软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(3):1231−1256 [doi: 10.13328/j.cnki.jos.007061]

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

深度神经网络修复策略综述*

梁 震[1], 刘万伟[2,3], 吴陶然[4,5], 薛 白[4,5], 王 戟[1], 杨文婧[1]

[1](量子信息研究所兼高性能计算国家重点实验室 (国防科技大学), 湖南 长沙 410073)

[2](国防科技大学 计算机学院, 湖南 长沙 410073)

[3](复杂系统软件工程实验室 (国防科技大学), 湖南 长沙 410073)

[4](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)

[5](中国科学院大学 计算机科学与技术学院, 北京 100190)

通信作者: 刘万伟, E-mail: wwliu@nudt.edu.cn

摘 要: 随着智能信息时代的发展, 深度神经网络在人类社会众多领域中的应用, 尤其是在自动驾驶、军事国防等安全攸关系统中的部署, 引起了学术界和工业界对神经网络模型可能表现出的错误行为的担忧. 虽然神经网络验证和神经网络测试可以提供关于错误行为的定性或者定量结论, 但这种事后分析并不能防止错误行为的发生, 如何修复表现出错误行为的预训练神经网络模型依然是极具挑战性的问题. 为此, 深度神经网络修复这一领域应运而生, 旨在消除有缺陷的神经网络产生的错误预测, 使得神经网络满足特定的规约性质. 目前为止, 典型的神经网络修复范式有 3 种: 重训练、无错误定位的微调和包含错误定位的微调. 介绍深度神经网络的发展和神经网络修复的必要性; 厘清相近概念; 明确神经网络修复的挑战; 详尽地调研目前已有的神经网络修复策略, 并对内在联系与区别进行分析和比较; 调研整理神经网络修复策略常用的评价指标和基准测试; 展望未来神经网络修复领域研究中需要重点关注的可行方向.

关键词: 深度神经网络修复; 错误行为; 重训练; 微调; 错误定位

中图法分类号: TP311

中文引用格式: 梁震, 刘万伟, 吴陶然, 薛白, 王戟, 杨文婧. 深度神经网络修复策略综述. 软件学报, 2024, 35(3): 1231–1256. http://www.jos.org.cn/1000-9825/7061.htm

英文引用格式: Liang Z, Liu WW, Wu TR, Xue B, Wang J, Yang WJ. Survey on Repair Strategies for Deep Neural Network. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1231–1256 (in Chinese). http://www.jos.org.cn/1000-9825/7061.htm

Survey on Repair Strategies for Deep Neural Network

LIANG Zhen[1], LIU Wan-Wei[2,3], WU Tao-Ran[4,5], XUE Bai[4,5], WANG Ji[1], YANG Wen-Jing[1]

[1](Institute of Quantum Information & State Key Laboratory of High Performance Computing (National University of Defense Technology), Changsha 410073, China)

[2](College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

[3](Key Laboratory of Software Engineering for Complex Systems (National University of Defense Technology), Changsha 410073, China)

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

[5](School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: With the development of the intelligent information era, applications of deep neural networks in various fields of human society, especially deployments in safety-critical systems such as automatic driving and military defense, have aroused concern from academic and industrial communities on the erroneous behaviors that deep neural networks may exhibit. Although neural network verification and neural

* 基金项目: 国家重点研发计划 (2022YFA1005101, 2021ZD014030); 国家自然科学基金 (91948303-1, 61836005, 61872371, 6203202)
梁震和刘万伟为共同第一作者.
收稿时间: 2023-06-02; 修改时间: 2023-08-19; 采用时间: 2023-09-25; jos 在线出版时间: 2023-12-20
CNKI 网络首发时间: 2023-12-25

network testing can provide qualitative or quantitative conclusions about erroneous behaviors, such post-analysis cannot prevent their occurrence. How to repair the pre-trained neural networks that feature wrong behavior is still a very challenging problem. To this end, deep neural network repair comes into being, aiming at eliminating the unexpected predictions generated by defective neural networks and making the neural networks meet certain specification properties. So far, there are three typical neural network repair paradigms: retraining, fine tuning without fault localization, and fine tuning with fault localization. This study introduces the development of deep neural networks and the necessity of deep neural network repair, clarifies some similar concepts, and identifies the challenges of deep neural network repair. In addition, it investigates the existing neural network repair strategies in detail and compares the internal relationships and differences among these strategies. Moreover, the study explores and sorts out the evaluation metrics and benchmark tests commonly used in neural network repair strategies. Finally, it forecasts the feasible research directions that should be paid attention to in the future development of neural network repair strategies.

Key words: deep neural network (DNN) repair; erroneous behaviors; retraining; fine tuning; fault localization

1 引 言

近年来, 随着计算硬件算力的提升和数据规模的不断增长, 深度神经网络 (deep neural network, DNN) 迅猛发展, 成为深度学习 (deep learning, DL) 的主要计算模型, 在图像识别[1−3]、自然语言处理[4−6]、自动驾驶[7,8]、医药医学[9,10]、恶意软件检测[11,12]等各个领域取得了巨大成功. 特别是以 ChatGPT[13]为代表的大模型的面世, 人类社会与深度神经网络模型的交织和联系将会更加紧密和深厚.

尽管深度神经网络在众多领域取得了极大的成功, 但是神经网络并不总是正确的, 也就是说在一些情况下, 它们可能产生不满足预期性质的输出结果. 例如对一张大熊猫的图片进行肉眼难以察觉的噪声扰动, 原来可以正确处理这张图片的神经网络以 99.3% 的概率误认为它是长臂猿[14]. 这样的错误行为可能导致系统故障[15,16]、错误的刑事拘捕[17,18]、甚至生命财产的损失[19]. 比如, 一个不公平的神经网络, 相较于白色人种, 它可能更倾向于预测黑色人种是犯罪嫌疑人, 从而引起警力资源的浪费和错误的拘捕; 对一张“STOP”道路指示牌进行局部的遮挡, 神经网络将会错误地将其识别为“限速”的路标, 这会使得自动驾驶汽车存在严重的安全隐患[20]. 因此, 鉴于深度神经网络表现出的错误行为, 其在安全攸关领域的应用受到极大限制, 这成为一个重要而亟待解决的研究问题.

为了获得用户对深度神经网络的信任, 减轻他们对于错误行为的担忧, 确保深度神经网络在部署到实际应用之前满足特定要求并产生预期输出就显得至关重要. 近几年, 深度神经网络验证 (DNN verification)[21−39]、深度神经网络测试 (DNN testing)[40−45]和基于性质满足的深度神经网络训练 (DNN property-satisfaction-guided training)[23,46−51]等方面的进展为解决这个问题提供了思路. 神经网络验证指的是判断训练好的神经网络模型是否满足某些特定的规约性质, 比如安全性 (safety)[52], 鲁棒性 (robustness)[14], 公平性 (fairness)[53]和可达性 (reachability)[54,55]等, 学术界提出了多种验证技术, 包括基于抽象解释 (abstract interpretation) 的[22−26], 基于可满足性模理论 (satisfaction module theory, SMT) 的[27−30]和基于凸优化理论 (convex optimization theory) 的[31−34]验证技术等, 对多种神经网络模型进行了验证. 与软件测试类似, 神经网络测试是通过大量的测试用例来评估神经网络的行为, 判断是否存在错误行为, 近几年在测试的覆盖标准[43]和用例生成[44,45]方面取得了重大进展. 然而, 神经网络验证和神经网络测试都是事后分析, 也就是说, 它们只能提供错误行为是否存在的定性或者定量结论, 当神经网络不满足某个性质时, 神经网络验证或者测试的结果并不能缓解这些违背情况. 相反, 基于性质满足的深度神经网络训练方法抛开了预训练模型, 专注于从头开始构建并训练一个满足特定性质的神经网络 (满足程度可以通过神经网络验证或者测试进行评估), 尤其是基于鲁棒性质的深度神经网络的训练方法获得了众多关注并取得了重大突破, 感兴趣的读者可以阅读文献 [50] 以更全面地了解该领域的进展.

无论是神经网络验证、神经网络测试还是基于性质满足的神经网络训练, 当面对一个具有错误行为的神经网络模型时, 它们都将失效, 也就是说这几种方法对于神经网络模型已经表现出的错误行为无能为力. 为了解决这个问题, 深度神经网络修复 (DNN repair)[56−77]应运而生, 即当发现神经网络表现出错误行为时 (通过神经网络验证或测试技术实现), 神经网络修复旨在消除这些错误行为, 使修复后的神经网络满足特定的规约性质, 同时最大限度地减少对网络模型性能的影响.

本文将主要介绍重训练 (retraining)、无错误定位的微调修复 (fine tuning without fault localization) 和包含错误定位的微调修复 (fine tuning with fault localization) 这 3 类常见的神经网络修复策略, 并对不同修复方法之间的异同进行分析和梳理. 本文第 2 节介绍了必要的基础知识并给出了神经网络修复问题的形式化定义和常见的修复要求. 第 3 节介绍了近几年提出的 3 类常见深度神经网络修复策略的核心思想, 包括基于反例制导的、基于神经网络验证的、基于优化问题的、基于约束求解的、基于搜索的、基于函数空间优化的修复策略, 并对不同的修复策略进行了比较和分析. 第 4 节介绍了目前常见的神经网络修复策略的评价指标和测试基准. 第 5 节展望了神经网络修复策略领域未来可能的研究方向. 第 6 节对全文内容进行了总结. 图 1 从年份和核心思想两个维度列举了本文介绍的神经网络修复策略, 其中重训练、无错误定位的微调修复和包含错误定位的微调修复分别用红色、绿色和蓝色矩形框展示. 需要注意图 1 的分类结果只考虑各个修复策略的主要创新, 不是严格唯一的.

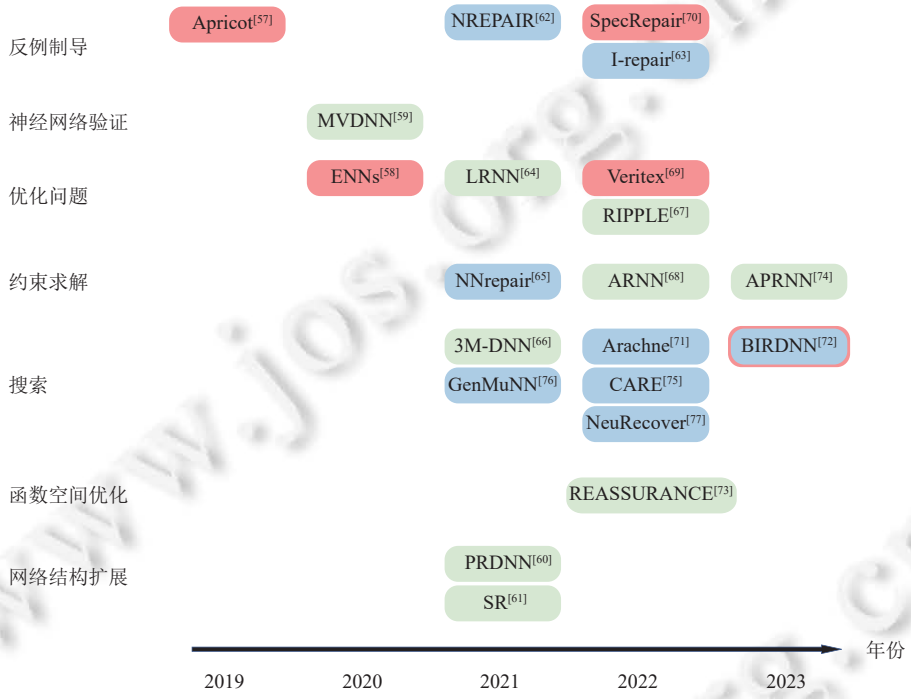

图 1 主要神经网络修复策略展示

2 神经网络修复

本节在补充介绍神经网络及其规约性质的基础上, 介绍了神经网络修复问题的定义和常见的神经网络修复要求. 此外, 为了更好地理解和辨清神经网络修复问题, 本节比较讨论了神经网络修复与其他相近概念的异同.

2.1 神经网络及规约性质

通常来说, 一个 L 层深度神经网络 N 包含一个输入层 (input layer), 一个输出层 (output layer) 和若干个连续的隐藏层 (hidden layer)[78], 分别为 $N^{l_0}, N^{l_1}, \ldots, N^{l_{L-1}}$, 其计算过程近似了一个高度复杂的函数 $N(\cdot): \mathbb{R}^m \to \mathbb{R}^n$, 其中 m 和 n 分别是输入层和输出层神经元的个数, 也称作输入/输出维度 (input/output dimension)[79,80]. 神经网络中相邻的两个网络层之间一般包含一个仿射变换 (affine transformation) 和一个非线性激活函数 (nonlinear activation function), 最后两层之间一般只有仿射变换. 相邻两层神经元之间边上的权重 (weight) 和偏置 (bias) 实现了仿射变换, 而神经网络对于高度复杂函数的近似能力则来源于非线性激活函数的使用, 常见的激活函数有 ReLU, tanh 和 Sigmoid 等, 其中 $\mathrm{ReLU}(x) = \max(0, x)$ 是分段线性的, tanh 和 Sigmoid 则是严格非线性的. 一般用 θ 统一指代神经网络参数,

并把具有这一组参数的神经网络记作 N_θ. 神经网络的训练过程是基于训练数据集 (training dataset) D 按照深度学习算法对神经网络的参数 θ 进行调整更新, 以期在训练数据集上获得较好的拟合能力, 并且在测试数据集 (testing dataset) D_T 上具有较好的泛化能力[78].

在部署到实际应用之前, 特别是在安全攸关领域, 深度神经网络需要满足一定的性质, 如分类正确性、鲁棒性、安全性、可达性等. 因此, 神经网络性质的形式化描述应运而生, 并被广泛应用于基于性质满足的神经网络训练和神经网络验证等相关领域. 神经网络 N_θ 的一个规约性质 $\mathcal{P}$ 通常由 3 个部分组成, 一个输入集合 X, 一个输出集合 Y 和这两个集合之间的映射关系 ϕ, 即 $\mathcal{P}:=(X;Y;\phi)$. 输入集合 X 是输入域 $\mathbb{R}^m$ 上的约束, 它描述了 N_θ 的一个输入的可行域, 输出集合 Y 则给出了 $\mathbb{R}^n$ 空间中的一个输出范围. 映射要求 ϕ 对 N_θ 在输入集合 X 和输出集合 Y 之间施加了一个行为要求, 当映射关系 ϕ 成立时, 神经网络 N_θ 满足性质 $\mathcal{P}$; 否则神经网络 N_θ 违背性质 $\mathcal{P}$. 特别地, 把样本 $x_p\in\{x|x\in X,N(x)\in Y\}$ 和 $x_n\in\{x|x\in X,N(x)\notin Y\}$ 分别称作性质 $\mathcal{P}$ 的正样本 (positive sample, 或者正例) 和负样本 (negative sample, 或者反例), 其中 $N(x)$ 是神经网络 N_θ 关于样本 x 的输出结果. 包含反例的性质 $\mathcal{P}$ 在神经网络 N_θ 上不成立, 网络 N_θ 关于性质 $\mathcal{P}$ 是一个待修复的错误神经网络 (buggy/defective DNN). 为了描述的简洁性, $N_\theta\vDash(X;Y;\phi)$ 表示神经网络 N_θ 满足性质约束 $\mathcal{P}$, 而 $N_\theta\nvDash(X;Y;\phi)$ 表示规约性质 $\mathcal{P}$ 不适用神经网络 N_θ. 表 1 给出了常见的几种神经网络规约性质的形式化描述, 此处区域以多面体 (polytope) 的半空间法 (H-representation)[81]进行举例表示.

