软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(11):5008−5028 [doi: 10.13328/j.cnki.jos.006679]

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

面向深度学习系统的模糊测试技术研究进展*

代贺鹏, 孙昌爱, 金 慧, 肖明俊

(北京科技大学 计算机与通信工程学院, 北京 100083)

通信作者: 孙昌爱, E-mail: casun@ustb.edu.cn

摘 要: 深度学习系统具有强大的学习与推理能力, 在无人驾驶、语音识别和机器人等领域应用广泛. 由于数据集的限制以及依赖人工标签数据, 深度学习系统易于出现非预期的行为. 近年来, 深度学习系统的质量问题受到广泛的关注, 特别是在安全攸关的领域. 由于模糊测试具有较强的故障揭示能力, 运用模糊测试技术对深度学习系统进行测试成为研究热点. 从测试用例生成 (包括种子队列构建、种子选择和种子变异)、测试结果判定、覆盖分析 3 个方面对已有的深度学习系统的模糊测试技术进行总结, 并介绍常用的数据集以及度量指标, 最后对其发展方向进行展望.

关键词: 深度学习系统; 模糊测试; 研究进展

中图法分类号: TP311

中文引用格式: 代贺鹏, 孙昌爱, 金慧, 肖明俊. 面向深度学习系统的模糊测试技术研究进展. 软件学报, 2023, 34(11): 5008–5028. http://www.jos.org.cn/1000-9825/6679.htm

英文引用格式: Dai HP, Sun CA, Jin H, Xiao MJ. State-of-the-art Survey on Fuzz Testing for Deep Learning System. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 5008–5028 (in Chinese). http://www.jos.org.cn/1000-9825/6679.htm

State-of-the-art Survey on Fuzz Testing for Deep Learning System

DAI He-Peng, SUN Chang-Ai, JIN Hui, XIAO Ming-Jun

(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract: Deep learning (DL) systems have powerful learning and reasoning capabilities and are widely employed in many fields including unmanned vehicles, speech recognition, intelligent robotics, etc. Due to the dataset limit and dependence on manually labeled data, DL systems are prone to unexpected behaviors. Accordingly, the quality of DL systems has received widespread attention in recent years, especially in safety-critical fields. Fuzz testing with strong fault-detecting ability is utilized to test DL systems, which becomes a research hotspot. This study summarizes existing fuzz testing for DL systems in the aspects of test case generation (including seed queue construction, seed selection, and seed mutation), test result determination, and coverage analysis. Additionally, commonly used datasets and metrics are introduced. Finally, the study prospects for the future development of this field.

Key words: deep learning (DL) system; fuzz testing; state-of-the-art survey

当今社会正从以单机应用为主要特征的数字化阶段, 以互联网为主要特征的网络化阶段, 过渡到以深度数据挖掘与融合应用为主要特征的智能化新阶段[1]. 基于深度学习算法的智能系统具有强大的推理与分析能力, 在自然语言处理[2]、图像分类[3]、游戏[4]等多个领域表现出不低于人类的智能. 深度学习系统在无人驾驶[5]、恶意软件检测[6]、飞机避碰系统[7]等安全攸关的系统中担任重要角色. 然而, 深度学习系统经常表现出不正确或不被期望的行为, 导致巨大的经济损失甚至造成人员伤亡[8].

软件测试是提高深度学习系统质量的一种重要手段. 传统软件的决策逻辑由开发人员编写的代码确定; 深度

* 基金项目: 国家自然科学基金 (61872039); 北京市自然科学基金 (4162040); 航空科学基金 (2016ZD74004)
收稿时间: 2021-08-18; 修改时间: 2021-12-23; 采用时间: 2022-03-15; jos 在线出版时间: 2022-05-24
CNKI 网络首发时间: 2023-04-28

学习系统则从大量数据中获取决策逻辑. 传统软件的决策逻辑可以通过控制流图进行分析[9], 然而深度学习系统的决策逻辑通常体现在连接不同层神经元的边的权重以及激活函数上[10]. 这些不同点导致了传统软件的测试技术与工具不再适用于深度学习系统.

人工收集尽可能多的测试数据并标记数据是深度学习系统缺陷检测常见方法[11,12]. 该方法需要消耗大量的测试资源, 并且收集的数据的正确性和多样性难以保障. 研究人员根据深度学习系统的特点, 对传统的软件测试技术改进并用于检测深度学习系统的缺陷[13], 例如符号执行[14]、蜕变测试[10,15,16]、基于梯度的测试方法[13]和模糊测试[17,18]等. 符号执行技术能够高效、精确地揭示传统程序故障, 应用于深度学习系统时面临如下挑战[14]: (1) 深度学习模型通常没有分支; (2) 深度学习系统的非线性程度普遍较高, 难以找到合适的约束求解器; (3) 深度学习系统的结构复杂, 超出了目前符号推理工具的能力范围. 应对这些挑战的一个可行方案是将深度学习模型转化为程序[13]. 蜕变测试利用多个输入与输出之间的关系 (蜕变关系) 判定测试结果, 缓解了测试预期问题, 在深度学习系统的测试中受到了广泛的关注. 深度学习系统本质是一个复杂的数学模型, 与其相关的函数 (例如损失函数、神经元的激活函数) 的梯度易于获取, 基于梯度的测试方法广泛应用于深度学习系统的测试中[13]. 模糊测试在传统软件测试中表现出较强的故障揭示能力[19,20], 受到 Adobe[21]、Cisco[22]、Google[23−25]、Microsoft[26,27]等的关注, 逐渐成为工业界的标准测试技术[28]. 近年来, 面向深度学习系统的模糊测试研究逐渐引起人们的关注[13,18]. 与符号执行相比, 模糊测试不需要将深度学习系统转化为程序, 节省了测试资源. 尽管基于梯度的测试方法能够有效地揭示深度学习模型存在的故障, 现有研究结果表明模糊测试能够检测到基于梯度的测试方法无法检测的故障[1]. 与蜕变测试相比, 模糊测试的主要优势是能够在有限的测试用例中产生大量的测试用例, 劣势是不具备测试结果判定机制.

人们开始对机器学习系统的测试技术进行了归纳与总结[13,29−31], 分析了常用的机器学习系统的测试技术, 关注机器学习系统在特定领域的测试方法. Zhang 等人[13]较为全面地调研了机器学习系统的测试技术, 包括机器学习系统属性 (正确性、鲁棒性和公平性等) 的测试方法、机器学习组件 (数据、学习程序和框架) 的测试方法、机器学习系统测试流程相关的研究 (测试用例生成、测试结果判定、测试充分性评估等) 和不同领域的机器学习系统的测试方法 (自动驾驶、机器翻译和自然语言处理); Huang 等人[29]调研了深度学习系统的验证、测试、对抗攻击、防御以及可解释性方面的工作; Tahir 等人[30]调研了基于覆盖的无人驾车测试方法; Riccio 等人[31]调研了机器学习系统的功能测试技术, 依据研究背景、特征和经验评估对调研的技术进行分类. 然而, 至今尚不存在深度学习系统的模糊测试相关的综述. 本文尝试全面、系统地总结面向深度学习系统的模糊测试技术. 具体说来, 首先使用系统文献调研方法收集当前面向深度学习系统的模糊测试相关的研究论文; 分别围绕测试用例生成 (包括种子队列构建、种子选择和种子变异)、测试结果判定和覆盖分析 3 个方面, 总结已有的测试技术和结论并介绍常用的数据集以及度量指标; 在此基础上, 讨论面向深度学习系统的模糊测试研究现状和分析未来的研究方向.

