. Springer, 2021. 463–480. [doi: 10.1007/978-3-030-92124-8_26]

[39] Li JL, Liu JC, Yang PF, 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 Static Analysis Symp. Porto: Springer, 2019. 296–319. [doi: 10.1007/978-3-030-32304-2_15]

[40] Dvijotham K, Stanforth R, Gowal S, Mann TA, Kohli P. A dual approach to scalable verification of deep networks. In: Proc. of the 34th Conf. on Uncertainty in Artificial Intelligence. 2018. 550–559.

[41] Wong E, Kolter JZ. Provable defenses against adversarial examples via the convex outer adversarial polytope. In: Proc. of the 35th Int'l Conf. on Machine Learning. 2018. 5283–5292.

[42] Raghunathan A, Steinhardt J, Liang P. Certified defenses against adversarial examples. In: Proc. of the 6th Int'l Conf. on Learning Representations. 2018.

[43] Fazlyab M, Morari M, Pappas GJ. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Trans. on Automatic Control, 2022, 67(1): 1–15. [doi: 10.1109/TAC.2020.3046193]

[44] Dathathri S, Dvijotham K, Kurakin A, Raghunathan A, Uesato J, Bunel R, Shankar S, Steinhardt J, Goodfellow I, Liang P, Kohli P. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 5318–5331.

[45] Wang SQ, Pei KX, Whitehouse J, Yang JF, Jana S. Formal security analysis of neural networks using symbolic intervals. In: Proc. of the 27th USENIX Conf. on Security Symp. Baltimore: USENIX Association, 2018. 1599–1614.

[46] Wang SQ, Pei KX, Whitehouse J, Yang JF, Jana S. Efficient formal safety analysis of neural networks. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 6369–6379.

[47] Kouvaros P, Lomuscio A. Towards scalable complete verification of ReLU neural networks via dependency-based branching. In: Proc. of the 30th Int'l Joint Conf. on Artificial Intelligence. 2021. 2643–2650.

[48] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7): 1527–1554. [doi: 10.1162/neco.2006.18.7.1527]

[49] Klotz E, Newman AM. Practical guidelines for solving difficult mixed integer linear programs. Surveys in Operations Research and Management Science, 2013, 18(1–2): 18–32. [doi: 10.1016/j.sorms.2012.12.001]

[50] Julian KD, Lopez J, Brush JS, Owen MP, Kochenderfer MJ. Policy compression for aircraft collision avoidance systems. In: Proc. of the

35th IEEE/AIAA Digital Avionics Systems Conf. (DASC). Sacramento: IEEE, 2016. 1–10. [doi: 10.1109/DASC.2016.7778091]
[51] LeCun Y. The MNIST database of handwritten digits. 1998. http://yann.lecun.com/exdb/mnist/
[52] Krizhevsky A. Learning multiple layers of features from tiny images. Technical Report, Toronto: University of Toronto, 2009.
[53] Khedr H, Ferlez J, Shoukry Y. PeregriNN: Penalized-relaxation greedy neural network verifier. In: Proc. of the 33rd Int'l Conf. on Computer Aided Verification. 2021. 287–300. [doi: 10.1007/978-3-030-81685-8_13]

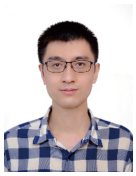
附中文参考文献:

[4] 李明慧, 江沛佩, 王骞, 沈超, 李琦. 针对深度学习模型的对抗性攻击与防御. 计算机研究与发展, 2021, 58(5): 909–926. [doi: 10.7544/issn1000-1239.2021.20200920]
[5] 李欣姣, 吴国伟, 姚琳, 张伟哲, 张宾. 机器学习安全攻击与防御机制研究进展和未来挑战. 软件学报, 2021, 32(2): 406–423. http://www.jos.org.cn/1000-9825/6147.htm [doi: 10.13328/j.cnki.jos.006147]
[6] 刘晓明, 张兆晗, 杨晨阳, 张宇辰, 沈超, 周亚东, 管晓宏. 在线社交网络文本内容对抗技术. 计算机学报, 2022, 45(8): 1571–1597. [doi: 10.11897/SP.J.1016.2022.01571]
[7] 张思思, 左信, 刘建伟. 深度学习中的对抗样本问题. 计算机学报, 2019, 42(8): 1886–1904. [doi: 10.11897/SP.J.1016.2019.01886]
[10] 余正飞, 闫巧, 周鋆. 面向网络空间防御的对抗机器学习研究综述. 自动化学报, 2022, 48(7): 1625–1649. [doi: 10.16383/j.aas.c210089]
[11] 袁珑, 李秀梅, 潘振雄, 孙军梅, 肖蕾. 面向目标检测的对抗样本综述. 中国图象图形学报, 2022, 27(10): 2873–2896. [doi: 10.11834/jig.210209]
[12] 杨孟飞, 顾斌, 段振华, 金芝, 詹乃军, 董云卫, 田聪, 李戈, 董晓刚, 李晓锋. 嵌入式软件智能合成框架及关键科学问题. 中国空间科学技术, 2022, 42(4): 1–7. [doi: 10.16708/j.cnki.1000-758X.2022.0046]
[13] 王赞, 闫明, 刘爽, 陈俊洁, 张栋迪, 吴卓, 陈翔. 深度神经网络测试研究综述. 软件学报, 2020, 31(5): 1255–1275. http://www.jos.org.cn/1000-9825/5951.htm [doi: 10.13328/j.cnki.jos.005951]
[15] 李自拓, 孙建彬, 杨克巍, 熊德辉. 面向图像分类的对抗鲁棒性评估综述. 计算机研究与发展, 2022, 59(10): 2164–2189. [doi: 10.7544/issn1000-1239.20220507]

董彦松(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为神经网络的形式化验证, 对抗样本攻击.

刘月浩(1997－), 男, 博士生, 主要研究领域为神经网络的形式化验证.

董旭乾(1996－), 男, 硕士, 主要研究领域为神经网络的形式化验证.



赵亮(1984－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为智能系统的形式化验证, 形式化方法.

田聪(1981－), 女, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为形式化方法, 可信软件, 人工智能系统安全.

于斌(1990－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为形式化方法.

段振华(1948－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为时序逻辑, 形式化方法, 高可信嵌入式系统.