表 1 常见神经网络规约性质举例

性质类型	输入集合	输出集合	映射要求
分类正确性	$X=\{x_1,x_2,\ldots,x_n\}$	$Y=\{y_1,y_2,\ldots,y_n\}$	$\arg\max N(x_i)=y_i$
安全性	$X=\{x\|Ax\leqslant b\}$	$Y=\{y\|Cy\leqslant d\}$	$N(x)\in Y,\ \forall x\in X$
鲁棒性	$X=\{x-\bar{x}\leqslant\varepsilon\}$	$y=\arg\max N(\bar{x})$	$N(x)=y,\ \forall x\in X$
可达性	$X=\mathbb{R}^m$	$Y=\{y\|Cy\leqslant d\}$	$\forall y\in Y,\ \exists x\in X,\ N(x)=y$

对于神经网络 N_θ 的一个规约性质集合 $\{\mathcal{P}_i\}_{i=1}^p:=\{(X_i;Y_i;\phi_i)\}_{i=1}^p$, $N_\theta\vDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p$ 和 $N_\theta\nvDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p$ 分别表示 $\{\mathcal{P}_i\}_{i=1}^p$ 在 N_θ 上的可满足性和不满足性, 分别定义为 $N_\theta\vDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p\Leftrightarrow N_\theta\vDash(X_i;Y_i;\phi_i),\forall i\in\{1,2,\ldots,p\}$ 和 $N_\theta\nvDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p\Leftrightarrow N_\theta\nvDash(X_i;Y_i;\phi_i),\exists i\in\{1,2,\ldots,p\}$. 也就是说, 规约性质集 $\{\mathcal{P}_i\}_{i=1}^p$ 在神经网络 N_θ 上成立当且仅当集合中的每个性质都在 N_θ 上成立, 而不成立意味着该规约集合中至少有一个性质被 N_θ 违反.

2.2 神经网络修复问题

基于上述的背景知识, 神经网络的修复问题定义如下.

定义 1. 神经网络修复问题 (DNN repair problem). 给定一个待修复的初始神经网络 N_θ 以及它所违背的规约性质集 $\{\mathcal{P}_i\}_{i=1}^p$, 即 $N_\theta\nvDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p$. 神经网络修复问题是指通过某种修复手段将参数集 θ 修改为 θ', 得到的修复后的神经网络 $N_{\theta'}$ 满足约束性质集合, 即 $N_{\theta'}\vDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p$.

值得注意的是, 上面的定义没有显式地要求初始待修复网络 N_θ 和修复后的网络 $N_{\theta'}$ 必须保持同样的网络结构, 这意味着修复策略可以保持原有网络结构, 只调整网络参数; 也可以在调整参数的同时, 基于原有的网络结构进行额外的结构设计或扩展, 从而完成神经网络修复. 目前已有的修复方法中, 这两种修复类型都存在并且也都取得了一定的成效.

因为神经网络规约性质及其描述类型的不同, 近几年出现了各种更具体的神经网络修复问题. 从其所修复的网络性质来看, 神经网络修复问题可以分为神经网络安全性修复 (DNN safety repair), 神经网络公平性修复 (DNN fairness repair), 神经网络鲁棒性修复 (DNN robustness repair) 等. 本文重点介绍另一种更为一般和简洁的分类方式, 即按照规约性质所描述的输入/输出集合的类型进行划分, 分为基于样本的神经网络修复问题 (sample-wise DNN repair problem, SRP) 和基于区域的神经网络修复问题 (domain-wise DNN repair problem, DRP), 它们的定义如下.

定义 2. 基于样本的神经网络修复问题 (sample-wise repair problem, SRP). 给定一个待修复的神经网络 N_θ 以及它所违背的规约性质集 $\{\mathcal{P}_i\}_{i=1}^p$, 即 $N_\theta\nvDash\{(X_i;Y_i;\phi_i)\}_{i=1}^p$, 其中每个性质 $\mathcal{P}_i$ 的输入集合 X 和输出集合 Y 描述了相应空间内的样本点的集合. 神经网络样本修复问题是指通过某种修复手段将参数集 θ 修改为 θ', 得到的修复后的神经

网络 $N_{\theta'}$ 满足这些以样本点形式描述的约束性质集合, 即 $N_{\theta'} \vDash \{(X_i;Y_i;\phi_i)\}_{i=1}^{p}$.

定义 3. 基于区域的神经网络修复问题 (domain-wise repair problem, DRP). 给定一个待修复的神经网络 N_θ 以及它所违背的规约性质集 $\{\mathcal{P}_i\}_{i=1}^{p}$, 即 $N_\theta \nvDash \{(X_i;Y_i;\phi_i)\}_{i=1}^{p}$, 其中每个性质 $\mathcal{P}_i$ 的输入集合 X 和输出集合 Y 描述了相应空间内的一个区域 (如区间, 多面体等). 神经网络区域修复问题是指通过某种修复手段将参数集 θ 修改为 θ', 得到的修复后的神经网络 $N_{\theta'}$ 满足这些以区域形式描述的约束性质集合, 即 $N_{\theta'} \vDash \{(X_i;Y_i;\phi_i)\}_{i=1}^{p}$.

此外, 将同时包含了样本点和区域的修复问题也分类为基于区域的神经网络修复问题, 这是因为修复区域内样本的无穷性和不可遍历性使得区域修复相较于样本修复更具挑战性. 为了表达简洁, 后文将初始待修复神经网络 N_θ 和修复后的神经网络 $N_{\theta'}$ 分别简记为 N 和 N', 基于样本 (区域) 的神经网络修复问题也会表述为神经网络样本 (区域) 修复问题.

2.3 神经网络修复要求

对于一个深度神经网络修复策略, 除了达到基本的修复目的之外, 在修复过程中还需要考虑其他的要求, 比如局部性、泛化性、效率等, 表 2 展示了主要的深度神经网络修复要求及其概念内涵.

表 2 常见神经网络修复要求

修复要求	概念内涵
性能 (efficacy)	修复后的神经网络 N' 能够满足原神经网络 N 所违背的规约性质, 正确处理 N 不能正确处理的输入样本或区域
最小性 (minimality)	修复后的神经网络 N' 要尽可能保持原神经网络 N 的性能, 当产生了性能损失时, 修复策略需要尽可能缩小这种性能衰退. 当然, 如果修复的同时产生了神经网络性能的提升, 那更好不过
泛化性 (generalization)	对于那些与待修复样本 (区域)相近的输入样本 (区域), 修复后的神经网络 N' 拥有较好的泛化能力, 也能够正确处理这些样本(区域)
局部性 (locality)	对于那些与待修复样本 (区域)相差较大的输入样本 (区域), 修复策略能够尽可能不改变原神经网络在这些样本(区域)上的行为, 即修复神经网络 N' 能够与原神经网络 N 表现接近
可扩展性 (scalability)	修复策略能否支持较高维度描述的待修复样本或者区域, 比如多面体的维度等
可证明性 (provability)	修复策略能否从理论上保证修复后的神经网络 N' 完全满足原始网络 N 所违背的规约性质
效率 (efficiency)	修复策略应该具有较高的计算效率

需要说明的是, 这些评价指标各有侧重, 它们之间并不是完全分割和独立的. 比如修复策略的可扩展性在一定程度上也反映了它们的效率和性能, 可证明性在一定程度上限制了修复策略的可扩展性, 而修复策略的最小性和局部性也有部分意义上的重叠. 第 3 节列举的神经网络修复策略或框架不同程度地考虑了以上修复要求的一个子集.

2.4 相近概念的讨论和区别

这一部分比较和区分了神经网络修复与神经网络程序修复、神经网络数据集修复、传统软件程序修复和神经网络辅助的程序修复等相近的概念.

2.4.1 神经网络修复与神经网络程序修复

神经网络程序修复 (DNN program repair) 指的是修复一个使用深度学习框架编写的报错的神经网络程序中存在的漏洞 (bug)[82−84], 其本质还是程序修复, 而神经网络修复是对神经网络程序正确运行后的神经网络产品进行修复. Islam 等人[84]使用由来自 Stack Overflow 的 415 个神经网络程序漏洞和来自 GitHub 的 555 个漏洞组成的深度神经网络程序漏洞数据集对 5 个深度学习库 Caffe、Keras、TensorFlow、Theano 和 PyTorch 的漏洞修复模式进行了全面研究, 分析了神经网络程序修复模式的以下几个方面: (1) 常见的漏洞修复模式; (2) 修复多种漏洞类型的模式; (3) 跨深度学习库的漏洞的修复模式; (4) 修复风险; (5) 修复神经网络程序漏洞的挑战.

2.4.2 神经网络修复与神经网络数据集修复

神经网络数据集修复 (DNN dataset repair) 是指通过使用一些技术来修复损坏或不完整的神经网络数据集. 在复杂的神经网络任务中, 数据集通常需要进行归一化、标准化、去噪或者与其他数据集进行集成来提高数据质量[85,86].

因此, 数据集修复是神经网络技术的重要组成部分, 可以帮助提高神经网络的准确性和可靠性. 常见的数据集修复技术包括数据清洗、数据采样、数据增强、数据集合等. 与神经网络的数据集修复不同, 在神经网络修复的问题设定下, 数据集是假设正确的, 问题的核心在于神经网络不能正确处理某些输入样本得到预期的输出结果.

2.4.3 神经网络修复与传统软件修复

传统软件修复 (software repair) 是修复程序中存在的漏洞, 使软件输出正确的预期的输出结果[87−89]. 与传统软件修复相比, 神经网络修复有着更大的区别. 当修复软件中的漏洞时, 常规做法是对已知的错误代码进行细致检查, 找出具体问题, 并对其进行适当的修改. 这些问题通常是由于代码逻辑错误、浮点计算误差以及其他各种常见错误引起的. 一旦问题被发现, 程序员可以快速修复缺陷. 此外, 软件开发人员有众多工具可以帮助他们识别和修复漏洞, 例如调试器和自动化测试工具等. 然而, 神经网络修复是一个更加复杂的过程. 神经网络模型的行为是由其参数确定的, 但参数并不是手动编写, 而是根据学习算法自动从训练数据中习得. 因此, 在修复神经网络模型的缺陷时, 需要更多技巧来找到正确的解决方案. 可能还需要通过调整网络模型结构、更改训练算法和优化训练过程等方法来纠正错误行为, 使模型能够更好地拟合数据. 但是, 成熟的传统软件修复技巧为神经网络的修复提供了很好的参考和指导.

2.4.4 神经网络修复与神经网络辅助的程序修复

神经网络在神经网络辅助的程序修复 (DNN aided program repair) 和神经网络修复两个领域中承担不同角色. 神经网络辅助的程序修复是近年来软件工程从业者利用神经网络来修复程序中漏洞的研究领域[90−92], 神经网络作为一种新型的修复工具被引入到传统软件修复中. 而神经网络修复则是指修复表现出错误行为的神经网络, 神经网络是待修复对象, 一般借助于软件工程领域的方法和工具进行神经网络修复.

3 神经网络修复策略的研究现状

目前已有的神经网络的修复策略或者框架主要分为以下 3 类: 重训练、无错误定位的微调修复和包含错误定位的微调修复. 本节将近几年来的神经网络修复策略相关的工作按照这 3 种大的类别进行组织和概述, 此外每小节对每一类方法进行了更为细致的对比, 使得修复策略的综述更加系统和容易理解. 接下来将详细介绍具体的神经网络修复策略.

本节所介绍的修复策略的总体对比展示在表 3 中, 分别从修复类型 (支持求解样本修复问题或区域修复问题)、支持修复的网络性质、支持修复的神经网络所采用的激活函数、修复策略的可证明性、对于原有网络结构的保持性、支持求解修复问题的规模以及局限性等几个方面进行了总体比较.

表 3 神经网络修复策略对比

范式	方法	修复类型		修复性质	激活函数		可证明性	保持结构	规模	局限
		样本	区域		ReLU	其他				
重训练	Apricot	√	—	分类准确性	√	√	—	√	大	—
	ENNs	√	—	多种性质	√	√	—	√	大	—
	Veritex	—	√	安全性	√	—	Δ	√	较小	精准可达集计算
	SpecRepair	—	√	多种性质	√	√	√	√	中等	反例检测
	BIRDNN*	—	√	多种性质	√	√	—	√	大	—
无错误定位的微调	MVDNN	√	—	多种性质	√	—	√	√	较小	验证器可扩展性
	3M-DNN	√	—	分类准确性	√	√	—	√	中等	网络划分和合并
	PRDNN	√	√	多种性质	√	☆	√	—	较小	线性区域计算
	LRNN	—	√	多种性质	√	—	√	√	较小	二次优化求解
	SR	—	√	安全性	√	√	√	—	中等	约束组合与求解
	REASSURANCE	—	√	多种性质	√	—	—	—	中等	补丁函数的计算
	RIPPLE	√	—	分类准确性	√	—	√	√	中等	线性约束求解
	ARNN	—	√	安全性	√	—	√	√	较小	SMT求解器
	APRNN	—	√	可达性	√	※	√	√	较小	线性区域计算

表 3 神经网络修复策略对比 (续)

范式	方法	修复类型		修复性质	激活函数		可证明性	保持结构	规模	局限
		样本	区域		ReLU	其他				
包含错误定位的微调	NREPAIR	—	√	多种性质	√	√	□	○	中等	输入区域划分
	GenMuNN	√	—	分类准确性	√	√	—	√	中等	突变产生和选择
	NNrepair	√	—	分类准确性	√	√	—	○	较小	符号执行
	Arachne	√	—	分类准确性	√	√	—	√	大	—
	CARE	√	√	多种性质	√	√	—	√	大	—
	NeuRecover	√	—	分类准确性	√	√	—	√	中等	模型历史分析
	I-repair	√	—	分类准确性	√	√	□	√	中等	循环修复过程
	BIRDNN	—	√	多种性质	√	√	—	√	大	—

注: Δ: Veritex修复的可证明性当距离损失函数项的值为0时达到; *: BIRDNN修复框架同时支持重训练修复和包含错误定位的微调修复, 因此在两部分分别进行了比较; ☆: PRDNN的样本修复可以支持其他激活函数, 而区域修复不支持; ※: APRNN支持具有线性部分的激活函数; □: 超时之前结束返回可证明的修复结果, 否则是终止前最优的修复结果; ○: 虽然修复策略保持神经网络结构, 但是会返回一个有同样网络结构的神经网络集合

3.1 基于重训练的修复策略

为了更好地理解重训练的神经网络修复方法, 首先需要和基于性质满足的深度神经网络训练方法进行区分. 基于性质满足的神经网络训练方法专注于从头构建和训练满足特定规约性质的神经网络, 而重训练的修复方式则是在原有的网络模型基础上继续进行神经网络训练. 从参数更新空间上来看, 基于性质满足的神经网络训练有更大的参数搜索空间, 而基于重训练的神经网络修复则受限于修复的要求 (比如局部性, 最小性等), 其参数搜索空间相对更小.