本文第 1 节介绍深度学习系统与模糊测试相关的背景知识. 第 2 节介绍深度学习系统的模糊测试研究相关的文献收集与汇总情况. 第 3 节从测试用例生成 (包括种子队列构建、种子选择和种子变异)、测试结果判定和覆盖分析 3 个方面, 讨论深度学习系统的模糊测试技术研究进展. 第 4 节总结深度学习系统的模糊测试技术评估相关的模型与数据集. 第 5 节归纳与总结已有的评估指标. 第 6 节总结研究现状并对未来进行展望.

1 背景介绍

本节首先介绍深度学习系统, 然后介绍面向深度学习系统的模糊测试框架与关键组件.

1.1 深度学习系统

深度学习系统是指至少包含一个深度神经网络组件的软件系统或者全部由深度神经网络组成的系统[32]. 深度学习系统的一般架构如图 1 所示, 主要包含感知设备、数据获取组件、数据处理组件和深度学习模型组件.

(1) 感知设备. 深度学习系统代替人类在特定的环境下进行决策时, 需要获取环境信息. 感知设备收集深度学习系统决策所必需的信息, 常见的感知设备包括传感器、雷达、摄像机、超声波等. 感知设备将获取的环境信息转化为图片、音频等数字信息.

(2) 数据获取组件. 该组件接受感知设备获取的数字信息, 并将其发送给数据处理组件.

(3) 数据处理组件. 该组件对非结构化数据进行预处理 (例如归一化和标准化等), 将其转化为深度学习模型识别的结构化数据.

(4) 深度学习模型. 该组件以结构化数据作为输入, 经过多层神经元的计算输出决策结果, 控制深度学习系统的行为.

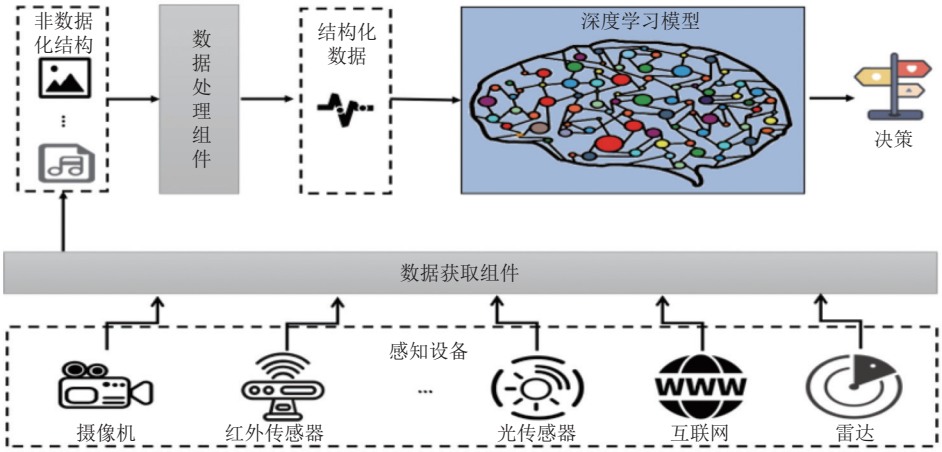

图 1 深度学习系统的架构

深度学习模型类似人类的大脑, 包含多个层. 每一层由数量庞大的神经元组成, 可以自动地从原始数据中识别、提取高水平的特征. 由于硬件技术的飞速发展[33−35]、高效的算法[3,36−38]和大量的训练数据集[11,39,40], 深度学习系统在诸多领域表现出不输于人类的智能.

神经元是深度学习模型的计算单元, 通过激活函数输入数据并输出计算结果. 常见的激活函数包括 Sigmoid、ReLU 等[41]. 图 2 展示了一个最简单的深度学习模型, 包括一个输入层、一个输出层和一个隐藏层. 每一层的神经元与下一层的所有神经元相连接, 每一个连接对应一个权重. 权重表明了神经元之间关联性的强弱, 通过在训练的过程中最小化代价函数获得.

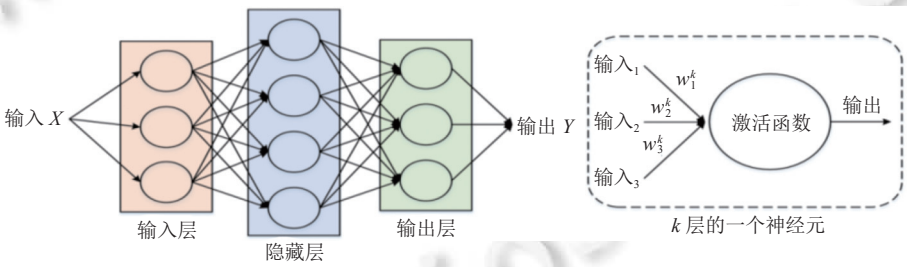

图 2 一个简单的深度学习模型和每一个神经元执行的计算[31]

多个神经元组成的层可以将输入中包含的信息转化为数据的高级表示. 例如, 在一个图像分类任务中, 靠近输入层的隐藏层将图像的像素值转化为低级的纹理特征 (例如边缘和颜色等), 并将其发送给更深的层. 最后几层依次提取、组合有意义的高级抽象特征 (例如眼睛和鼻子等) 并将其发送给输出层, 由输出层做出分类决策[42].

与传统软件获取决策逻辑的方式 (即开发者编写代码与确定执行逻辑) 不同, 深度学习模型定义了一种新的开发范式: 从数据中学习决策逻辑. 开发者只能通过改变训练数据、特征和模型结构的方式间接影响模型的决策逻辑 (如图 3 所示).

此外深度学习模型的决策逻辑表现在连接神经元边的权重上. 这些新特点为深度学习模型的测试研究带来了挑战: (1) 深度学习模型的决策逻辑不能通过控制流进行分析, 导致以最大化语句或者分支覆盖为目的的传统软件

测试技术不再适用于深度学习模型; (2) 深度学习模型中包含大量浮点计算以及非线性公式, 导致可满足性模理论求解器不能为深度学习模型生成测试用例[43]; (3) 深度学习模型往往解决现实中非常复杂问题 (例如无人驾驶、飞机避碰等), 开发人员难以理解深度学习模型的决策逻辑. 为了应对上述深度学习模型测试的挑战, 人们提出了各种面向深度学习模型的测试技术.

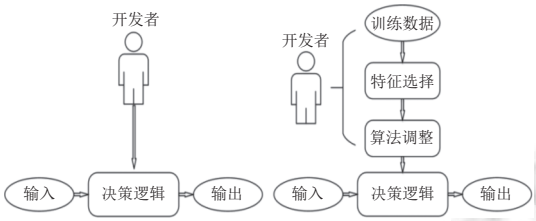

图 3 传统软件与深度学习模型获取决策逻辑的方式[32]

1.2 面向深度学习系统的模糊测试

模糊测试首先是由 Miller 等人[17]提出, 核心思想是利用少量有效的测试用例作为种子, 产生大量的格式良好、不符合预期的测试数据, 尽可能地覆盖边界情况[19,20,44]. 关键步骤是生成合适的测试用例. 常用的测试用例生成方法包括面向覆盖的测试用例生成方法、遗传算法、符号执行、污点分析等[19]. 图 4 展示了模糊测试的一般流程, 方框内是模糊测试的关键步骤, 方框外是种子文件、规格说明、目标程序和故障报告.

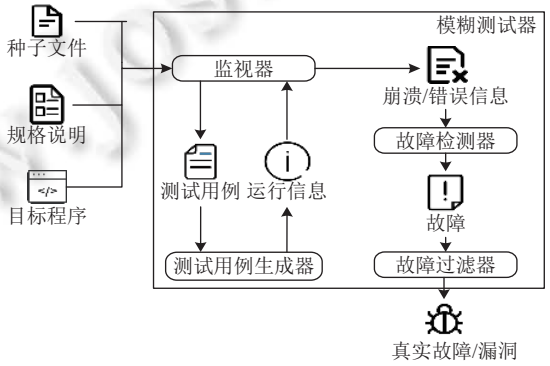

图 4 模糊测试的一般过程[19]

(1) 监视器通常内置在白盒或灰盒模糊测试技术中, 利用代码检测、污点分析等技术获取代码覆盖率、污点数据流或待测程序的其他有用的运行时信息.

(2) 测试用例生成器生成测试用例的主要方法有两种: 基于变异的方法和基于语法的方法[44]. 基于变异的方法借助随机变异格式良好的种子文件或使用根据运行时收集的待测程序信息, 调整变异策略并生成测试用例. 基于语法的方法根据规范说明 (例如语法) 自动生成测试用例. 在很多情况下, 测试用例通常是半有效数据, 这些数据的有效部分足以通过待测程序的早期解析阶段, 而无效部分用来触发待测程序深层逻辑中的缺陷.

(3) 故障检测器帮助用户在目标程序中发现潜在的错误. 当待测程序崩溃或报告某些错误时, 该模块收集并分析相关信息 (例如堆栈跟踪[45]) 来确定是否存在错误.

(4) 故障过滤器从所有报告的故障中提取用户感兴趣的故障 (例如与正确性和安全性相关的故障).

深度学习模型从数据中学习决策逻辑, 因此数据对于深度学习系统的开发与测试有重要的影响. 实践中, 通常由数据工程师收集带标签的数据构建训练数据和测试数据集[39]. 然而, 收集数据的开销是十分高昂的. 基于变异的模糊测试可以在有限的种子 (原始数据) 基础上自动生成大量的测试用例 (利用变异策略), 节省了测试资源, 并且具有高效的故障揭示能力[10,18]. 图 5 展示了基于变异的深度学习系统模糊测试框架. 该框架主要包含种子选择策略、种子变异策略、测试结果判定和覆盖分析这 4 个组件.

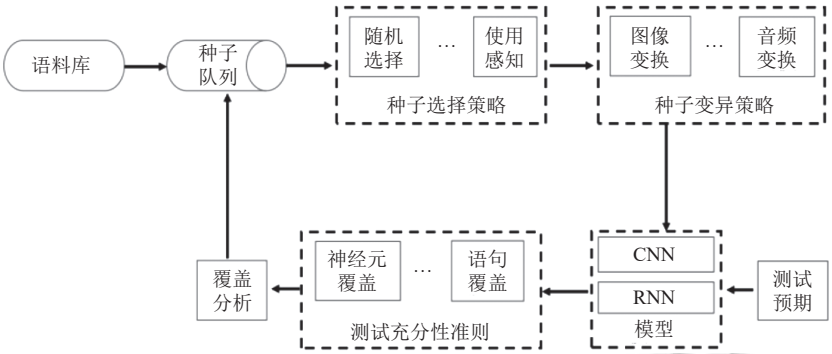

图 5 面向深度学习系统的模糊测试框架

(1) 种子选择策略根据预先定义的规则从种子队列中选择种子, 并将选择的种子发送给种子变异策略组件. 已有的工作主要采用顺序或者随机的方式从种子队列中选择种子.

(2) 种子变异策略是当前面向深度学习系统的模糊测试研究热点. 可以利用领域知识[10]、白噪声[16]、生成对抗网络[46]、神经元的覆盖信息[18,32]等设计变异策略, 限制变异程度使得变异后的数据与原始种子具有语义不变性.

(3) 测试结果判定回答深度学习系统的测试结果是否通过. 由于测试人员难以理解深度学习模型的决策逻辑, 并且深度学习系统往往帮助人类解决不确定问题[13], 测试结果判定是深度学习系统的模糊测试难点. 差异测试[31]和蜕变测试[15,47−49]是目前广泛应用的缓解深度学习系统测试预期问题的方法.

(4) 覆盖分析主要利用深度学习系统的测试充分性准则, 分析变异种子执行深度学习模型的情况, 提取有价值的信息. 这些信息不仅可以作为是否将变异种子添加到种子队列的依据, 还可以指导种子变异[18].

基于上述面向深度学习系统的模糊测试框架, 我们进一步讨论模糊测试的主要操作步骤, 如算法 1 所示. 算法 1 的输入是种子队列 I、待测的深度学习模型 DNN 以及预先定义的种子变异次数 k, 输出是揭示模型缺陷的变异种子集合 F. 在执行测试的过程中, 首先从种子队列中依据种子选择策略选择一个种子 s (算法 1 的第 3 行), 利用变异策略对 s 进行 k 次变异, 得到变异种子集合 T, 执行所有的变异种子并记录每个种子的覆盖信息 cov 和执行结果 $result$ (算法 1 的第 4–6 行); 然后分析变异种子 s*的执行结果: 如果 s*揭示了待测模型的缺陷, 将 s*加入 F 中; 如果 s*增加了新的覆盖, 将 s*加入种子队列中 (算法 1 的第 7–11 行).

算法 1. The main steps of fuzzing for deep learning system.

输入: I: seed queue; D: DNN under test; k: mutation times;
输出: F: a set of successful adversarial inputs.

1. $F \leftarrow \varnothing$;
2. $Q \leftarrow I$;
3. while $s \leftarrow selectSeed(Q)$ do
4. $\quad Q \leftarrow Q \setminus \{s\}$;
5. $\quad$ Mutate selected seed (s) k times: $T \leftarrow mutateSeed(s, k)$;
6. $\quad$ Execute mutated seeds: cov, $result \leftarrow run(D, T)$;
7. $\quad$ for each s* in T do
8. $\quad\quad$ if $isFailedTest(s^*, result)$ then
9. $\quad\quad\quad F \leftarrow F \cup \{s^*\}$;
10. $\quad\quad$ else if $isCovergeGain(cov)$ then
11. $\quad\quad\quad Q \leftarrow Q.append(s^*)$;

```
12.       end_if
13.    end_for
14. end_while
```

总体说来, 面向深度学习系统模糊测试的实施步骤与传统模糊测试基本步骤相似. 另一方面, 深度学习系统的新特点使得模糊测试面临一些新挑战. 具体内容如下.

(1) 符合深度学习系统特点的种子队列构建: 由于种子影响模糊测试探索待测模型内部状态, 构建的种子队列需要具有多样性. 传统模糊测试的种子多样性度量指标是面向基本的数据类型. 深度学习系统的输入格式复杂, 如何有效衡量复杂输入的多样性成为构建深度学习系统的模糊测试的种子队列构建的关键.

(2) 高效的种子选择策略: 在模糊测试过程中, 种子揭示故障与触发新覆盖的概率具有不确定性. 传统模糊测试技术采用不同的覆盖准则 (代码覆盖、分支覆盖等) 选择种子[20]. 然而, 这些覆盖准则可能无法有效地指导深度学习系统的测试[32]. 一个挑战性问题是探索高效的种子选择策略, 提升深度学习系统的故障揭示能力或新覆盖触发概率.

(3) 针对深度学习系统复杂应用场景的多样化变异策略: 深度学习模型通常与外界环境交互密切, 应用场景相较于传统软件更加复杂. 在此情况下, 传统模糊测试技术的种子变异策略 (例如位反转、字节拷贝、字节删除等[19]) 难以应对深度学习系统多样化测试场景. 一个关键问题是如何在少量测试数据基础上通过多样化变异策略生成深度学习系统的多样性测试用例.

(4) 测试结果难以判定: 深度学习系统的输出具有概率性并且触发故障的输入不会引起深度学习系统产生明显的异常 (例如系统崩溃), 从而增加了测试结果判定的难度. 如何高效、正确地判定测试结果成为深度学习系统的模糊测试亟待解决的关键问题之一.