3.1.1 Apricot[57]

Zhang 等人[57]提出的 Apricot 修复框架旨在通过权重自适应方法迭代地修复神经网络模型, 其核心思想在于如果在原始训练数据集的许多不同子集上训练深度学习的神经网络模型, 则所得简化深度学习模型 (reduced deep learning model, rDLM) 中的权重可以提供关于原始模型 N_θ 权重调整方向和大小的见解, 以处理原始网络模型错误分类的测试用例. 这种方法基于两个观察结果, 首先随着数据集 D_0 规模的增加, 神经网络模型将越来越难以保留大部分权重来捕获所有基本特征. 第 2 个观察是, 考虑一对神经网络, 分别记为 N_0 和 N_{sub}, 它们的训练过程相同, 只是模型 N_0 是在整个数据集 D_0 上训练, 而模型 N_{sub} 是在 D_0 的一个子集 D_{sub} 上训练, 模型 N_{sub} 就是简化深度学习模型. 一般来说每个单独的简化深度学习模型可能不能完全捕获正确分类一个特定测试用例所需的基本特征, 但这组简化深度学习模型的集中趋势可能会正确地对测试用例进行分类.

后文图 2 展示了 Apricot 修复策略的工作流程: 给定神经网络模型 N_0 和训练数据集 D_0, Apricot 首先生成一组简化深度学习模型. 然后以 N_0 作为第 1 次迭代的输入深度学习模型 (input DLM, iDLM). 在第 k 次迭代时, 对于 N^k 的每个失败测试用例 x, Apricot 将简化深度学习模型集划分为两个类别, 分别表示为 $IncorrectSubModel(x)$ 和 $CorrectSubModel(x)$, 这两个类别中的每个简化深度学习模型分别对样本 x 进行了错误和正确的分类. 然后, 对于 N^k 的权重 w, Apricot 取两个类别中简化网络模型的相应权重的集中趋势来调整 w. 接下来, 它用 D_0 训练调整后的 N^k, 以产生 $k+1$ 次迭代步骤的下一个输入模型 N^{k+1}. 在所有迭代完成后, Apricot 输出一个具有权重自适应的神经网络模型, 即修复神经网络 $N_{\theta'}$.

3.1.2 可编辑神经网络 (editable neural networks, ENNs)[58]

目前大部分的神经网络, 对单个输入样本的预测要依赖所有的模型参数, 因此针对特定的输入对神经网络进行行为纠正很难不影响模型在其他输入上的性能表现. 基于此, Sinitsin 等人[58]提出可编辑神经网络以及与之相对应的可编辑训练 (editable training) 的神经网络训练方法. 可编辑训练采用了新的目标函数 $Obj(\theta, l_e) = L_{\text{base}}(\theta) + c_{\text{edit}} \cdot L_{\text{edit}}(\theta) + c_{\text{loc}} \cdot L_{\text{loc}}(\theta)$. 训练过程是多目标优化的过程, 其中, 基底目标函数 $L_{\text{base}}(\theta)$ 保证初始模型为 N_θ 的训练; 编辑训练目标函数 $L_{\text{edit}}(\theta) = \max(0, Edit(\theta, l_e))$, 其中 $Edit(\theta, l_e)$ 被称为编辑函数, 目标在于在训练过程中调整参数集 θ 使

得 $l_e(\theta) \leqslant 0$, $l_e(\theta)$ 则是根据具体任务设计的修复目标函数, 比如对于分类任务, $l_e(\theta) = \max_{y_i} \log p(y_i|x,\theta) - \log p(y_{ref}|x,\theta) \leqslant 0$ 保证了 y_{ref} 是网络的预期最大输出, 即分类结果. 局部性目标函数 $L_{\text{loc}}(\theta) = E_{x\sim p(x)} D_{KL}(p(y|x,\theta) \| p(y|x, Edit(\theta, l_e)))$ 则保证了修复的局部性, 即尽量保证修复后的神经网络 $N_{\theta'}$ 在其他输入样本上的性能表现不受影响. 超参数 c_{edit}, c_{loc} 设置充分小, 使得训练过程中保证 $L_{\text{base}}(\theta)$ 受到较小的影响, 同时也保证网络修复的局部性. 因为网络参数的冗余性, 可以保证上述目标函数得到优化.

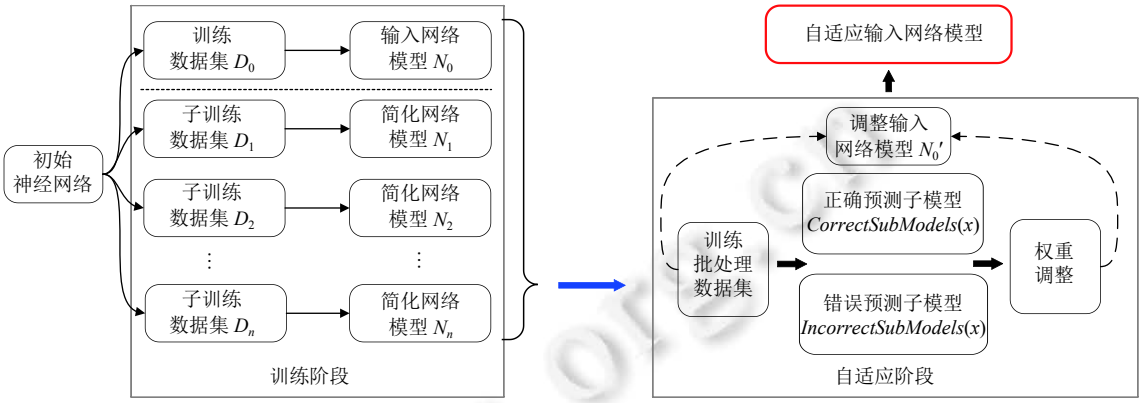

图 2 Apricot 神经网络修复策略工作流程

图 3 展示了可编辑神经网络的前向传播和反向传播过程. 编辑函数采用梯度下降实现, 作者也找到了一些使得权重更新更加鲁棒且改善局部性的方法, 比如采用 RPROP[93]、signSGD[94]、RMSProp[95]等弹性反向传播 (resilient backpropagation) 的参数更新策略.

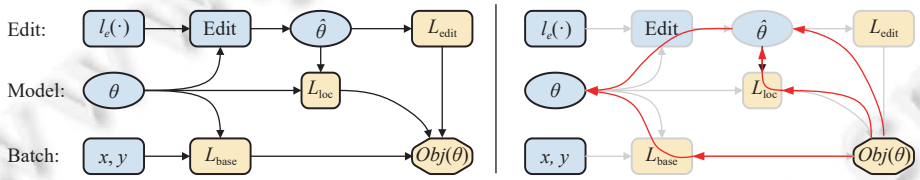

图 3 可编辑神经网络的训练过程

3.1.3 Veritex[69]

Veritex 是 Yang 等人[69]提出的一种利用可达性分析来修复安全攸关系统中不安全的 ReLU 神经网络的方法. Veritex 首先通过可达性分析来计算神经网络的精确的不安全的多面体可达域 (这也是激活函数限定为 ReLU 函数的原因)[96], 然后在重训练过程中使用一个损失函数项来表示不安全可达域与安全区域的距离 $L_{\text{dist}}(\theta)$. 由于神经网络参数的细微变化可能导致意想不到的模型性能下降, 因此 Veritex 在最小化可达域与安全域距离的同时, 也考虑了另外一个损失函数项 $L_{\text{diff}}(\theta, \theta')$ 来追求最小的神经网络修复, 损失项 L_{diff} 主要是优化原有网络 N_θ 和修复后的神经网络 $N_{\theta'}$ 在原有数据集上的输出差距. 最终重训练过程采用两项损失函数的线性组合来进行网络参数更新. 此外, Veritex 修复策略还可以与深度强化学习 (deep reinforcement learning, DRL) 算法[97]集成来修复神经网络智能体 (DNN agent) 的错误行为.

3.1.4 SpecRepair[70]

Bauer-Marquart 等人[70]提出的 SpecRepair 是重训练修复范式的又一个代表性工作, 其核心思想在于将安全规约性质转换为目标函数, 该目标函数对于所有违反性质的网络输入都是负值, 然后使用全局优化方法检测反例. 接着利用这些检测到的反例, 通过惩罚性的重训练使得神经网络变得安全. SpecRepair 专注于神经网络安全性修复, 构造了一个原始网络损失函数 $L_{\text{base}}(\theta)$ 与安全性质满足性 $L_{\text{sat}}(\theta)$ 相结合的目标函数来指导重训练过程. SpecRepair

将违反安全性的反例生成问题转化为优化问题进行求解. 基于反例和神经网络原有的训练数据, 框架设计了一种自动修复机制, 该机制使用上述损失函数在原有神经网络的基础上进行额外的训练迭代, 追求在保持高精度的同时消除错误行为. 重训练完成后, 反例生成组件再次执行. 如果没有反例被检测到, 神经网络验证器会检查修复网络是否遵从所要求的安全性质. 由于反例搜索是快速但不完整的, 验证器可能仍然会找到反例, 在这种情况下, SpecRepair 会重新返回到修复组件. 否则, 修复后的神经网络 $N_{\theta'}$ 被 SpecRepair 验证并返回.

3.1.5 重训练方法对比分析

在表 4 中, 从核心思想、性质改进、性能保持和特点等 4 个方面对基于重训练的神经网络修复策略进行了总结和比较, 其中 BIRDNN 框架的重训练修复策略的相关介绍参见本文第 3.3.9 节. 可以看到基于重训练的神经网络修复策略的核心在于构造一个可以更好兼顾改进被违背的规约性质和保持原有性能的损失函数. 损失函数的设计极大影响了修复策略的评估, 比如 Veritex 修复策略需要计算准确可达集, 这就导致其对激活函数的高要求和较差的可扩展性. 一般而言, 重训练修复方法的计算代价比较高昂, 尤其是大规模神经网络的训练可能需要耗费数日. 此外原始神经网络的训练数据集也并不总是公开可用的, 这也为重训练修复带来一定的困难和挑战.

表 4 重训练神经网络修复方法对比

方法	核心思想	性质改进	性能保持	特点
Apricot	权重自适应方法迭代修复神经网络模型	模型组平均趋势	原始网络损失函数	权重修改过程中有明确方向和大小
ENNs	可编辑神经网络及可编辑训练的训练方法	编辑目标函数	原始网络损失保持性能; 函数局部性目标函数保证了修复的局部性	编辑训练函数允许多种修复性质
Veritex	可达性分析	最小化不安全可达域与安全区域的距离	最小化原有网络和修复网络在原有数据集上的输出差距	可集成深度强化学习
SpecRepair	将安全规约性质转换为目标函数	基于反例的惩罚性训练	原始网络损失函数	后端集成验证器
BIRDNN	行为模仿	模仿正样本的输出给负样本分配标签	最小化原有网络和修复网络在原有数据集上的输出差距	概率近似正确的方式修复网络模型

3.2 无错误定位的微调修复策略

在重训练修复策略中, 全局性且没有区分的参数更新可能会导致神经网络的行为发生剧烈的变化, 因此, 神经网络微调修复致力于局部地、有倾向地、轻微地修复有缺陷的神经网络, 即修补神经网络参数的特定子集, 以消除神经网络的错误预测.

3.2.1 MVDNN (modification with verification for DNNs)[59]

MVDNN 试图将神经网络的修复问题转化为神经网络验证问题, 借助目前更加成熟的验证器来进行可证明的神经网络修复. 在修改权重的过程中, MVDNN 限制了对神经网络进行更改的参数的数量和修改值的大小. 具体来说, MVDNN 的目标确保修复后的神经网络 $N_{\theta'}$ 满足给出的规约性质的同时最小化原始网络 N_{θ} 和修复网络 $N_{\theta'}$ 之间的参数差异. 这很容易借助于神经网络验证表示为一个优化问题, 并通过现有的 SMT 验证工具来解决. 但是基于 SMT 的神经网络验证工具一般可扩展性较差, 因此需要进一步降低优化问题的复杂度, 使这种修复方法可以应用于大型网络, 为此 MVDNN 将修改的网络权重限制在单层的权重. 然而, 即使当神经网络修复问题仅限于单层权重修改时, 针对较大输入区域的网络性质也会使得到的神经网络验证问题的求解变得困难. 因此, Goldberger 等人[59]进一步将单层修复限制在网络的输出层修复上, 此时神经网络验证问题将变成复杂度更低的线性规划 (linear programming) 问题 (输出层不包含激活函数). 直观地说, 仅更改输出层的权重会使得验证问题完全线性化, 这意味着它可以使用线性规划求解器来解决, 这往往比其他神经网络验证工具更具有可扩展性.

3.2.2 3M-DNN (minimal multi-layer modifications of DNNs)[66]

3M-DNN 的关键思想是依据某些分离层 (separation layer) 将原始神经网络 N 分成多个子网络, 然后尝试分别为每个子网络找到最小的修复补丁, 从而给初始网络 N 带来期望的整体变化. 事实上, 修改神经网络的多层可能会

产生许多难以预测的复杂影响 (尤其是可能引入新的错误行为), 而仅改变网络的一层可能又不足以完全修复网络. 因此, Refaeli 等人[66]试图将多层神经网络修复问题分解为一系列更容易解决的单层修复问题. 3M-DNN 由两个逻辑层次组成, 即顶层的搜索和底层的单层修复. 3M-DNN 首先对分离层可能的修改值进行启发式搜索, 文章特别研究了 3 种不同的启发式: 随机搜索 (random search)、贪婪搜索 (greedy search) 和蒙特卡罗树搜索 (Monte Carlo tree search, MCTS)[98], 理论上其他搜索方式也是可行的. 在使用启发式搜索找到每个分离层的修改值的指派后, 3M-DNN 将计算使用该修复方案的总体修复代价 (通过合并更改每个单独子网络来计算). 3M-DNN 无法保证每次都能获得期望结果的修改方案, 因此 3M-DNN 进行下一次迭代来进一步优化探索分离层值的不同变化, 并记录到目前为止遇到的最小修复代价的修复方案. 3M-DNN 是随时可用的, 无论何时停止, 它都会返回到目前为止发现的最佳 (最小) 更改.

3.2.3 PRDNN (provable repair of DNNs)[60]

Sotoudeh 等人[60]提出的 PRDNN 是一种可证明的神经网络样本修复和区域 (特指多面体) 修复技术, 同时, PRDNN 可以保证原有网络 N 和修复网络 N' 的差距 (参数距离) 是单层修复中最小的. 针对于样本修复问题, 即使是单层神经网络的可证明修复也是一个 NP-hard 问题, 在实际求解过程中可行性相对较低[59]. 然而, 如果神经网络的最后一层是线性的, 那么网络修复则将转换成一个线性规划问题, 可以在多项式时间内解决. PRDNN 样本修复对神经网络使用的激活函数没有限制, 其关键见解是引入了一种新的深度神经网络架构, 称为解耦深度神经网络 (decoupled deep neural networks, DDNNs), 每个深度神经网络都存在一个等效的解耦深度神经网络, 并且修复解耦深度神经网络中的任何单层都可以简化为线性规划问题[59]. 针对于分段线性神经网络的多面体区域修复, 网络行为在每个线性区域内保持凸性, 凸性保证任何给定的多面体当且仅当其顶点被映射到另一个多面体的顶点, 因此多面体修复问题可以简化为样本修复问题, 从而克服了输入区域中样本点的无限性和不可遍历性. 值得注意的是, 激活函数分段线性的假设保证了线性区域的计算, 但是, 由于计算神经网络的线性多面体区域的计算复杂性[99], 修复多面体的维度通常只有一维或二维是可行的; 将得到的解耦深度神经网络转换得到标准和等效的深度神经网络依然是一个亟待解决的问题.

3.2.4 LRNN (local repair of neural networks using optimization)[64]

LRNN 是 Majd 等人[64]提出的一种神经网络修复框架, 使用混合整数二次规划 (mixed-integer quadratic program, MIQP)[100]来生成满足给定神经网络规约性质的网络修复方案. LRNN 将规约性质表述为一组谓词, 这些谓词在修复区域上对神经网络的输出施加约束, 同时将神经网络修复问题定义为一个混合整数二次规划, 以根据给定的谓词调整单层神经网络的权重, 同时最小化原始训练区域上的原始损失函数. 此外, 由于输出层不包含非线性激活函数, 因此对神经网络的输出层应用修复不会产生混合整数约束, 也就是说, 输出层的最小修复问题是一个更易求解的二次规划 (quadratic program, QP) 问题.

3.2.5 SR (self-repairing Transformer)[61]

自修复转换器是一个以给定一组分类性质和待修复神经网络 N 作为输入, 产生一个新的神经网络 N' 的函数. 新产生的网络 N' 需要满足以下两个性质: 首先, 对于每个规约性质, 如果 x 在性质的定义域内并且存在满足该性质的 y, 那么 $N'(x)$ 也将满足该性质; 其次, 当且仅当 x 不在性质域内或不存在满足性质的值时, $N'(x)$ 将返回无效. 由于神经网络是由一系列网络层组成的, Leino 等人[61]在原有网络添加了一个额外的网络层, 称为自我修复层 (self-repair layer, SR-Layer). 自我修复层接收原始输入 x 和原始神经网络的输出 $N(x)$, 它或者返回无效 (invalid, 输入不在性质区域内), 或者产生满足给定性质的输出 (输入在性质区域内). 为了产生满足给定性质的输出, 如果输入 x 在原始网络中满足规约性质, 那么自我修复层直接输出原始网络的输出 $N(x)$, 否则自我修复层尝试找到一个在输出层上可满足的约束, 该约束是许多单一约束的不同组合, 使得 x 对应的输出满足所有性质要求. 之后, 自我修复层检查创建的约束是否可满足, 如果单一约束都不能满足, 那么算法将判别原始网络无法被修复, 返回无效. 否则, 自我修复层根据前面得到的约束对输出进行排序, 选择最大程度保持原始网络输出的约束来完成网络修复. 图 4 展示了 SR 修复策略的简要流程.

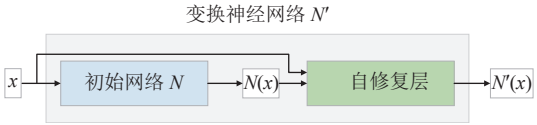

图 4 SR 神经网络修复策略流程

3.2.6 REASSURANCE[73]

与一般的神经网络权重修改策略不同, REASSURANCE 修复方法在函数空间进行神经网络修复以满足被违背的网络性质. 如图 5 所示, 该方法针对 ReLU 神经网络修复提出了两个技巧: 补丁函数 (patch function) 和支撑网络 (support network). 补丁函数是指为一个待修复的线性区域 A 寻找一个函数 $p_A=cx+d$ 使得对于多面体区域 $A=a_ix\leqslant b_i, i\in I$ 中的任意样本 x, $f(x)+p_A(x)$ 满足规约性质. 其中 $f(x)$ 是原始神经网络的输出, 即 $N(x)$. 为了避免求解过程中枚举多面体的顶点, 因此 Fu 等人[73]考虑了规约性质表达为网络输出限制在某个范围内的情况, 得到了更易求解的线性规划问题. 补丁函数可以保证指定的线性区域在修复后满足预期的性质, 但是, 补丁函数不能保证其他线性区域不受影响, 因此作者又提出了支撑网络来保证补丁函数几乎只在修复区域 A 发挥作用. 区域 A 的支撑网络定义为 $g_A(x,\gamma)=\sum_{i\in I}\sigma(g(b_i-a_ix,\gamma)-|I|+1)$, 其中 $g(x,y)=\sigma(\gamma x+1)-\sigma(rx)$, $\sigma(x)=\max(0,x)$, γ 是一个超参数, 表示修复的线性区域的限制程度 (即, 它对周围线性区域的影响程度). 可以看到支撑网络在区域 A 内成立, 离开区域 A 后很快就会变为 0. 综合补丁函数和支撑网络, 最终对神经网络的修复为 $f(x)+\sigma(p_A(x)+g_A(x,\gamma)-1)-\sigma(-p_A(x)+g_A(x,\gamma)-1)$, 该修复方案包含了正反两个方向的激活功能, 以确保能够在正确的方向上修改神经网络.

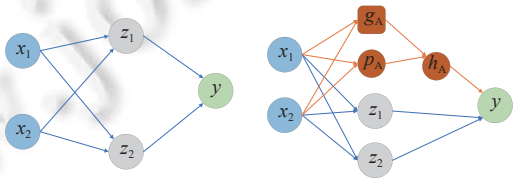

图 5 REASSURANCE 修复策略图示

3.2.7 RIPPLE (repair with linear programming concerning positive samples)[67]

基于线性规划, Sun 等人[67]提出了一种神经网络分类器的修复策略, 通过修改深度神经网络的参数来提高分类器的准确率. 首先, RIPPLE 通过采用线性规划中的约束条件和目标函数将深度神经网络修复问题编码为线性规划模型. 其次, 为了减小线性规划问题的求解复杂度, RIPPLE 考虑通过只修改神经网络输出层的参数来高效地修复神经网络. 第三, 为了在不牺牲之前正确预测样本的精度的前提下提高之前错误预测样本的精度, RIPPLE 在优化目标中采用了正确分类的样本和错误分类的样本, 来分别保证原有神经网络的性能保持和违背性质的改善.

3.2.8 ARNN (automated repair of neural networks)[68]

ARNN 的目标是修复神经网络以保证网络安全性, 它利用形式化方法直接搜索模型参数, 而不是在验证和重训练方法之间迭代直到模型是安全的. ARNN 提出了一个框架来自动和可证明地修复预训练的不安全的神经网络模型, 同时保持修复的函数表示与原始神经网络函数表示尽可能相似. 首先, ARNN 使用传统算法在指定的任务上训练模型, 然后利用可满足性模理论[101]来形式化神经网络安全性修复问题, 并搜索模型满足所需安全性的正确参数. 与提供测试错误率界限的对抗性训练相反, ARNN 框架产生了一个精确的安全的网络表示, 并且在某种意义上是通用的, 它可以执行任何一阶逻辑规范的可确定片段[102]. 此外, 除了展示如何利用现成的 SMT 求解器来自动修复不安全的神经网络, Cohen 等人还提出了一些启发式方法来保持与原始网络的相似性, 并对这些启发式方法进行了比较.

3.2.9 APRNN (architecture-preserving provable repair of DNNs)[74]

虽然 PRDNN 可以提供理论可证明的修复, 但是如何将得到的解耦深度神经网络转换得到标准和等效的深度

神经网络依然是一个开放的问题, 为此 Tao 等人[74]在后续工作 APRNN 中提出了一种保持神经网络结构的多面体可证明修复策略. 保持结构的神经网络修复只修改深度神经网络的参数, 而不修改网络结构. APRNN 首先扩展了原来的如果将修复限制为只修改单层参数可以避免优化过程中出现二次项的观察结果[59], 即如果将修复限制为仅修改某些单层 N^{l_k} 中的权重, 但允许修改所有 $N^{l_j}, j \geqslant k$ 中的偏置项, 则可以避免二次项. 虽然这个想法简单, 但允许修复更多的参数可以提高实际修复的质量. APRNN 的第 1 个关键见解是找到一个隐含违背性质 $\mathcal{P}_i$ 的线性公式 $\mathcal{P}_{\text{lin}}$, 即任何满足 $\mathcal{P}_{\text{lin}}$ 赋值的都满足规约性质 $\mathcal{P}_i$. 第 2 个关键见解是如果神经网络对于多面体的值是局部线性的, 那么修复神经网络在多面体 P 的顶点上的行为, 就足以保证其满足多面体 P 内的无穷点集, 相当于将区域修复问题规约到点修复问题, 极大降低了求解问题的复杂性. 这也引出了 APRNN 的另一个贡献, 即可以修改神经网络的参数使得原有神经网络 N 对于给定的多面体来说是局部线性的, 然后可以进一步修改参数来修复多面体顶点上的行为以实现区域修复. APRNN 修复策略具有修改神经网络多层参数的灵活性, 并且在多项式时间内运行. 它支持具有一些线性部分的激活函数的神经网络, 以及全连接、卷积、池化和残差层等网络层.

3.2.10 无错误定位的微调修复策略的对比分析

表 5 从修复粒度、核心思想、修复指标和特点等 4 个方面对无错误定位的微调修复策略进行了对比和分析, 相较于重训练修复策略对整个神经网络参数的改变, 可以看到绝大部分无错误定位的修复策略都只改变了单层网络权重; 相较于中间层的权重修改, 神经网络输出层的权重修改可以极大降低求解复杂度; 大部分修复策略都将违背的网络性质编码成约束或者优化问题, 从而使得优化问题的可解性耦合了修复的可证明性, 或者利用神经网络验证器得到网络关于性质的验证结论.

表 5 无错误定位的神经网络微调修复策略对比

方法	修复粒度	核心思想	修复指标	特点
MVDNN	网络层级	将神经网络修复问题转化为验证问题, 借助于验证器来进行求解	验证问题可满足	输出层修复复杂度低
3M-DNN	网络层级	将多层神经网络修复问题分解为一系列更容易解决的单层修复问题	最低修复成本	—
PRDNN	网络层级	在神经网络等价的解耦深度神经网络结构进行单层修复	约束存在可行解	区域修复规约为样本修复
LRNN	网络层级	神经网络修复问题定义为一个混合整数二次规划	混合整数二次规划存在可行解	输出层修复更易求解
SR	网络层级	初始网络中引入自修复层	约束满足的可行解	—
REASSURANCE	网络级	补丁函数和支撑网络	线性规划存在可行解	函数空间的修复
RIPPLE	网络层级	线性规划编码修复问题	线性规划存在可行解	—
ARNN	网络层级	可满足性问题编码修复问题	SMT问题可满足	一阶逻辑规范的可确定片段
APRNN	网络层级	神经网络的局部线性	约束存在可行解	PRDNN后续工作

3.3 包含错误定位的微调修复

错误定位策略的必要性在于神经网络中的参数数量通常是巨大的, 甚至相对较小的神经网络也可能由数千个权重和偏置参数组成, 试图优化所有的神经权重代价高昂, 并且难以扩展. 此外, 任意指定的修复层也缺乏目的性和针对性, 因此一些神经网络修复策略考虑了错误定位, 利用错误定位来确定对错误行为有更大影响的参数子集, 使微调修复过程更有倾向性和目的性.

3.3.1 DeepFault[103]

DeepFault 是第 1 个错误定位的深度神经网络白盒测试方法. DeepFault 的目标有两个: (1) 识别可疑神经元, 即可能对不正确的神经网络行为负责的神经元; (2) 合成新的输入来激活已识别的可疑神经元. 与传统软件工程的错误定位相比, DeepFault 分析神经网络的神经元在训练后的行为, 以建立其命中谱 (hit spectrum), 并通过使用可

疑度量来识别可疑神经元 (可能在深度神经网络性能 (低精度/高损失) 方面做出重大贡献的可疑神经元), 该度量使用频谱相关信息计算每个神经元的怀疑度分数. DeepFault 实例化了 3 种可疑指标: Tarantula[104]、Ochiai[105]和 D*[106]. 怀疑度分数越高的神经元更有可能被训练得不满意, 对错误的神经网络决策贡献更多, 因此这些神经元的权重需要进一步校准. DeepFault 采用怀疑引导算法合成新的输入, 通过修改正确分类的输入来获得可疑神经元的高激活值, 涉及到对抗样本生成等领域, 不在本文讨论范围. 遗憾的是, DeepFault 工作并没有进一步用来修复神经网络, 但 Eniser 等人[103]的初步探索却是神经网络错误定位的重要工作.

3.3.2 NREPAIR[62]

Dong 等人[62]提出的 NREPAIR 修复策略的总体思路通过划分输入区域, 识别一个最小的违背规约性质的输入区域集合, 然后调整那些最应该为错误行为负责的神经元, 以便满足原始神经网络违背的规约性质. 一旦发现神经网络违反了某些约束性质, NREPAIR 就会尝试为每个特定的违反性质的输入区域构建一个新的神经网络. 因此, NREPAIR 的输出是一个原始网络和多个新构建的修复网络, 实际使用中, 它将根据输入落在哪个分区来决定采用哪个网络进行预测.