(5) 符合深度学习系统测试特点的测试充分性指标: 深度学习系统与传统软件在编程范式上的差异使得传统软件的测试充分性指标 (语句覆盖、分支覆盖、MC/DC 覆盖等) 难以有效地评估深度学习系统的测试充分程度. 另一方面, 模糊测试依赖测试充分性指标分析变异种子的执行过程. 如何针对深度学习系统特点, 提出新的测试充分性指标成为实施深度学习系统模糊测试的关键.

自 Pei 等人[32]提出深度学习模型的白盒测试方法后, 研究者开始探索面向深度学习系统的模糊测试技术, 在测试用例生成、测试结果判定和覆盖分析等方面进行优化, 提高面向深度学习系统的模糊测试揭示故障的有效性与效率.

2 文献收集与汇总

本文在检索文献时, 主要采用了谷歌学术搜索、ACM Digital Library、IEEE Xplore Digital Library、Springer Link Online Library、ScienceDirect、DBLP 和 arXiv 等英文数据库, 以及中国知网和万方知识服务平台这两个中文数据库. 面向深度学习的模糊测试技术的研究, 属于软件工程与机器学习的交叉领域. 因此, 发表的论文主要分布在软件工程和人工智能两个领域. 检索的论文主要来自发表在软件工程和人工智能领域的国际顶级会议 ESEC/FSE、ICSE、ASE、PLDI、NIPS、IJCAI、AAAI、ICML、ACL、ICLR.

本文通过对人工智能领域与软件工程领域顶级会议/期刊自 2017–2020 年录用的论文进行统计, 从中筛选与面向深度学习的模糊测试相关的论文. 表 1 展示了检索过程中使用的中英文关键词.

对于各个数据库中检索到的文献, 应用下述标准来进行筛选. 我们将符合全部以下条件的文献考虑在内.

(1) 利用模糊测试技术检测深度神经网络模型的缺陷.

(2) 与深度学习系统测试充分性指标相关的研究.

(3) 发表形式为会议、期刊、书籍等.

将符合以下任一条件的文献排除在外.

(1) 利用机器学习算法优化模糊测试技术.

(2) 利用模糊测试技术检测深度学习的框架.

(3) 以中文和英文之外的语言发表.

深度学习系统测试作为一个新的研究领域, 部分研究工作并没有明确指明技术类别. 此外, 为了应对深度学习系统新特点带来的测试挑战, 模糊测试与其他测试技术相结合用来测试深度学习系统, 导致技术类别的界限模糊不清. 在搜集文献阶段, 我们将没有明确指明技术类别的测试技术, 如果满足模糊测试框架的特点则视为模糊测试范畴. 此外, 深度学习系统的测试充分性指标是面向深度学习系统的模糊测试的重要组件, 部分与测试充分性指标相关的研究工作由于发表时间的限制, 没有应用到模糊测试中. 但是这些测试充分性指标可以作为模糊测试候选方案. 因此, 我们收集了与深度学习系统测试充分性指标相关的研究.

表 1 论文检索关键字

英文关键字	中文关键字
machine learning AND fuzzing	“机器学习” AND (“模糊测试” OR “fuzzing”)
deep learning AND fuzzing	“深度学习” AND (“模糊测试” OR “fuzzing”)
neural networks AND fuzzing	“神经网络” AND (“模糊测试” OR “fuzzing”)
reinforcement learning AND fuzzing	“强化学习” AND (“模糊测试” OR “fuzzing”)
recurrent neural network AND fuzzing	“循环神经网络” AND (“模糊测试” OR “fuzzing”)
convolutional neural networks AND fuzzing	“卷积神经网络” AND (“模糊测试” OR “fuzzing”)

最后, 应用滚雪球方法对筛选后的文献进行进一步扩充, 即对包含文献的所有参考文献都应用上述标准进行筛选, 补充发现新的相关文献, 并重复这一步骤直到不再加入新的文献为止.

我们最终收集了 36 篇研究论文. 其中, 16 篇论文仅探讨了深度学习系统的模糊测试技术; 4 篇论文既讨论了深度学习系统的模糊测试技术且提出了新的测试充分性指标; 16 篇仅讨论了深度学习系统的测试充分性指标. 提出了不同测试充分性指标的相关论文总计 20 篇. 其中, 仅有 5 篇论文提出的测试充分性指标在深度学习系统的模糊测试技术中得到了应用, 因此, 提出测试充分性指标而未得到应用的文献占比为 15/20. 图 6 给出了这些论文的在不同的会议或者期刊上的发表情况. 从图中可以看出, 67% 的研究工作发表在软件工程领域, 例如 ICSE、ASE、ISSTA 等; 11% 的研究工作发表在人工智能领域, 例如 AAAI 和 ICLR; 14% 的工作发表在 arXiv (预印本平台) 上, 未通过同行评议; 8% 的工作发表在工程技术和网络与信息安全领域.

统计收集到的论文中经验研究的数据格式, 我们发现研究者的兴趣集中在图像处理的深度学习系统, 占所有工作的 95%; 剩下工作研究的数据格式涉及文本、安卓 APP 和视频.

我们将第一作者与该作者的所属国相互关联, 发现所有的研究工作来自 10 个国家, 并且中国、美国和新加坡的论文最多, 见表 2: 67% 的工作出自亚洲, 22% 的工作来自美洲, 11% 的工作来自欧洲. 这表明面向深度学习系统的模糊测试研究工作是由小群体构成, 但是研究者分布于全球各地.

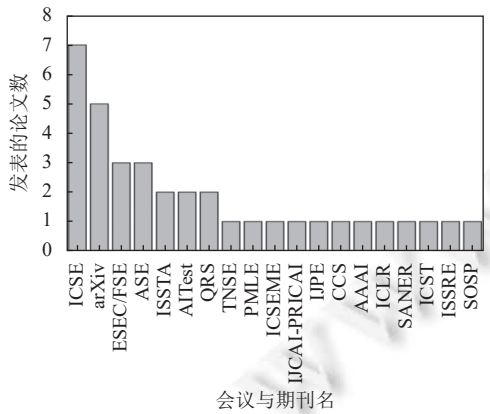

图 6 研究论文在不同的会议或者期刊上的发表情况

表 2 面向深度学习系统的模糊测试论文地域分布

国家	论文数
中国	13
美国	6
新加坡	6
韩国	4
英国	3
日本	1
加拿大	1
哥伦比亚	1
土耳其	1

3 面向深度学习系统的模糊测试技术

本节围绕面向深度学习系统模糊测试技术的关键组件, 详细介绍测试用例生成 (包括种子队列构建、种子选择和种子变异)、测试结果判定和覆盖分析等方面的技术进展.

3.1 测试用例生成

模糊测试的测试用例生成可以分为种子队列构建、种子选择和种子变异. 接下来, 我们围绕这 3 个关键步骤介绍现有的工作.

3.1.1 种子队列构建

种子队列构建是执行面向深度学习系统的模糊测试的第 1 步. 模糊测试在生成的种子队列基础上, 多次对种子进行变异操作, 得到大量待执行的变异种子. 因此, 种子队列是模糊测试的基础, 其多样性决定了模糊测试的性能.

图 7 展示的深度学习系统的决策流图, 不同层的部分神经元共同作用负责一条学习规则 (例如, 灰色神经元是深度学习模型将输入识别为人的主要贡献者). 在利用模糊测试检测深度学习系统的缺陷时, 多样性的种子队列经过变异后能够激活更多的神经元, 即检测深度学习模型学到的更多规则; 反之, 探索的神经元数目有限.