NREPAIR 策略的修复框架如图 6 所示, 首先基于规约性质 $\mathcal{P}_i$ 验证原始网络 N, 如果 N 满足规约性质 $\mathcal{P}_i$, 则直接返回原始神经网络. 否则, 它将检查输入区域是否可以进一步划分, 如果有进一步的划分可用, NREPAIR 将把输入区域划分为两个互不重叠的区域, 并使用这两个新的输入区域重新开始验证过程, 直到输入区域不能被分割, 验证器生成一个反例 (counterexample) 提供给修复过程. 在修复过程中, 给定被违反的性质和反例样本, NREPAIR 定义了一个损失函数, 然后通过修改神经元的输出来最小化反例的损失值. 接着, 根据所提供的反例和损失函数的值计算每个神经元的梯度并按梯度降序对每个神经元节点进行排序, 找到梯度最大且修改次数不超过允许次数的神经元. 确定了这些可疑神经元之后, NREPAIR 将其输出修改为其原始输出减去该神经元的梯度和预定步长的乘积. 最后, 再次通过验证器运行新的神经网络来检验修复是否完成, 如果网络已经修复, 则返回新网络; 否则, 重复该过程直到找到一个解决方案或者超时.

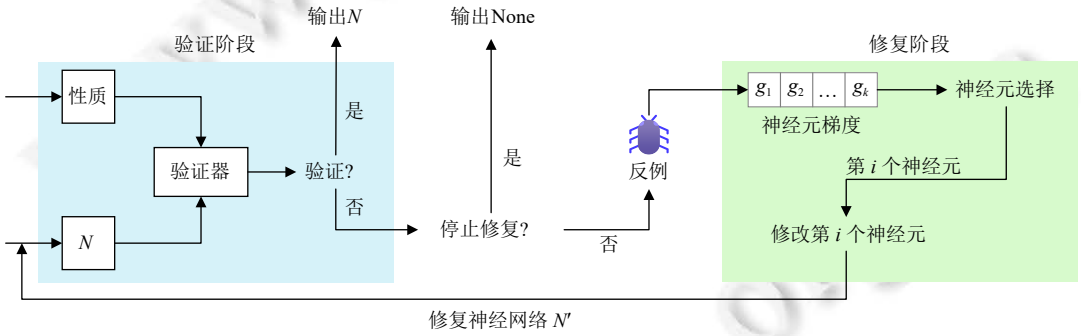

图 6 NREPAIR 神经网络修复框架图示

3.3.3 GenMuNN[76]

许多传统的程序修复方法是基于突变的, 如 GenProg[107]在修复传统软件方面取得了很好的效果. 它通过模拟生物体的交叉突变产生新的程序突变体[108], 并选择适合度高的个体在下一轮突变, 直到所有测试用例都通过. 考虑到 GenProg 等基于突变的修复方法在传统程序修复中的成功应用, Wu 等人[76]提出了一种基于突变的神经网络模型修复方法 GenMuNN. 第 1 步是错误定位. 为了避免可能引入的错误行为, 他们采用了一种更加细粒度的错误定位方法, 即对神经元的缺陷权重进行定位, 得到一个影响神经网络模型精度的可疑权重递减序列. 为了得到可疑权重的降序, GenMuNN 将测试数据输入到神经网络模型中, 并从输入层传播到输出层, 对于每个神经元, 将测试数据的输出按比例分配给每个权重作为覆盖信息. 然后根据覆盖信息计算权重的怀疑度, 并将权重排序, 可疑值越高的权重越有可能存在缺陷. 第 2 步是第 1 步产生的权重序列中的权重依次发生突变. 首先, 对于 W 中的一个权重 w_i, $\bar{w}$ 为可疑权重所在神经元所有权重的平均权重 (不包括 w_i). 然后, 生成一组突变体 $\{M_{i,1}, M_{i,2}, \ldots\}$ 其中 $M_{i,j}$

是将 w_i 更新为 $w_i+r_{i,j}$ 生成的, $r_{i,j}$ 是 $[-a\times\bar{w},a\times\bar{w}]$ 中的随机值, a 是可调参数. 假设原模型的精度为 p, 突变体的精度为 $p_{i,1},p_{i,2},\ldots$, 如果 $p_{i,j}-p>\varepsilon$, 突变体 $M_{i,j}$ 将作为合格突变体保留, 否则, 突变体 $M_{i,j}$ 将作为不合格突变体被消除. 为了加速修复过程, 每轮突变只保留 n 个合格的突变体. 对于合格的突变体, 准确率越高, 分配到下一轮突变的概率就越大. 最后, 如果连续 s 轮没有产生符合条件的突变体, 则停止当前权重的突变, 选择 W 中权重 w_{i+1} 进行下一轮突变. 当连续 t 个权重没有产生合格的突变体时, 修复过程将停止, 整个修复过程如图 7 所示.

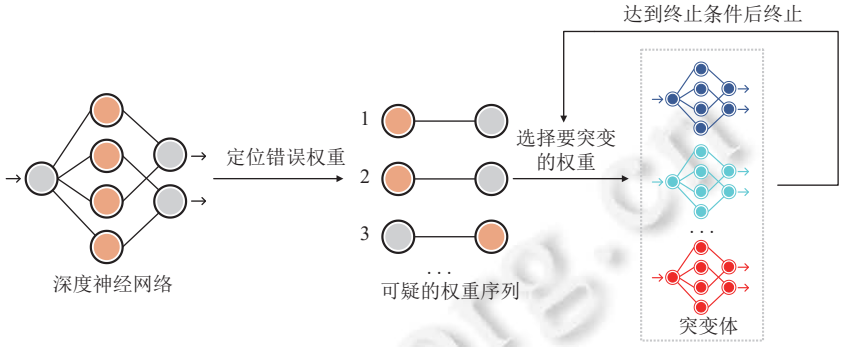

图 7 GenMuNN 修复框架图示

3.3.4 NNrepair (neural network repair)[65]

直观地说, 神经网络输出层的修复更容易, 因为输出层的修复可以直接影响决策, 并且可以使用网络的预期输出作为约束来指导修复. 然而, 这种修复方法的泛化性较差, 而且网络可能在某些中间隐藏层出现故障. 中间层的修复可以对网络的行为产生更大的影响, 但由于不清楚使用何种约束或者启发式策略来指导修复, 因此更困难. NNrepair 修复策略则提出了可以同时支持中间层和输出层修复的技术. 在这项工作中, Usman 等人[65]提出使用神经元激活模式 (activation pattern)[109]作为中间层修复的指导. 由于同时修复所有输出类非常困难, 因此 NNrepair 转而求解一组专家网络, 每个目标类一个, 将这些专家网络组合起来得到最终修复的分类器. 具体来说, NNrepair 修复策略包含以下几个步骤: (1) 错误定位: 这一步的目标是识别一组可疑神经元和输入边的可疑权重. (2) 符号化执行 (concolic execution)[110]: 在可疑边的权重上添加 δ 值, 然后使用该网络沿正、负样本并行执行, 收集可疑神经元的符号表达式值. (3) 约束求解: 由一组修复约束组合形成的符号表达式, 然后用已有的求解器进行求解. 本质上, 修复性质需要对正样本的网络决策进行编码, 并对负样本的网络决策进行修改. 对于最后一层修复, 这相当于添加决策层的约束. 对于中间层修复, 使用激活模式的约束将修复保持在网络层的局部. 从求解器中得到的符号 δ 的解用于更新网络的权重, 从而获得特定类的专家网络. (4) 专家组合: 将每个类得到的专家网络组合得到修复后的分类器. 这一步需要注意避免专家网络之间的冗余计算, 并且不损害分类网络的整体准确性.

3.3.5 Arachne[71]

Arachne 是一个典型的基于搜索的包含错误定位的神经网络微调修复框架, 其修复策略主要包含两个环节: 错误定位和补丁生成. 在错误定位阶段, 针对一组给定的导致错误行为的输入样本, Arachne 将识别一组可能与观察到的错误行为相关联的权重. 修改这些权重的值可能会影响模型的行为, 以期在正确的方向上修复神经网络. 定位策略根据损失函数和错误输入对权重梯度进行排序, 定位的目的在于识别对网络中错误行为负责的权重. 除了梯度损失之外, Arachne 的错误定位过程还考虑每个权重对最终输出的影响程度, 定义为给定权重和前一层相应神经元输出的乘积. 而后, 综合权重梯度和影响程度, Arachne 选择前 k 个权重作为候选修复权重, 错误定位过程如图 8 所示. 在补丁生成阶段, Arachne 则采用了差分演化算法 (differential evolution, DE)[111]来生成候选权重更新值. 在差分演化优化算法中, Arachne 使用适应度函数 (fitness function) 来评估每个候选解并指导搜索过程. 与其他 G&V 技术[112,113]一样, Arachne 的适应度函数主要由两个部分组成: 修复错误分类和保留正确分类, 并且采用超参数在修复过程中平衡这两个部分.

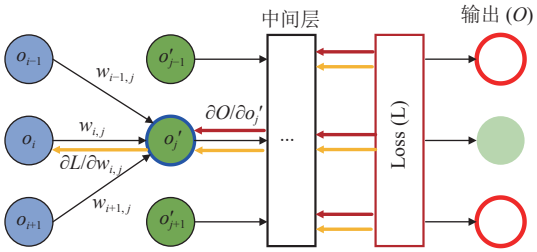

图 8 Arachne 修复框架错误定位策略图示

3.3.6 CARE (causality-based repair)[75]

近年来, 因果关系推断在可解释人工智能 (explainable artificial intelligence, XAI) 中得到了越来越多的关注, 研究者设计了多种基于因果关系的方法来解释机器学习模型中神经网络组件的决策过程[114,115]. 与传统方法相比, 因果方法确定了网络模型组成部分的原因和结果, 从而促进了对网络模型决策的推理和分析. CARE 框架的目标是通过最小限度地调整给定神经网络的参数来构建满足性质的神经网络. 具体来说, Sun 等人[75]提出了一种基于因果关系的神经网络修复技术, 主要包括两个步骤: (1) 执行基于因果关系的错误定位和 (2) 优化识别神经元的参数以减少性质违背情况. 基于因果的定位策略识别了一组较少的需要对错误行为承担责任的神经元进行修复. 基于因果分析的错误定位策略不同于基于梯度的方法, 基于梯度的方法根据统计相关性得出结论, 可解释性较差. 识别到错误神经元之后, CARE 采用粒子群优化算法 (particle swarm optimization, PSO)[116]进行神经元参数更新. 粒子群算法模拟动物的智能集体行为, 如鱼群和鸟群, 是一种广泛使用的连续空间优化算法. PSO 算法为在定位环节中识别的神经元的权重参数搜索小范围的调整, 以满足指定的性质且尽可能保持原有网络的性能, 完成网络修复.

3.3.7 NeuRecover[77]

采用重训练来修复神经网络的错误行为通常会对其他行为产生影响, 导致回归问题的产生, 即修复后的神经网络不能正确处理在原始神经网络上正确处理的输入. 为了解决这个问题, NeuRecover 通过在错误定位步骤中使用训练历史来抑制回归问题的产生. NeuRecover 的基本思想是, 通过对比历史模型和当前模型, 找到某一数据样本在训练历史中可以被模型正确分类而现在却被错误分类的检查点 (checkpoint), 然后确定可以安全地纠正错误分类的权重. 具体来说, NeuRecover 识别具有以下属性的权重: 它们的值在训练过程中发生了显著变化, 并且它们不影响改进数据的输出 (即, 最初被错误分类但随后在训练过程中被正确分类的数据), 但影响回归数据的输出 (即, 曾经被正确分类但随后在训练过程中被错误分类的数据). 然后, 通过对识别出的权重进行粒子群优化, NeuRecover 可以修复神经网络模型, 得到更多的改进数据和更少的回归数据.

3.3.8 I-repair (repair using influence)[63]

因为原始训练数据的隐私性, 一些神经网络模型的原始训练数据并不公开, 基于这种情况, Henriksen 等人[63]设计提出的 I-repair 修复策略考虑在不能访问原始训练集的情况下, 针对有限的分类样本集合修复神经网络的问题. 给定一个输入 x 和一个待修复的神经网络 N, I-repair 首先估计每个参数在多大程度上影响输出 $N(x)$, 主要思想是, 如果改变某个参数的值导致 $N(x)$ 发生剧烈变化, 则这个参数是一个“有影响”的参数. 如果一个参数对于待修复数据集 D_R 有很强的影响, 但是对于正确输入样本集合 D 却影响较弱, 那么这个参数是一个很好的候选修复参数, 修改它的值可能使得网络决策发生预期的变化. 每个参数的影响力是基于给定的损失函数对参数的梯度来进行计算的. 一旦选择了一组参数进行修复, I-repair 进行反向传播, 目的是修复 D_R 中的错误分类, 同时保持 D 中输入的输出不变; 这个过程会重复进行, 直到所有输入都被修复或超时. 在此之后, I-repair 算法报告在终止之前可以修复的输入和不能修复的输入.

3.3.9 BIRDNN (behavior-imitation based repair of DNNs)[72]

BIRDNN 修复策略与其他较高层次的研究神经网络状态空间或者推理的神经网络修复方法不同, 该修复框架更加注重研究底层的神经元的行为, 并且基于神经元行为模仿 (behavior imitation) 修复神经网络. BIRDNN 同时支

持重训练和微调的神经网络修复范式. 对于基于重训练的修复方法, BIRDNN 通过模仿附近正确样本的输出, 为错误样本分配新的正确标签, 然后重新训练原始神经网络以改善错误行为. 同时, 为了追求较小的网络修改, BIRDNN 在训练过程中也考虑保持原有数据集的输入输出关系. 对于基于微调的方法, 错误定位通过分析正确样本和错误样本在具体神经元上的行为差异 (behavior difference) 来识别对错误预测最应该承担责任的神经元, 即行为差异最大的那些神经元. 然后, 和 CARE 策略类似, BIRDNN 利用粒子群优化算法来修复这些错误神经元, 适应度函数的构造保证搜索过程在保持原始神经网络性能的同时最小化性质违背. 为了解决更有挑战性的神经网络区域修复问题, Liang 等人[72]还在 BIRDNN 集成了蒙特卡罗采样技术从正确样本和错误样本两个方面来刻画神经网络的区域行为. 区域行为刻画技巧将区域修复问题转换为基于样本的修复问题, 以概率近似正确 (probably approximate correct, PAC) 的方法修复有缺陷的神经网络.

3.3.10 包含错误定位的神经网络修复策略的对比分析

表 6 从定位部分和修复部分两个方面对包含错误定位的神经网络修复策略进行了对比和分析, 定位部分主要比较核心思想、定位粒度和定位指标, 修复部分则是比较核心思想和修复指标. 可以看到, 与无错误定位的修复策略相比 (表 5), 包含错误定位的修复策略定位粒度更为细致, 主要是神经元级和权重级定位; 各个修复框架的定位策略基本都是启发式, 具有一定的可解释性. 对于神经元或者权重的修复, 主要算法集中在空间搜索上, 比如差分演化算法、粒子群优化算法等, 因而这些修复的可证明性较差, 但是修复效率和可扩展性高.