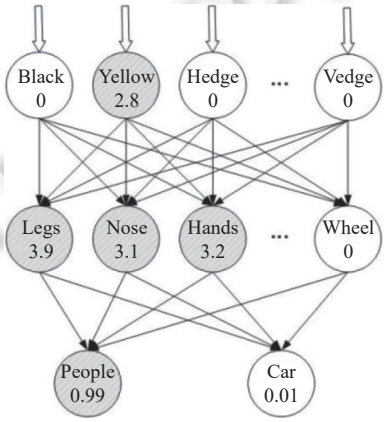

图 7 深度学习系统的决策流图

面向深度学习系统的模糊测试研究工作[10,16,18,32,49−54]采用随机的方式从语料库 (包括测试用例、测试用例的预期输出、测试用例的覆盖信息等) 中选出部分测试用例构建种子队列; 而部分工作[1,46,55−61]没有明确指出种子队列的构建策略. 因此, 利用随机的方法构建种子队列是目前已知的面向深度学习的模糊测试研究中唯一的策略. 该策略具有简单、易实现的特点, 但是没有利用任何深度学习系统、测试用例本身的特点或者其他信息, 导致在一些情况下模糊测试不够高效.

3.1.2 种子选择

模糊测试需要从构建的种子队列中迭代式地选择种子, 为种子变异做准备. 从一次选择的种子数量角度出发, 可以将种子选择策略分为单一选择和批量选择. 单一选择是指每次从种子队列中选择一个种子作为候选变异种子; 批量选择是指每次从种子队列中选择若干个种子并构成种子向量. 从选择种子的方式角度出发, 可以将种子选择策略分为随机选择[54]、新近感知[54]、频率感知[54]和聚类选择[58].

随机选择是指在种子队列中随机地选择种子 (在种子队列中顺序选择种子视为随机选择种子的特殊情况). 由于没有利用任何测试的过程信息, 该方法可能降低模糊测试揭示故障的效率. 在传统程序中, 分支之间存在层次关系[49]. 这种特殊关系使得一个种子覆盖了某个分支后, 则更有可能通过变异该种子来覆盖后续的分支. 因此, 基于覆盖的模糊测试通常选择最新加入种子队列的种子. 基于相似的想法, Odena 等人[50]提出了新近感知的种子选择策略, 即最近使用的、触发了新覆盖的种子优先被选择. 然而在一些情况下, 一个种子连续多次被选中, 可能导致

该种子的变异种子不能或者具有较小的概率触发新的覆盖. Xie 等人依据种子被选择的次数设置种子的选择概率, 使得最近触发新覆盖的种子具有较高的被选择概率, 并且当种子被选择的次数到达预先设置的阈值时, 该种子具有很小的选择概率[49]. Zhang 等人利用种子加入种子队列的时间信息, 设置种子的选择概率, 该方法的优点是最新加入种子队列的种子具有较高的被选择概率, 但是这种优势会随着时间的推移不断减少[1,46]. Demir 等人[58]提出了基于聚类算法的种子选择策略, 对语料库中的测试数据进行聚类并随机选择一个簇, 然后在选择的簇中随机选择一个或者多个测试数据进行变异. 遗憾的是, 实验评估部分没有将基于聚类的种子选择策略与已有的种子选择策略进行比较.

3.1.3 种子变异

种子变异阶段将选择的种子 (下文称为原始种子) 作为输入, 根据种子变异策略生成大量的变异种子. 变异策略直接影响模糊测试的故障揭示效率, 成为种子变异阶段的核心. 已有的面向深度学习系统的种子变异策略依据改变数据的手段可以分为基于梯度的变异方法、基于领域知识的变异方法、基于神经网络的变异方法、基于搜索的变异方法和基于关键像素值的方法等. 具体内容如下.

(1) 基于梯度的变异方法的核心思想是将测试目标作为目标函数, 计算目标函数的导数并将计算的导数作为梯度, 最后在得到的梯度基础上变异原始种子. Pei 等人将最大化神经元覆盖和不同模型的输出差异作为联合目标函数. 实现第 1 个目标的方法是每次迭代选择一个未激活的神经元, 然后改变输入的值使得神经元的输出大于选择的神经元的阈值; 实现第 2 个目标的方法是改变输入的值使得至少一个模型的输出与其他模型不同[32]. Guo 等人将最大化神经元覆盖和最大化模型预测错误的概率作为联合目标函数. 实现第 1 个目标的方法是改变输入的值来激活由 4 种种子选择策略选中的神经元. 实现第 2 个目标的方法是改变输入的值使得模型做出错误决策的概率最大[18]. Lee 等人将最大化神经元覆盖作为目标函数, 识别了 29 种神经元特征, 并将每一种神经元特征作为一个神经元选择策略, 然后提出一种动态组合神经元选择策略, 最后计算选择的神经元的梯度值并变异原始种子[56].

(2) 基于领域知识的变异方法根据输入的特点对原始种子进行变异, 不改变原始种子的语义. Tian 等人提出了 9 种图像转换策略 (包括改变亮度、改变对比度、平移、缩放、水平剪切、旋转、模糊、雾效果和雨效果) 与一种图像转换策略的贪婪组合算法. 从 9 种图像转换策略中随机选择两种策略对种子进行变异, 如果变异后的种子能够触发新的覆盖, 则选中的策略在随后的过程中赋予更大的概率[10]. Du 等人提出了不改变语义的 8 种语音转换策略, 包括添加白噪声、升高/降低音调、删除开始/结尾的沉默、随机加速/减速音频、随机调节音量、降低高分贝声音的音量或放大低分贝声音的音量、衰减频率高于阈值的声音、衰减频率低于阈值的声音[16]. Odena 等人提出了两种不改变语义的图像转换方法: 1) 在原始图像的基础上添加白噪声; 2) 在原始图像的基础上添加白噪声, 但是限制变异后的图像与原始图像的差异[50]. Xie 等人利用 8 种图像转换策略 (通过改变像素值或仿射实现) 连续多次对种子进行变异. 为了保证变异后的种子与原始种子的语义相同, Xie 等人规定仿射变换只能使用一次[49]. Du 等人利用改变语音的速度与音量、低/高频滤波、噪音混合 3 种策略变异语音, 利用图像对比度、亮度、平移、缩放、剪切、旋转和白噪声 7 种策略变异图像[52]. Park 等人在原始种子的基础上添加随机生成的噪音, 并限制变异种子与原始种子的差异[53]. Rios 利用风格转换器将标准美式英语文本转化为非裔美国人方言文本[61].

(3) 基于神经网络的变异方法利用生成对抗网络生成与原始种子相似的数据. Zhang 等人首先利用数据集中的数据训练生成对抗网络[62], 然后将原始种子输入训练后的对抗网络中, 得到变异种子[1,46].

(4) 基于搜索的变异方法利用基于搜索的算法来指导种子变异. Braiek 等人利用基于群体的启发式规则, 在初始图像转化策略集合上演化新的图像转化策略, 利用得到的图像转换策略生成变异种子[51]. Demir 等人首先利用蒙特卡罗树搜索的方法获得原始种子的变异区域与变异策略, 利用获得的变异策略变异原始种子的变异区域[58]. Gao 等人首先利用遗传算法搜索每一批种子中最可能影响深度学习模型鲁棒性的一个种子, 然后判断该种子是否值得被增强. 如是, 则用该种子的变异种子代替该种子训练模型; 否则, 继续迭代[60].

(5) 基于关键像素的变异方法的核心思想是首先识别原始种子中的关键像素值, 然后有针对地改变原始种子的像素值, 生成对抗输入. Wang 等人利用规模不变的特征转换算法[63]识别图像的关键点, 基于这些关键点拟合高斯分布模型来变异其他像素值[59]. Zhang 等人基于分类器的原理, 在满足一定距离的原始种子空间中, 通过改变原始种子的像素值获得对抗输入[55]. Qin 等人设计了两个模糊测试器, 根据原始种子与指导种子 (与原始种子不处于

同一类别的任意种子) 的差异, 利用攻击指导策略不断改变原始种子的像素值来生成对抗输入, 进一步在对抗输入附近找到更多的对抗输入[57].