表 6 包含错误定位的微调修复策略对比

方法	定位部分			修复部分	
	核心思想	定位粒度	定位指标	核心思想	修复指标
DeepFault	神经元在样本下的命中谱	神经元级	怀疑度度量	—	—
NREPAIR	神经元关于损失函数的梯度	神经元级	梯度	原始输出减去梯度和预定步长的乘积	验证器验证
GenMuNN	权重关于损失函数的梯度	权重级	梯度	遗传变异算法	模型精度
NNrepair	神经元的激活模式	神经元级	激活模式	符号执行、约束求解	模型精度
Arachne	神经元关于损失函数的梯度	神经元级	梯度、前向传播影响	差分演化优化算法	适应度函数
CARE	神经元与错误输出的因果关系	神经元级	因果度量	粒子群优化算法	适应度函数
NeuRecover	错误行为在模型演化历史中的检查点分析	权重级	参数变化值和输出变化值的差异性	粒子群优化算法	适应度函数
I-repair	参数改变量对样本输出的影响	权重级	参数变化值和输出变化值的差异性	反向传播更新	梯度
BIRDNN	神经元关于正负样本的行为差异	神经元级	状态差异	粒子群优化算法	适应度函数

4 评价指标和测试基准

在本节, 笔者基于对已有工作的比较和分析, 介绍了深度神经网络修复策略常见的评价指标, 并基于第 2 节中对修复问题的分类方式, 从神经网络样本修复和神经网络区域修复两个方面列举了常见的测试基准和实验设置. 但是, 神经网络修复作为一个近几年逐渐兴起的研究领域, 目前在评价指标和测试基准上还没有形成较为统一的共识和规范, 这也是未来该领域需要进一步关注和发展的研究方向.

4.1 评价指标

和本文第 2.3 节列出的神经网络修复的要求相对应, 表 7 列出了神经网络修复策略常见的评价指标及其内涵解释, 主要包括性质提升、性能退化、准确度、参数距离、预测距离和运行时间. 需要说明的是, 这里只给出了常见评价指标的概念原型, 更为形式化的定义需要根据实验设置进一步规范和约束, 在已有工作中, 因为修复策略和实验设置的不同, 这些指标的计算方式也不尽相同. 此外, 部分评价指标之间存在意义重叠, 需要研究者在实际使用中综合考虑.

表 7 常见修复策略的评价指标及意义

评价指标	意义
性质提升 (improvement)	修复后的神经网络 N' 和原始神经网络 N 相比, 被违反的修复性质满足程度如何, 比如原来不能正确处理的样本修复后可以正确处理的比例
性能退化 (drawdown)	修复后的神经网络 N' 和原始神经网络 N 相比, 和修复性质不相关的部分性能保持如何, 比如二者分类准确度的差异
准确度 (accuracy)	该指标是性质提升和性能退化的综合评价, 性质提升越大, 性能退化越小(甚至有提升), 那么修复后的神经网络的准确度越高
参数距离 (parameter distance)	评估修复后的神经网络 N' 和原始神经网络 N 的参数距离, 即 $\|\theta'-\theta\|_L$, 其中 $\|\cdot\|_L$ 是L-距离
预测距离 (prediction distance)	评估修复后的神经网络 N' 和原始神经网络 N 在输入输入样本 x 上的输出距离, 即 $\|N_{\theta'}(x)-N_{\theta}(x)\|_L$
运行时间 (running time)	反映修复策略的效率, 在包含错误定位的微调修复中, 错误定位时间也可以反映定位策略的效率

4.2 基于样本的神经网络修复基准

本节主要介绍基于 MNIST 数据集的、基于 Natural Adversarial Example 数据集的和基于 Census Income 数据集的神经网络样本修复测试基准以及相应的神经网络修复的实验设置. 作为实验案例, 其他神经网络和数据集可以考虑采用类似的实验设置.

4.2.1 基于 MNIST 数据集的神经网络样本修复测试基准

MNIST 数据集[117]由 70k 张分类标签为 $0,1,2,\ldots,9$ 的手写数字的灰度图像组成, 其中 60k 为训练数据, 10k 是测试数据. 基于 MNIST 训练数据集训练得到的神经网络 (目前实验中采用的神经网络大部分都是作者自己构建并训练的前馈全连接神经网络, 且一般是较小规模的神经网络) 通常在测试数据集上并不能取得 100% 的分类准确度, 因此可以考虑根据那些神经网络不能正确分类的样本来修复预训练的神经网络模型, 进一步提升预训练模型的准确性.

类似的基于数据集样本的神经网络修复还包括 Fashion-MNIST (涵盖来自 10 种类别的不同商品的图片, 其大小、格式与原始的 MNIST 完全一致)[118], CIFAR-10 (包含 10 种类别的不同物品的彩色图像数据集)[119], CIFAR-100 (包含 100 类物品的图片数据集, 其大小、格式与 CIFAR-10 完全一致)[119], Labelled Faces in the Wild (LFW)(人脸识别数据集, 共有 5 749 人的 3 233 张人脸图像)[120]等数据集上的神经网络模型准确性的修复, 其中基于 Fashion-MNIST 和 CIFAR-10 数据集的修复一般针对于较小规模、较为简单的神经网络, 而基于 CIFAR-100 和 LFW 数据集的修复则一般在较大规模、较复杂结构的神经网络上进行.

4.2.2 基于 Natural Adversarial Example 数据集的神经网络样本修复测试基准

自然对抗样本 (Natural Adversarial Example) 数据集提供了 7 500 个自然界中存在的可能被 SqueezeNet 神经网络错误分类的输入样本[121], SqueezeNet[122]是基于 ImageNet 数据集设计训练的卷积神经网络, 该网络一共有 18 层, 包含 727 626 个参数. 相较于基于 MNIST 数据集的小型神经网络修复, 较大规模的 SqueezeNet 网络模型的修复更加具有挑战性. 自然对抗样本数据集中存在的错误分类样本表明 SqueezeNet 神经网络的分类准确性依然存在进一步被修复的必要, 即实验中需要修复的分类样本. 实验设计上, 可以针对自然对抗样本数据集中的全部错误分类样本进行修复, 也可以从中选择某些子集进行样本修复, 实验评估指标则是修复后的 SqueezeNet 网络在分类准确性上的提升.

4.2.3 基于 Census Income 数据集的神经网络样本修复测试基准

Census Income 数据集[123]由包含 13 个特征的 32 561 个数据实例组成, 用于预测个人年收入是否超过 5 万美元. 在所有属性中, 性别、年龄和种族是隐私特征, 标签是个人年收入是否超过 5 万美元, 其中隐私特征是指预期与标签预测独立的特征. 当某个样本实例在神经网络的预测与隐私特征不独立时, 则神经网络的公平性被破坏. 基于 Census Income 数据集的神经网络公平性修复旨在提升神经网络的公平性. 类似的基于数据集样本的神经网络公平性修复还包括基于 German Credit 数据集 (1 000 个实例, 20 个特征, 用于评估个人信用. 年龄和性别是隐私特征, 标签是个人信用是否良好)[124]和 Bank Marketing 数据集 (45 211 个实例, 17 个特征. 年龄是隐私特性, 标签是客

户是否会认购定期存款)[125]的神经网络修复.

4.3 基于区域的神经网络修复基准

本节主要介绍基于 MNIST 数据集的神经网络区域修复、ACAS Xu 神经网络区域修复和 HorizontalCAS 神经网络区域修复测试基准及相应的神经网络修复的实验设置. 作为实验案例, 其他神经网络和数据集可以考虑采用类似的实验设置.

4.3.1 基于 MNIST 数据集的神经网络区域修复测试基准

文献 [126] 考虑了在神经网络验证工作中广泛使用的一个神经网络, 该神经网络是基于 MNIST 手写数字数据集的以 ReLU 作为激活函数, 由 3 个网络层和 88 010 个参数组成的神经网络, 在 MNIST 测试集上的准确率为 96.5%. 考虑构建如下一条线段, 该线段的一个端点是未损坏的 MNIST 手写数字图像, 另一个端点是来自 MNIST-C 数据集, 即被雾化损坏的 MNIST 图像[127]. 考虑该神经网络的一个鲁棒性性质规约, 即如果 I 是未损坏的 MNIST 图像, I' 是一个相应的雾化损坏图像, 那么从 I 到 I' 这条线上的所有 (无限) 的输入样本必须具有与 I 相同的分类. 对于此修复问题, 从 I 到 I' 的线段是该神经网络的修复区域, 修复目标则在于使得修复后的神经网络满足鲁棒性规约.

4.3.2 ACAS Xu 神经网络区域修复测试基准

ACAS Xu 神经网络[128]的安全性修复是一个被广泛使用和评估的神经网络区域修复测试基准. ACAS Xu 神经网络包含 45 个神经网络阵列 (组织为 5×9 阵列, 即 $N_{1,1},N_{1,2},\ldots,N_{1,9},\ldots,N_{5,9}$), 用来生成机载避撞系统 X 的无人版本的机动建议, 机载避撞系统 X 是由美国联邦航空管理局设计开发的高度安全攸关的系统. 更具体地说, 这些深度神经网络采用 5 维输入, 代表航天器周围的场景 (航天器的速度, 与入侵器的距离等), 并输出 5 维的预测结果, 指示 5 种可能的操作建议 (clear-of-conflict, weak left, weak right, strong left 或者 strong right). ACAS Xu 神经网络以每个离散化的状态作为输入, 网络使用全连接神经网络, 采用 ReLU 作为激活函数, 有 6 个隐藏层. 为了更好地描述 ACAS Xu 神经网络的实际运行需求, 研究者提出了 10 个与之相关联的安全性质 ($\mathcal{P}_1,\mathcal{P}_2,\ldots,\mathcal{P}_{10}$). 这些安全性规约要求与特定输入区域内的输入相对应的输出必须落在给定的安全的多面体区域中. 现有的一些神经网络验证工具[27]已经证实这 45 个神经网络模型中, 有 35 个神经网络违反了至少一个安全性质, 例如神经网络 $N_{2,9}$ 违反了安全性质 $\mathcal{P}_8$. ACAS Xu 神经网络修复是目前广泛使用的神经网络修复的比较基准. 但是, 需要注意的是, ACAS Xu 神经网络的原始训练数据集并不公开, 这也为网络修复带来了一定的困难, 目前主要是通过采样神经网络的输入输出序偶来近似模拟原始训练数据.

4.3.3 HorizontalCAS 神经网络区域修复测试基准

HorizontalCAS (HCAS) 神经网络[129]是 ACAS Xu 神经网络的简化版本, HCAS 神经网络是 Julian 等人提出的神经网络阵列, 共包含 40 个神经网络模型, 每个模型包含 3 个节点的输入层, 5 个隐藏层, 每个隐藏层有 25 个节点, 最后有 5 个节点的输出层. 与 ACAS Xu 同时进行垂直速度和水平转弯速度建议不同, HCAS 只发布航天器的水平转弯速度建议. 可以采用 ACAS Xu 神经网络的安全性质进行验证和修复, 例如基于安全性质 $\mathcal{P}_5$ 修复神经网络 $N_{1,4}$ [71]. 此外, VerticalCAS (VCAS) 神经网络阵列[129]发布航天器的垂直速度建议, 是 ACAS Xu 神经网络的另一简化版本, 也可以考虑类似的实验设计进行神经网络修复.

5 未来研究方向

本文主要从重训练、无错误定位的微调修复和包含错误定位的微调修复 3 个方面介绍了深度神经网络修复策略的最新研究进展, 并比较和分析了不同修复策略之间的异同与联系. 通过分析比较, 目前学术界对于神经网络修复问题的研究已经逐渐形成体系, 在不同种类的神经网络和不同类型的神经网络性质的修复上取得了优异的表现. 但是随着深度学习与人类社会的不断融合, 对神经网络修复带来挑战的同时也将提出更高的要求, 神经网络修复策略领域仍需进一步发展与完善. 展望神经网络修复策略领域的后续发展, 笔者总结得到以下几个未来可行的或者需要关注的研究方向.

5.1 从启发式经验修复到理论可证明修复

虽然目前已有的神经网络修复策略在多个测试基准上取得了出色的表现, 但是大部分的修复工作依然是经验性修复, 即是面向神经网络模型的启发式修复, 缺乏足够的理论支撑和修复保证. 目前只有少数的修复方法是理论可证明的修复, 而这些方法对修复的神经网络种类有较高的要求, 同时可扩展性也存在较大局限, 因而实际应用中受到较大的限制. 因此未来需要进一步发展理论可证明的神经网络修复策略, 扩展理论可证明的修复方法的应用范围; 也可以考虑采用神经网络验证或者测试工具作为后端为神经网络修复策略提供性能佐证.

5.2 权重级细粒度定位和修复

目前大部分的修复策略是网络级、网络层级或者神经元级的神经网络权重修复. 以神经元级的权重定位和修复为例, 定位到一个神经元通常意味着可能更改所有与之相关联的权重, 更不要说网络级和网络层级的权重修复可能更改整个原有网络或者某一层的所有权重. 从这一点来看, 这 3 种粒度的定位和修复依然是比较粗粒度的, 可以考虑进一步将错误定位和修复细化到权重级, 这是比较具有挑战性的工作, 因为即便是小规模的神经网络也包含大量的网络参数, 而且权重的定位又往往和神经元的状态密切相关. 权重级的定位和修复应当在之后的工作中予以重点关注和发展, 可以进一步满足神经网络修复的最小性和局部性, 也能够更大程度地避免引入新的不必要的错误行为.

5.3 可能引入的错误行为的评估

在神经网络模型的修复中, 极小的修改都可能会引入不必要的错误行为已是学界的共识, 但是目前的研究进展中对这些可能新引入的错误行为的识别和评估还没有形成规范. 大部分的修复策略都是通过追求修复的最小性和局部性来最小化可能引入的错误行为. 另外, 大部分修复策略的核心思想是待修复权重在样本集合上的拟合近似, 过拟合风险的存在会进一步加剧可能引入错误行为的情况. 这也是未来研究中需要进一步推动发展的方向, 在理论可证明的修复基础上明确神经网络修复策略所产生的副作用.

5.4 集成化的修复策略的测试基准平台

相较于神经网络验证成熟的、集成化的比较和测试平台 ERAN[126]、Marabou[30]、GPUPoly[26]、PRODeep[130]等, 神经网络修复策略方面的工作依然是欠缺和有待发展的. 一个标准化的测试和比较平台将会更加有助于该领域的快速发展. 现在已有的神经网络修复策略的集成化测试平台 AIREPAIR[131]和 NeVer 2.0[132]只实现了少数的神经网络修复框架. 此外, 在实现更为集成化的修复策略测试基准平台的同时, 研究者需要对神经网络修复策略领域的相关问题进一步明确和规范.