3.2 测试结果判定

深度学习系统在构建过程中输入、初始化权重等多种不确定性因素以及决策逻辑的难以理解, 导致深度学习系统的输出具有不确定性, 测试结果难以判定. 根据深度学习系统的特点, 研究人员提出了一些能够缓解深度学习系统测试预期问题的方法, 包括蜕变关系/蜕变转换/蜕变变异与差分测试. 此外, 通常将“在原始测试用例的基础上, 添加人眼不可识别的扰动得到的新输入和原始输入的执行结果一样”这一观察作为测试预期[1,18,46,50,53,55−61].

3.2.1 利用蜕变关系判定测试结果

依据待测软件的蜕变属性获取蜕变关系, 执行原始测试用例 (采用测试用例生成技术获得) 与衍生测试用例 (根据蜕变关系获得), 检查对应的输出是否违反蜕变关系. 如果违反了某种蜕变关系, 则表明存在故障 (如后文图 8 所示)[15]. 蜕变关系不仅用作测试预期, 并能生成新的测试数据, 在多个领域得到成功的应用[47]. 在深度学习系统测试中, 深度学习系统本身的不确定性 (包括运行环境、输出不确定等) 使得测试数据生成以及测试预期问题更加困难. 研究者利用蜕变关系判定面向深度学习系统的模糊测试结果, 并指导原始种子进行变异[10,16,51]. Tian 等人为无人驾驶车的方向控制系统设计了如下蜕变关系[10]:

$$(\phi_t-\phi_{ti})^2 \leqslant \gamma MSG,$$

其中, ϕ_t 和 ϕ_{ti} 分别表示原始种子和原始种子对应的变异种子的输出; MSG 是均方误差; γ 是超参数. 该蜕变关系表示原始种子和基于原始种子产生的变异种子的输出误差要在一定范围内. 如果超过了这个范围, 则表明方向控制系统存在缺陷[10]. Du 等人针对语音识别系统设计了一条蜕变关系[16]: 在原始种子的基础上, 利用不改变语义的转换规则得到变异种子. 如果原始种子与变异种子的输出一致, 则没有违反蜕变关系; 否则, 违反蜕变关系. Braiek 等人针对图像识别系统设计了一条蜕变关系: 在原始种子的基础上利用不改变语义的转换规则得到变异种子, 如果原始种子与变异种子的输出一致, 则没有违反蜕变关系; 否则, 违反蜕变关系[51]. Xie 等人将原始种子与相应变异种子输出一致这种蜕变关系定义为蜕变变异[49,54]. Du 等人[16,52]将蜕变变异的思想引入深度循环神经网络的测试中, 并提出了多种种子变异策略.

3.2.2 利用差分测试/N-版本编程判定测试结果

差分测试通过观察类似的应用程序对于相同的输入是否产生不同的输出来检测缺陷[64,65]. N-版本编程基于规格说明书生成多个功能上等价的程序, 然后用同一个输入执行程序, 通过观察这些程序的行为是否一致检测程序中的缺陷[66]. 图 9 展示了差分测试/N-版本编程的测试流程. 如果存在一个程序与其他程序的输出结果不一致, 则表明至少该程序存在缺陷. Pei 等人将差分测试应用于深度学习系统的模糊测试, 将同一个种子输入到多个功能相似的模型中, 通过观察模型的行为判断是否存在缺陷[32]. 差分测试/N-版本编程在实际应用时, 面临多个因素的制约: (1) 难以找到多个功能相似的模型; (2) 编写多个程序版本的代价高昂.

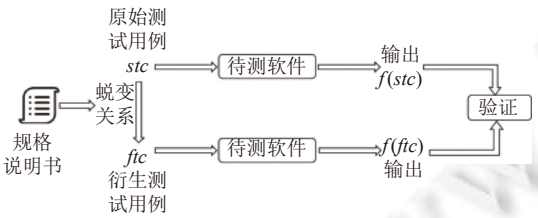

图 8 蜕变测试框架

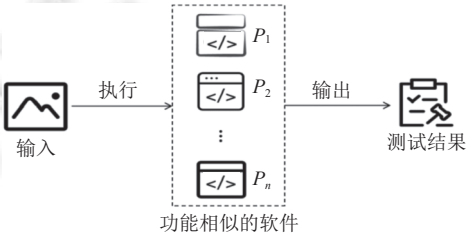

图 9 差分测试/N-版本编程测试框架

3.3 覆盖分析

覆盖分析依据测试充分性指标对变异种子的执行过程进行分析, 判断变异的种子是否应加入种子队列中. 针对深度学习系统的测试充分性指标, 依据是否考虑深度学习模型的内部元素 (例如神经元、层等), 这些充分性指

标划分可以分为: 基于结构的测试充分性指标和基于非结构的测试充分性指标. 目前, 基于覆盖的深度学习系统模糊测试技术均采用基于结构的测试充分性指标. 但是, 基于非结构的测试充分性指标也可以作为候选的覆盖分析方法. 因此本文也总结了常见的基于非结构的测试充分性指标.

3.3.1 基于结构的测试充分性指标

代码覆盖准则评估测试用例集覆盖待测软件代码的程度. 一般认为, 高覆盖率的测试, 软件中潜藏的故障则较少[67]. Pei 等人发现随机选择一个测试用例就能使深度学习系统的代码覆盖率达到 100%[32]. 由于智能软件的决策逻辑的准确性不仅受实现代码的影响, 还受到训练数据的影响, 因此传统软件测试的充分性准则不适用于深度学习系统. 在覆盖测试充分性准则 (语句覆盖、分支覆盖等) 基础上, 针对深度学习系统的特点, 人们提出了神经元覆盖 (NC)[32]、K 段神经元覆盖 (KMNC)、强覆盖 (SNAC)、边界覆盖 (NBC)、Top-k神经元覆盖 (TKNC)、Top-k 神经元模式 (TKNP)[68]、状态覆盖[16]、MC/DC 覆盖[69]和组合覆盖[70].

图 10 展示了部分基于结构的测试充分性指标[71]. Pei 等人针对深度学习系统由大量个计算单元构成这一特点, 提出了神经元覆盖准则, 量化测试用例集激活的神经元数目占所有神经元数目的比例[32].

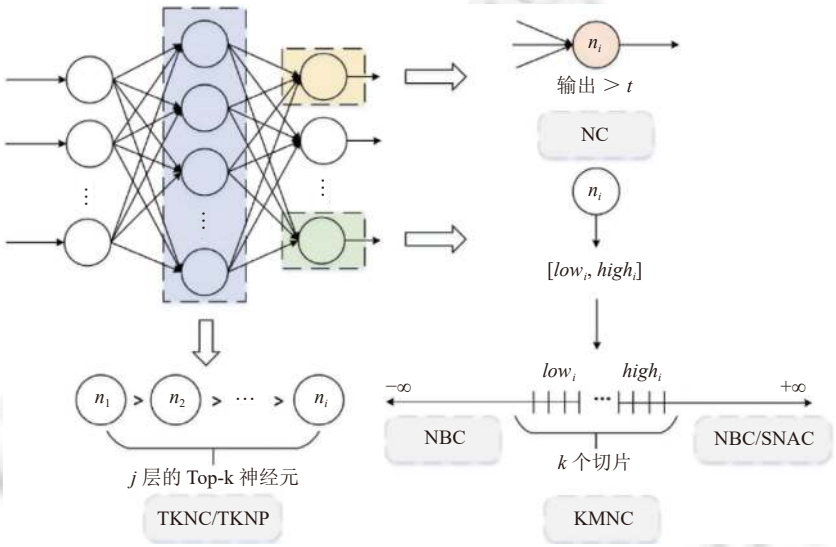

图 10 几种基于结构的测试充分性指标[71]