5.5 神经网络智能系统的修复

相较于深度神经网络的修复, 神经网络智能系统的修复 (以神经网络作为主要决策部件的系统) 显得更为重要, 也更加具有实际应用前景, 比如自动驾驶系统等. 与基于输入样本或者区域的神经网络修复不同, 神经网络智能系统的实际运行环境对其修复设置了更多的困难和挑战, 比如气候、天气、温度、人为因素甚至一些突发因素等. Yu 等人[133]提出的 DeepRepair 和 Majd 等人[134]的工作做了这个方面的一些探索, 但是目前修复的神经网络系统的规模依然较小, 修复算法的可扩展性需要进一步提升. 很多神经网络智能系统的修复需求是实时的, 这也意味着更加高效的修复算法. 此外, 理想情况下神经网络习得的信息对于实际神经网络智能系统的预测可能产生过拟合数据, 如何找到一个切合实际场景的子集来修复训练集, 使得神经网络可以忘记那些不相关或不重要的历史信息, 从而提高模型的泛化能力和预测准确率也是需要进一步研究的课题[135].

6 结 语

深度神经网络修复的相关研究正处于刚刚兴起并快速发展的阶段, 随着神经网络智能系统在社会生活中日益广泛的应用和部署, 神经网络模型的修复也将更加紧迫和重要. 虽然目前神经网络的修复已经逐渐形成体系, 但是已有的修复框架大多为面向模型的启发式修复, 少数的可证明修复策略停留在一般 (中等) 规模的神经网络和较为简单的网络性质, 求解规模上依然受限. 同时, 对于神经网络修复过程可能引入的新的错误行为的识别和评价目前没有形成规范和共识, 集成化的神经网络修复的基准测试平台也是匮乏的. 本文调研比较了近几年神经网络修

复的相关文章, 梳理了神经网络修复的国内外研究现状, 比较了不同神经网络修复策略的内在联系和区别. 最后, 文章讨论了神经网络修复目前常见的评价指标和测试基准, 展望了神经网络修复领域未来可能的研究方向, 期望进一步推动神经网络修复领域的研究和发展.

References:

[1] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma SA, Huang ZH, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. ImageNet large scale visual recognition challenge. Int'l Journal of Computer Vision, 2015, 115(3): 211–252. [doi: 10.1007/s11263-015-0816-y]

[2] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6): 84–90. [doi: 10.1145/3065386]

[3] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L. Large-scale video classification with convolutional neural networks. In: Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition. Columbus: IEEE Computer Society, 2014. 1725–1732. [doi: 10.1109/CVPR.2014.223]

[4] Pennington J, Socher R, Manning CD. GloVe: Global vectors for word representation. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing. Doha: Association for Computational Linguistics, 2014. 1532–1543. [doi: 10.3115/v1/D14-1162]

[5] Andor D, Alberti C, Weiss D, Severyn A, Presta A, Ganchev K, Petrov S, Collins M. Globally normalized transition-based neural networks. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin: Association for Computational Linguistics, 2016. 2442–2452. [doi: 10.18653/v1/P16-1231]

[6] Hinton G, Deng L, Yu D, Dahl GE, Mohamed AR, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 2012, 29(6): 82–97. [doi: 10.1109/MSP.2012.2205597]

[7] Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang JK, Zhang X, Zhao J, Zieba K. End to end learning for self-driving cars. arXiv:1604.07316, 2016.

[8] Scheiner N, Appenrodt N, Dickmann J, Sick B. Radar-based road user classification and novelty detection with recurrent neural network ensembles. In: Proc. of the 2019 IEEE Intelligent Vehicles Symp. (IV). Paris: IEEE, 2019. 722–729. [doi: 10.1109/IVS.2019.8813773]

[9] Amato F, López A, Peña-Méndez EM, Vaňhara P, Hampl A, Havel J. Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine, 2013, 11(2): 47–58. [doi: 10.2478/v10136-012-0031-x]

[10] Lipton ZC, Kale DC, Elkan C, Wetzel RC. Learning to diagnose with LSTM recurrent neural networks. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan, 2016.

[11] Wang QL, Guo WB, Zhang KX, Ororbia AG, Xing XY, Liu X, Giles CL. Adversary resistant deep neural networks with an application to malware detection. In: Proc. of the 23rd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Halifax: ACM, 2017. 1145–1153. [doi: 10.1145/3097983.3098158]

[12] Yuan ZL, Lu YQ, Wang ZG, Xue YB. Droid-Sec: Deep learning in Android malware detection. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 371–372. [doi: 10.1145/2740070.2631434]

[13] Teubner T, Flath CM, Weinhardt C, van der Aalst W, Hinz O. Welcome to the Era of ChatGPT et al. The prospects of large language models. Business & Information Systems Engineering, 2023, 65(2): 95–101. [doi: 10.1007/s12599-023-00795-x]

[14] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego, 2015.

[15] Xu K, Xie M, Tang LC, Ho SL. Application of neural networks in forecasting engine systems reliability. Applied Soft Computing, 2003, 2(4): 255–268. [doi: 10.1016/S1568-4946(02)00059-5]

[16] Berwick RC. The failure of deep neural networks to capture human language's cognitive core. In: Proc. of the 20th IEEE Int'l Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC). Banff: IEEE, 2021. 3. [doi: 10.1109/ICCICC53683.2021.9811297]

[17] Hern A. Facebook translates 'good morning' into 'attack them', leading to arrest. 2017. https://www.theguardian.com/technology/2017/oct/24/facebook-palestine-israel-translates-good-morning-attack-them-arrest

[18] Hill K. Wrongfully accused by an algorithm. New York Times, 2020. https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html

[19] Lee D. US opens investigation into Tesla after fatal crash. BBC, 2016. https://www.bbc.co.uk/news/technology-36680043

[20] Seshia SA, Sadigh D, Sastry SS. Toward verified artificial intelligence. Communications of the ACM, 2022, 65(7): 46–55. [doi: 10.1145/3503914]

[21] Liu Y, Yang PF, Zhang LJ, Wu ZL, Feng Y. Survey on robustness verification of feedforward neural networks and recurrent neural networks. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3134–3166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6863.htm [doi: 10.13328/j.cnki.jos.006863]

[22] 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]

[23] Mirman M, Gehr T, Vechev M. Differentiable abstract interpretation for provably robust neural networks. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 3578–3586.

[24] 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.

[25] 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]

[26] Müller C, Serre F, Singh G, Püschel M, Vechev M. Scaling polyhedral neural network verification on GPUs. In: Proc. of the 4th MLSys Conf. San Jose, 2021. 733–746.

[27] 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]

[28] 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]

[29] Gopinath D, Katz G, Pasareanu CS, Barrett C. DeepSafe: A data-driven approach for checking adversarial robustness in neural networks. arXiv:1710.00486, 2020.

[30] 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]

[31] Müller MN, Makarchuk G, Singh G, Püschel M, Vechev M. PRIMA: General and precise neural network certification via scalable convex hull approximations. Proc. of the ACM on Programming Languages, 2022, 6(POPL): 43. [doi: 10.1145/3498704]

[32] Dvijotham K, Gowal S, Stanforth R, Arandjelovic R, O'Donoghue B, Uesato J, Kohli P. Training verified learners with learned verifiers. arXiv:1805.10265, 2018.

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

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

[35] Liu WW, Song F, Zhang THR, Wang J. Verifying ReLU neural networks from a model checking perspective. Journal of Computer Science and Technology, 2020, 35(6): 1365–1381. [doi: 10.1007/s11390-020-0546-7]

[36] Zhang YD, Zhao Z, Chen GK, Song F, Zhang M, Chen TL, Sun J. QVIP: An ILP-based formal verification approach for quantized neural networks. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Rochester: ACM, 2022. 82. [doi: 10.1145/3551349.3556916]

[37] Zhao Z, Zhang YD, Chen GK, Song F, Chen TL, Liu JX. CLEVEREST: Accelerating CEGAR-based neural network verification via adversarial attacks. In: Proc. of the 29th Int'l Symp. on Static Analysis (SAS). Auckland: Springer, 2022. 449–473. [doi: 10.1007/978-3-031-22308-2_20]

[38] 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]

[39] Huang XW, Kroening D, Ruan WJ, Sharp J, Sun YC, Thamo E, Wu M, Yi XP. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review, 2020, 37: 100270. [doi: 10.1016/j.cosrev.2020.100270]

[40] Tambon F, Khomh F, Antoniol G. A probabilistic framework for mutation testing in deep neural networks. Information and Software Technology, 2023, 155: 107129. [doi: 10.1016/i.infsof.2022.107129]

[41] Ige T, Marfo W, Tonkinson J, Adewale S, Matti BH. Adversarial sampling for fairness testing in deep neural network. Int'l Journal of Advanced Computer Science and Applications, 2023, 14(2): 7–13. [doi: 10.14569/IJACSA.2023.0140202]

[42] Han G, Li Z, Tang P, Hu CY, Guo SQ. FuzzGAN: A generation-based fuzzing framework for testing deep neural networks. In: Proc. of the 24th IEEE Int'l Conf. on High Performance Computing & Communications; the 8th Int'l Conf. on Data Science & Systems; the 20th

Int'l Conf. on Smart City; the 8th Int'l Conf. on Dependability in Sensor, Cloud & Big Data Systems & Application. Hainan: IEEE, 2022. 1601–1608. [doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00244]

[43] Xie XF, Ma L, Juefei-Xu F, Xue MH, Chen HX, Liu Y, Zhao JJ, Li B, Yin JX, See S. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. In: Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis (ISSTA). Beijing: ACM, 2019. 146–157. [doi: 10.1145/3293882.3330579]

[44] Zhang XY, Xie XF, Ma L, Du XN, Hu Q, Liu Y, Zhao JJ, Sun M. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 739–751. [doi: 10.1145/3377811.3380368]

[45] Chen ZP, Cao YB, Liu YQ, Wang HY, Xie T, Liu XZ. Understanding challenges in deploying deep learning based software: An empirical study. arXiv:2005.00760, 2020.

[46] Gowal S, Dvijotham K, Stanforth R, Bunel R, Qin CL, Uesato J, Arandjelovic R, Mann T, Kohli P. On the effectiveness of interval bound propagation for training verifiably robust models. arXiv:1810.12715, 2018.

[47] Zhang H, Chen HG, Xiao CW, Gowal S, Stanforth R, Li B, Boning D, Hsieh CJ. Towards stable and efficient training of verifiably robust neural networks. In: Proc. of the 8th Int'l Conf. on Learning Representations. Addis Ababa: OpenReview.net, 2020.

[48] Pauli P, Koch A, Berberich J, Kohler P, Allgöwer F. Training robust neural networks using Lipschitz bounds. IEEE Control Systems Letters, 2022, 6: 121–126. [doi: 10.1109/LCSYS.2021.3050444]

[49] Liang Z, Wu TR, Liu WW, Xue B, Yang WJ, Wang J, Pang ZB. Towards robust neural networks via a global and monotonically decreasing robustness training strategy. Frontiers of Information Technology & Electronic Engineering, 2023, 24(10): 1375–1389. [doi: 10.1631/FITEE.2300059]

[50] Liang Z, Liu WW, Wu TR, Ren DJ, Xue B. Advances and prospects of training methods for robust neural networks. Science and Technology Foresight, 2023, 2(1): 78–89 (in Chinese with English abstract). [doi: 10.3981/j.issn.2097-0781.2023.01.006]

[51] Zhao HJ, Zeng X, Chen TL, Liu ZM, Woodcock J. Learning safe neural network controllers with barrier certificates. Formal Aspects of Computing, 2021, 33(3): 437–455. [doi: 10.1007/s00165-021-00544-5]

[52] Liang Z, Ren DJ, Liu WW, Wang J, Yang WJ, Xue B. Safety verification for neural networks based on set-boundary analysis. In: Proc. of the 17th Int'l Symp. on Theoretical Aspects of Software Engineering. Bristol: Springer, 2023. 248–267. [doi: 10.1007/978-3-031-35257-7_15]

[53] Sun B, Sun J, Dai T, Zhang LJ. Probabilistic verification of neural networks against group fairness. In: Proc. of the 24th Int'l Symp. on Formal Methods. Springer, 2021. 83–102. [doi: 10.1007/978-3-030-90870-6_5]

[54] Meyer PJ. Reachability analysis of neural networks using mixed monotonicity. IEEE Control Systems Letters, 2022, 6: 3068–3073. [doi: 10.1109/LCSYS.2022.3182547]

[55] Goubault E, Putot S. RINO: Robust inner and outer approximated reachability of neural networks controlled systems. In: Proc. of the 34th Int'l Conf. on Computer Aided Verification. Haifa: Springer, 2022. 511–523. [doi: 10.1007/978-3-031-13185-1_25]

[56] Nicholas FL. A survey of repair strategies for deep neural networks [MS. Thesis]. Ames: Iowa State University, 2022.

[57] Zhang H, Chan WK. Apricot: A weight-adaptation approach to fixing deep learning models. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 376–387. [doi: 10.1109/ASE.2019.00043]

[58] Sinitsin A, Plokhotnyuk V, Pyrkin D, Popov S, Babenko A. Editable neural networks. In: Proc. of the 8th Int'l Conf. on Learning Representations (ICLR). Addis Ababa: OpenReview.net, 2020.

[59] Goldberger B, Adi Y, Keshet J, Katz G. Minimal modifications of deep neural networks using verification. In: Proc. of the 23rd Int'l Conf. on Logic for Programming, Artificial Intelligence and Reasoning. Alicante, 2020. 260–278. [doi: 10.29007/699q]

[60] 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]

[61] Leino K, Fromherz A, Mangal R, Fredrikson M, Parno B, Păsăreanu C. Self-repairing neural networks: Provable safety for deep networks via dynamic repair. arXiv:2107.11445, 2021.

[62] Dong GL, Sun J, Wang XG, Wang XY, Dai T. Towards repairing neural networks correctly. In: Proc. of the 21st IEEE Int'l Conf. on Software Quality, Reliability and Security (QRS). Hainan: IEEE, 2021. 714–725. [doi: 10.1109/QRS54544.2021.00081]

[63] Henriksen P, Leofante F, Lomuscio A. Repairing misclassifications in neural networks using limited data. In: Proc. of the 37th ACM/SIGAPP Symp. on Applied Computing. ACM, 2022. 1031–1038. [doi: 10.1145/3477314.3507059]

[64] Majd K, Zhou SY, Amor HB, Fainekos G, Sankaranarayanan S. Local repair of neural networks using optimization. arXiv:2109.14041, 2021.

[65] Usman M, Gopinath D, Sun YC, Noller Y, Păsăreanu CS. NNrepair: Constraint-based repair of neural network classifiers. In: Proc. of

the 33rd Int'l Conf. on Computer Aided Verification. Los Angeles: Springer, 2021. 3–25. [doi: 10.1007/978-3-030-81685-8_1]

[66] Refaeli I, Katz G. Minimal multi-layer modifications of deep neural networks. In: Proc. of the 5th Int'l Workshop and the 15th Int'l Workshop Software Verification and Formal Methods for ML-enabled Autonomous Systems. Haifa: Springer, 2022. 46–66. [doi: 10.1007/978-3-031-21222-2_4]

[67] Sun S, Yan J, Yan RJ. Layer-specific repair of neural network classifiers. In: Proc. of the 31st Int'l Conf. on Artificial Neural Networks and Machine Learning. Bristol: Springer, 2022. 550–561. [doi: 10.1007/978-3-031-15919-0_46]

[68] Cohen D, Strichman O. Automated repair of neural networks. arXiv:2207.08157, 2022.