针对神经元覆盖没有考虑到不同神经元的输出在统计分布之间的差异性, Ma 等人[68]提出了多粒度的测试充分性指标, 包括神经元级别的测试充分性指标 (KMNC、SNAC 和 NBC) 和层级别的测试充分性指标 (TKNC 和 TKNP). Ma 等人[68]认为数据工程师采集的数据具有一定的代表性, 模型在训练的过程中能够从数据中学习到主要的功能. 由此, 将深度学习系统的行为划分为主要功能区和边角区. 如果对 $\forall n \in N$ (n 为神经元, N 为神经元集合), $\exists x \in T$ (x 为测试用例, T 为测试用例集), 使得 $\phi(x,n) \in [low_n, high_n]$ ($\phi(x,n)$ 为神经元 n 在测试用例 x 上的输出; low_n 和 $high_n$ 分别是神经元 n 在整个训练过程中输出的最小值和最大值), 则认为深度学习模型的行为处于主要功能区; 如果 $\phi(x,n) \in [-\infty, low_n]$ 或者 $\phi(x,n) \in [high_n, +\infty]$, 则认为深度学习模型的行为处于边角区域. KMNC 将 $[low_n, high_n]$ 划分为 k 个切片, 并统计在测试的过程中神经元覆盖每一个切片的情况. NBC 统计在测试的过程中输出 $\phi(x,n) \in [-\infty, low_n]$ 或者 $\phi(x,n) \in [high_n, +\infty]$ 的神经元占所有神经元数量的比例. SNAC 衡量在测试的过程中输出 $\phi(x,n) \in [high_n, +\infty]$ 的神经元占所有神经元的比例. TKNC 衡量每一层有多少个神经元曾经是最活跃的神经元. TKNP 表示每一层最活跃的神经元的不同激活情景.

Gerasimou 等人认为 KMNC 忽略了神经元从数据中学习到的语义信息, 将深度学习模型对输入做出的决策进行分解, 并利用层间的关联性传播对每层的每个神经元进行重要性分析 (识别对决策有关键贡献的神经元), 然后

利用聚类算法对重要神经元的输出值进行聚类, 将重要神经元的输出范围划分切片 (激活值簇). 在此基础上, 提出了重要性驱动的覆盖准则 IDC, 衡量测试用例集覆盖重要神经元激活值簇组合的情况[72].

Sekhon 等人发现 2 个 MNIST 数据集中的测试用例就能实现 LeNet-1 模型 100% 的神经元覆盖, 并认为神经元覆盖准则的粒度较粗且不够充分. 基于“几乎所有的故障都是由相对较少的参数之间的相互作用引起的”这一观察, 提出了 2-way 覆盖, 衡量测试用例集覆盖 ($n_{i,k-1},n_{j,k-1},n_{q,k}$) 三元组的情况, 其中 k 表示深度学习模型的第 k 层; q 表示第 k 层的第 q 个神经元[73].

针对已有的基于结构的测试充分性指标只考虑了有向图中的节点 (神经元)、没有考虑有向图中的边这一问题, Wang 等人将深度学习网络视为有向图, 首先定义了神经元状态 (分为激活状态和抑制状态) 的概念和子路径状态 (路径中神经元状态构成的元组) 的概念, 然后基于神经元状态和子路径状态提出路径覆盖, 衡量测试用例集覆盖不同路径的不同状态的程度[74].

上述的基于结构的测试充分性指标均是针对卷积深度网络, 不适用于循环神经网络. 这是由于循环神经网络中存在回路以及网络内部状态之间的转移. Du 等人提出了状态级别和转换级别两种基于抽象状态转换模型的测试覆盖准则, 以捕获其动态的状态转换行为[16].

Sun 等人认为神经元覆盖的粒度太粗, 较少的测试用例集就能实现 100% 覆盖[69,75]. 为此, Sun 等人在传统程序测试的 MC/DC 覆盖基础上等将 l 层的一个特征 $F_{i,l}$ (神经元集合) 视为判定, 将 $l-1$ 层连接 $F_{i,l}$ 的特征视为条件, 提出了信号改变 (给定两个输入 x_1 和 x_2, 如果神经元 n 的激活状态发生了改变, 就认为神经元 n 的信号发生了改变) 和值改变 (给定两个输入 x_1 和 x_2, 如果神经元 n 两次输出的值超过了规定的阈值, 就认为神经元 n 的值发生了改变) 两个新概念, 并在信号改变和值改变的基础上提出了信号-信号覆盖、值-信号覆盖、信号-值覆盖和值-值覆盖[69,75].

Ma 等人将深度学习系统的内部结构信息与组合测试结合, 提出面向深度学习模型的组合覆盖准则 DeepCT[70], 以及神经元激活配置 (NAC) 的概念. 对于给定的神经元集合 $L=\{n_1,n_2,\ldots,n_k\}$ (k 为 L 中神经元的数目), NAC 是一个 k 元组, 表示为 $NAC=\{b_1,b_2,\ldots,b_k\}$ (b_i 表示第 i ($1\leqslant i\leqslant k$) 个神经元的激活状态: 如果 n_i 被测试用例集激活, 则 $b_i=1$; 反之, $b_i=0$). 在 NAC 的基础上, 提出了 t-way 稀疏覆盖、t-way 稠密覆盖和完全覆盖[70].

在传统的软件测试中, 变异分析[76]被用来评估测试用例集揭示故障的能力. 深度学习系统的行为与学习程序 (代码)、训练数据和选择的算法模型相关, 将变异测试应用到智能软件测试中需要在代码、数据和算法模型中注入缺陷, 并依据变异的粒度 (模型、代码和数据) 将变异分析分为模型水平、代码和数据水平的变异分析[77−79].

3.3.2 基于非结构的测试充分性

对于智能软件系统而言, 一个好的测试集应该是系统多样化的, 包括从类似于训练数据的输入到明显不同和对抗的输入[80,81]. 为此, Kim 等人提出了惊讶指标[82], 用来衡量一个新的输入与系统所使用的训练数据相比的惊讶程度. 该指标需要测试人员预先设置一个惊讶上限 U, 并将区间 $[0,U]$ 划分 k 段, 每段所有可能取值的集合用 S_i ($0\leqslant i\leqslant k$) 表示, 计算每一个测试用例相较于训练集的惊讶程度, 最后统计所有测试用例覆盖 $[0,U]$ 的情况.

3.3.3 测试充分性指标的有效性研究

研究者提出了大量面向深度学习模型的测试充分性指标, 它们的有效性需要进一步地研究. Kim 等人认为基于结构的测试充分性指标过于简单且粒度太粗, 不能用来指导测试用例生成[82]; Trujillo 等人认为神经元覆盖不足以得出关于深度强化学习网络的设计或结构的实质性结论[83]. Chen 等人认为 MC/DC 覆盖在应用于大规模的数据集和模型时的时间开销太大, 难以运用[71]. Li 等人认为 DeepCT 的粒度太细, 无法让可用的自然输入达到有意义的覆盖范围[84]. Jahangirova 等人对面向深度学习系统的变异分析进行了经验研究, 实验结果表明变异分析是一个有前途的测试充分性指标, 然而已有的部分变异算子需要用训练数据重新训练模型, 导致变异分析的开销变得异常庞大[85]. Chen 等人总结了基于结构的测试充分性准则和基于非结构的测试充分性准则均基于“测试充分性准则与对抗输入生成或输入的错误揭示能力有关”这一假设[71]. 通过经验研究的方式验证上述假设是否成立. 研究结果表明, 对于一般的基于结构的测试充分性准则而言, 这一假设不成立; 但对于超过半数的非结构性覆盖而言, 这一假设仍然成立. 然而, 基于非结构的测试充分性准则缺乏参数设置指导, 导致在实际中难以运用[71].