[69] Yang XD, Yamaguchi T, Tran HD, Hoxha B, Johnson TT, Prokhorov D. Neural network repair with reachability analysis. In: Proc. of the 20th Int'l Conf. on Formal Modeling and Analysis of Timed Systems. Warsaw: Springer, 2022. 221–236. [doi: 10.1007/978-3-031-15839-1_13]

[70] Bauer-Marquart F, Boetius D, Leue S, Schilling C. SpecRepair: Counter-example guided safety repair of deep neural networks. In: Proc. of the 28th Int'l Symp. on Model Checking Software. Springer, 2022. 79–96. [doi: 10.1007/978-3-031-15077-7_5]

[71] Sohn J, Kang S, Yoo S. Arachne: Search-based repair of deep neural networks. ACM Trans. on Software Engineering and Methodology, 2022, 32(4): 85. [doi: 10.1145/3563210]

[72] Liang Z, Wu TR, Zhao CY, Liu WW, Xue B, Yang WJ, Wang J. Repairing deep neural networks based on behavior imitation. arXiv:2305.03365, 2023.

[73] Fu FS, Li WC. Sound and complete neural network repair with minimality and locality guarantees. In: Proc. of the 10th Int'l Conf. on Learning Representations. OpenReview.net, 2022.

[74] Tao Z, Nawas S, Mitchell J, Thakur AV. Architecture-preserving provable repair of deep neural networks. Proc. of the ACM on Programming Languages, 2023, 7(PLDI): 124. [doi: 10.1145/3591238]

[75] 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]

[76] Wu HH, Li Z, Cui ZQ, Zhang JM. A mutation-based approach to repair deep neural network models. In: Proc. of the 8th Int'l Conf. on Dependable Systems and Their Applications (DSA). Yinchuan: IEEE, 2021. 730–731. [doi: 10.1109/DSA52907.2021.00106]

[77] Tokui S, Tokumoto S, Yoshii A, Ishikawa F, Nakagawa T, Munakata K, Kikuchi S. NeuRecover: Regression-controlled repair of deep neural networks with training history. In: Proc. of the 2022 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering. Honolulu: IEEE, 2022. 1111–1121. [doi: 10.1109/SANER53432.2022.00128]

[78] Qiu XP. Neural Networks and Deep Learning. Beijing: China Machine Press, 2020 (in Chinese).

[79] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 1989, 2(4): 303–314. [doi: 10.1007/BF02551274]

[80] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2(5): 359–366. [doi: 10.1016/0893-6080(89)90020-8]

[81] Ziegler GM. Lectures on Polytopes, Vol. 152. New York: Springer Science & Business Media, 2012. [doi: 10.1007/978-1-4613-8431-1]

[82] Zhang YH, Chen YF, Cheung SC, Xiong YF, Zhang L. An empirical study on TensorFlow program bugs. In: Proc. of the 27th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis (ISSTA). Amsterdam: ACM, 2018. 129–140. [doi: 10.1145/3213846.3213866]

[83] Islam MJ, Nguyen G, Pan R, Rajan H. A comprehensive study on deep learning bug characteristics. In: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Tallinn: ACM, 2019. 510–520. [doi: 10.1145/3338906.3338955]

[84] Islam MJ, Pan R, Nguyen G, Rajan H. Repairing deep neural networks: Fix patterns and challenges. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Soul: ACM, 2020. 1135–1146. [doi: 10.1145/3377811.3380378]

[85] Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. Journal of Big Data, 2019, 6(1): 60. [doi: 10.1186/s40537-019-0197-0]

[86] Semenoglou AA, Spiliotis E, Assimakopoulos V. Data augmentation for univariate time series forecasting with neural networks. Pattern Recognition, 2023, 134: 109132. [doi: 10.1016/j.patcog.2022.109132]

[87] Huang K, Xu ZZ, Yang S, Sun HY, Li XJ, Yan Z, Zhang YQ. A survey on automated program repair techniques. arXiv:2303.18184, 2023.

[88] Huq F, Hasan M, Haque MMA, Mahbub S, Iqbal A, Ahmed T. Review4Repair: Code review aided automatic program repairing. Information and Software Technology, 2022, 143: 106765. [doi: 10.1016/j.infsof.2021.106765]

[89] Ji S, Choi SM, Ko SK, Kim D, Im H. RepCoder: An automated program repair framework for probability-based program synthesis. In: Proc. of the 37th ACM/SIGAPP Symp. on Applied Computing. ACM, 2022. 1554–1561. [doi: 10.1145/3477314.3507083]

[90] Yao J, Rao BB, Xing WW, Wang LQ. Bug-transformer: Automated program repair using attention-based deep neural network. Journal of Circuits, Systems and Computers, 2022, 31(12): 2250210. [doi: 10.1142/S0218126622502103]

[91] Wang WS, Wu C, Cheng L, Zhang Y. Tea: Program repair using neural network based on program information attention matrix. arXiv:2107.08262, 2021.

[92] Li Y, Wang SH, Nguyen TN. Improving automated program repair using two-layer tree-based neural networks. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 316–317. [doi: 10.1145/3377812.3390896]

[93] Riedmiller M, Braun H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proc. of the 1993 Int'l Conf. on Neural Networks. San Francisco: IEEE, 1993. 586–591. [doi: 10.1109/ICNN.1993.298623]

[94] Bernstein J, Wang YX, Azizzadenesheli K, Anandkumar A. signSGD: Compressed optimisation for non-convex problems. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 560–569.

[95] Dauphin YN, de Vries H, Chung J, Bengio Y. RMSProp and equilibrated adaptive learning rates for non-convex optimization. arXiv:1502.04390, 2015.

[96] Yang XD, Johnson TT, Tran HD, Yamaguchi T, Hoxha B, Prokhorov D. Reachability analysis of deep ReLU neural networks using facet-vertex incidence. In: Proc. of the 24th ACM Int'l Conf. on Hybrid Systems: Computation and Control. Nashville: ACM, 2021. 18. [doi: 10.1145/3447928.3456650]

[97] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan, 2016.

[98] Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S. A survey of Monte Carlo tree search methods. IEEE Trans. on Computational Intelligence and AI in Games, 2012, 4(1): 1–43 [doi: 10.1109/TCIAIG.2012.2186810]

[99] Sotoudeh M, Tao Z, Thakur AV. SyReNN: A tool for analyzing deep neural networks. Int'l Journal on Software Tools for Technology Transfer, 2023, 25(2): 145–165. [doi: 10.1007/s10009-023-00695-1]

[100] Jarre F, Vavasis SA. Convex optimization. In: Atallah MJ, Blanton M, eds. Algorithms and Theory of Computation Handbook: General Concepts and Techniques. 2nd ed., Boca Raton: Chapman & Hall/CRC, 2010.

[101] Shostak RE. A practical decision procedure for arithmetic with function symbols. Journal of the ACM, 1979, 26(2): 351–360. [doi: 10.1145/322123.322137]

[102] Freire RA. First-order logic and first-order functions. Logica Universalis, 2015, 9(3): 281–329. [doi: 10.1007/s11787-015-0126-8]

[103] Eniser HF, Gerasimou S, Sen A. DeepFault: Fault localization for deep neural networks. In: Proc. of the 22nd Int'l Conf. on Fundamental Approaches to Software Engineering. Prague: Springer, 2019. 171–191. [doi: 10.1007/978-3-030-16722-6_10]

[104] Jones JA, Harrold MJ. Empirical evaluation of the tarantula automatic fault-localization technique. In: Proc. of the 20th IEEE/ACM Int'l Conf. on Automated Software Engineering. Long Beach: ACM, 2005. 273–282. [doi: 10.1145/1101908.1101949]

[105] Ochiai A. Zoogeographical studies on the soleoid fishes found in Japan and its neighbouring regions-I. Nippon Suisan Gakkaishi, 1957, 22(9): 522–525. [doi: 10.2331/suisan.22.522]

[106] Wong WE, Debroy V, Gao RZ, Li YH. The DStar method for effective software fault localization. IEEE Trans. on Reliability, 2014, 63(1): 290–308. [doi: 10.1109/TR.2013.2285319]

[107] Le Goues C, Nguyen T, Forrest S, Weimer W. GenProg: A generic method for automatic software repair. IEEE Trans. on Software Engineering, 2012, 38(1): 54–72. [doi: 10.1109/TSE.2011.104]

[108] Forrest S. Genetic algorithms: Principles of natural selection applied to computation. Science, 1993, 261(5123): 872–878. [doi: 10.1126/science.8346439]

[109] Gopinath D, Converse H, Pasareanu C, Taly A. Property inference for deep neural networks. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 797–809. [doi: 10.1109/ASE.2019.00079]

[110] Sen K, Marinov D, Agha G. CUTE: A concolic unit testing engine for C. In: Proc. of the 10th European Software Engineering Conf. Held Jointly with the 13th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. Lisbon: ACM, 2005. 263–272. [doi: 10.1145/1081706.1081750]

[111] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341–359. [doi: 10.1023/A:1008202821328]

[112] Weimer W, Nguyen T, Le Goues C, Forrest S. Automatically finding patches using genetic programming. In: Proc. of the 31st Int'l Conf. on Software Engineering. Vancouver: IEEE, 2009. 364–374. [doi: 10.1109/ICSE.2009.5070536]

[113] Yuan Y, Banzhaf W. ARJA: Automated repair of java programs via multi-objective genetic programming. IEEE Trans. on Software Engineering, 2020, 46(10): 1040–1067. [doi: 10.1109/TSE.2018.2874648]

[114] Chattopadhyay A, Manupriya P, Sarkar A, Balasubramanian VN. Neural network attributions: A causal perspective. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 981–990.

[115] Zhang JZ, Bareinboim E. Fairness in decision-making—The causal explanation formula. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence and the 30th Innovative Applications of Artificial Intelligence Conf. and the 8th AAAI Symp. on Educational Advances in Artificial Intelligence. New Orleans: AAAI, 2018. 248.

[116] Poli R, Kennedy J, Blackwell T. Particle swarm optimization: An overview. Swarm Intelligence, 2007, 1(1): 33–57. [doi: 10.1007/s11721-007-0002-0]

[117] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

[118] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

[119] Krizhevsky A, Hinton GE. Learning multiple layers of features from tiny images. 2009. https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf

[120] Huang GB, Mattar M, Berg T, Learned-Miller E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In: Proc. of the 2008 Workshop on faces in 'Real-Life' Images: Detection, alignment, and recognition, Erik Learned-Miller and Andras Ferencz and Frédéric Jurie. Marseille, 2008.

[121] Hendrycks D, Zhao K, Basart S, Steinhardt J, Song D. Natural adversarial examples. In: Proc. of the 2021 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Nashville: IEEE, 2021. 15257–15266. [doi: 10.1109/CVPR46437.2021.01501]

[122] Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360, 2016.

[123] Census income. UCI Machine Learning Repository. 1996. https://doi.org/10.24432/C5GP7S

[124] Statlog (German Credit Data). UCI Machine Learning Repository. 1994. https://doi.org/10.24432/C5NC77

[125] Bank marketing. UCI Machine Learning Repository. 2012. https://doi.org/10.24432/C5K306

[126] Singh G. ETH robustness analyzer for neural networks (ERAN). 2020. https://github.com/eth-sri/eran

[127] Mu N, Gilmer J. MNIST-C: A robustness benchmark for computer vision. arXiv:1906.02337, 2019.

[128] Julian KD, Kochenderfer MJ, Owen MP. Deep neural network compression for aircraft collision avoidance systems. Journal of Guidance, Control, and Dynamics, 2019, 42(3): 598–608. [doi: 10.2514/1.G003724]

[129] Julian KD, Kochenderfer MJ. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In: Proc. of the 38th IEEE/AIAA Digital Avionics Systems Conf. (DASC). San Diego: IEEE, 2019. 1–10. [doi: 10.1109/DASC43569.2019.9081748]

[130] Li RJ, Li JL, Huang CC, Yang PF, Huang XW, Zhang LJ, Xue B, Hermanns H. PRODeep: A platform for robustness verification of deep neural networks. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. ACM, 2020. 1630–1634. [doi: 10.1145/3368089.3417918]

[131] Song XD, Sun YC, Mustafa MA, Cordeiro LC. AIREPAIR: A repair platform for neural networks. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering: Companion Proc. Melbourne: IEEE, 2023. 98–101. [doi: 10.1109/ICSE-Companion58688.2023.00033]

[132] Guidotti D, Pulina L, Tacchella A. NeVer 2.0: Learning, verification and repair of deep neural networks. arXiv:2011.09933, 2020.

[133] Yu B, Qi H, Guo Q, Juefei-Xu F, Xie XF, Ma L, Zhao JJ. DeepRepair: Style-guided repairing for deep neural networks in the real-world operational environment. IEEE Trans. on Reliability, 2022, 71(4): 1401–1416. [doi: 10.1109/TR.2021.3096332]

[134] Majd K, Clark G, Khandait T, Zhou SY, Sankaranarayanan S, Fainekos G, Amor HB. Safe robot learning in assistive devices through neural network repair. In: Proc. of the 6th Conf. on Robot Learning (CoRL). Auckland: PMLR, 2023. 2148–2158.

[135] Tanno R, Pradier MF, Nori A, Li YZ. Repairing neural networks by leaving the right past behind. In: Proc. of the 36th Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 13132–13145.

附中文参考文献:

[21] 刘颖, 杨鹏飞, 张立军, 吴志林, 冯元. 前馈神经网络和循环神经网络的鲁棒性验证综述. 软件学报, 2023, 34(7): 3134–3166. http://www.jos.org.cn/1000-9825/6863.htm [doi: 10.13328/j.cnki.jos.006863]

[50] 梁震, 刘万伟, 吴陶然, 任德金, 薛白. 鲁棒神经网络的训练方法研究进展与前景. 前瞻科技, 2023, 2(1): 78–89. [doi: 10.3981/j.issn.2097-0781.2023.01.006]

[78] 邱锡鹏. 神经网络与深度学习. 北京: 机械工业出版社, 2020.

梁震(1997－), 男, 博士生, 主要研究领域为人工智能形式化验证与可解释性.

薛白(1986－), 男, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为混成系统, 人工智能的形式化验证.

刘万伟(1980－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为自动机理论, 形式化方法, 人工智能形式化验证.

王戟(1969－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为软件方法学, 软件分析与验证, 并行与分布计算.

吴陶然(2000－), 男, 硕士生, 主要研究领域为混成系统, 人工智能的形式化验证.

杨文婧(1988－), 女, 博士, 副研究员, 主要研究领域为机器学习, 机器人软件, 高性能计算.