4 评估模型和数据集

表 3 总结了深度学习系统的模糊测试相关研究的实验评估中用到的数据集和模型. 评估模型的输入可以分为图像、音频、文本、安卓 APP 和视频. 在图像数据集中, 包含的图像数量是非常巨大的, 例如 ImageNet 具有的图片数量超过了 140 万. 表 3 中, N/A 表示不确定具体模型的种类 (使用 AAVE 和 OLD 的相关研究属于黑盒模糊测试, 因此不能确定具体的模型种类); <*x*,*y*,…> 表示第 1 层包含 *x* 个神经元, 第 2 层包含 *y* 个神经元.

表 3 面向深度学习系统的模糊测试的数据集与模型

输入	数据集	模型
图像	MNIST[86]	LeNet-1/4/5[86,87]、LeNet[88]、MNIST-LSTM[89]、MNIST-GRU[90]、CNN[50]
	CIFAR-10[91]	ResNet-20/32/50[36]、VGG-16/19[3]、WideResNetw32-10[92]、20 CNN[58]、CifarNet[93]
	ImageNet[11]	VGG-16/19[3]、ResNet-50[36]、MobileNet[94]、Inception-V3[95]
	SVHN[96]	WideResNetw32-10[92]、Task3[97]
	HMB_3.bag[98]	Chauffeur[99]、Epoch[100]、Rambo[101]
	GTSRB[102]	VGG-16/19[3]、Traffic-Sign-Recognition[103]、Traffic-Sign-Classification[104]
	Fashion-MNIST[105]	Fashion-MNIST-CNN-Keras[106]、Fashion-MNIST-with-Keras[107]
音频	Librispeech[108]	DeepSpeech 0.1.1/0.3.0[109]
	Fisher[110]	DeepSpeech 0.3.0
	Switchboard[111]	DeepSpeech 0.3.0
	CVD[112]	DeepSpeech 0.3.0
文本	Contagio[113]	<200, 200>、<200, 200, 200>、<200, 200, 200, 200>[32]
	VirusTotal[114]	<200, 200>、<200, 200, 200>、<200, 200, 200, 200>[32]
	IMDB[115]	VGG-16/19
	AAVE[116]	N/A
	OLD[117]	N/A
APP	Drebin[118,119]	<200, 200>、<50, 50>、<200, 10>[120]
视频	Driving[121]	DAVE-orig/norminit/dropout[122−124]

图 11 总结了深度学习系统的模糊测试研究中的数据集使用情况. 可以看出, 图像类的数据集的使用频率最高 (特别地, MNIST、CIFAR-10 和 ImageNet 这 3 个数据集的使用频率明显高于其他数据集), 音频次之, 文本、安卓 APP 和视频类的数据集使用的最少.

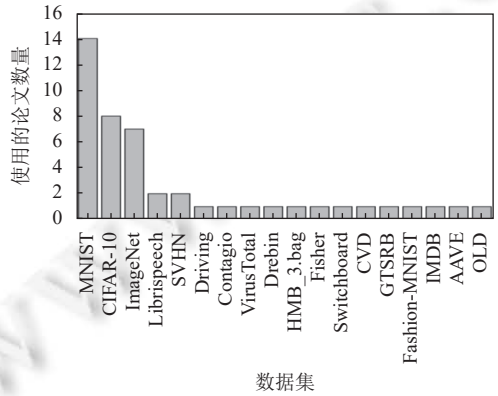

图 11 不同数据集的使用情况

5 评估标准

度量标准可以从不同的角度量化模糊测试技术的表现, 是检验测试技术效率和有效性的重要手段. 表 4 总结了深度学习系统的模糊测试相关研究的度量指标. 其中 n 表示神经元, N 表示神经元集合, l 表示神经网络的层数, x 表示测试用例, T 表示测试用例集, $\phi(x,n)$ 表示神经元 n 在 x 上的输出, t 表示阈值; $[low_n, high_n]$ 表示神经元 n 在测试用例集 T 上的输出范围 S_i^n 是 $[low_n, high_n]$ 上的第 i 个切片, $top_k(x,i)$ 表示 x 激活的第 i 层最活跃的 k 个神经元的集合, UCN 是输出值 $\phi(x,n)\in[high_n,+\infty]$ 的神经元集合, LCN 是输出值 $\phi(x,n)\in[-\infty,low_n]$ 的神经元集合.

表 4 度量标准使用情况

度量指标	形式化描述	物理含义
代码覆盖率[32]	$ELOC/LOC$	测试用例覆盖的代码行数($ELOC$)与总代码行数(LOC)的比值
神经元覆盖率[32]	$\frac{\lvert\{n\lvert x\in T,\phi(x,n>t)\}\rvert}{\lvert N\rvert}$	测试用例集激活的神经元数目占所有神经元的比例
k段神经元覆盖率[49]	$\frac{\sum_{n\in N}\{S_i^n\lvert x\in T:\phi(x,n)\in S_i^n\}}{K\times\lvert N\rvert}$	测试用例集覆盖每一个神经元的切片数与切片总数的比值
神经元边界覆盖率[49]	$\frac{\lvert UCN\rvert+\lvert LCN\rvert}{\lvert N\rvert\times 2}$	度量处于边角区域的神经元数目占所有神经元数目的比例
强神经元激活覆盖率[49]	$\frac{\lvert UCN\rvert}{\lvert N\rvert}$	度量 UCN 中的神经元数目占所有神经元数目的比例
Top-k神经元覆盖率[49]	$\frac{\lvert\bigcup_{x\in T}(\bigcup_{1\leqslant i\leqslant l}top_k(x,i))\rvert}{\lvert N\rvert}$	度量最活跃的神经元激活数目占所有神经元数目的比例
对抗输入数量[18]	N/A	产生的对抗输入数量
产生一个对抗输入的时间[18]	N/A	生成一个对抗输入的时间
模型精确度[32]	N/A	模型在测试用例集上预测的准确度
无效变异率[49]	$\frac{M_i}{M_{\text{all}}}$	无效变异种子占所有变异种子的比例, 其中 M_i 表示无效的变异种子集合; M_{all} 表示生成的所有变异种子
成功率[55]	$\frac{\sum_{i=1,\ldots,k}findAdv(x_i)}{k}$	对抗输入占所有输入的比例, 其中 $findAdv(x_i)$ 表示 x_i 是对抗输入
平均失真率[55]	$\frac{\sum_{i=1,\ldots,k}\frac{\lVert x-x^*\rVert_{L_\infty}}{\lVert x\rVert_{L_\infty}}}{k}$	原始输入 x 与对抗输入 x^* 差异的平均值
平均查询[55]	N/A	检测一定数目的故障, 需要执行的种子数目
鲁棒精度[60]	N/A	不随任意扰动改变模型输出的测试用例占所有测试用例的比值

人们从不同的角度提出了深度学习系统的模糊测试度量标准. 这些度量标准以量化的方式评估了待测模型的质量和模糊测试技术的性能. 具体说来, 有以下几方面.

(1) 待测模型鲁棒性方面: 鲁棒精度可以直观、有效地展现待测模型的抗干扰能力. 当模糊测试被用来增强训练数据集时, 鲁棒精度可以量化训练后模型的抗干扰能力, 进而说明模糊测试增强训练数据集的有效性.

(2) 待测模型精度方面: 模型精确度直观地量化了训练后模型在测试集上的表现. 部分研究工作将对抗输入作为新的训练数据重新训练模型, 通过对比重新训练前后的模型精确度, 表明模糊测试技术有助于提高待测模型的质量.

(3) 产生对抗输入方面: 对抗输入数量、成功率、平均查询率、产生一个对抗输入的时间、无效变异率与平均失真率这 6 个度量指标都与对抗输入有关, 但是侧重点不同: ① 对抗输入数量与成功率反映了模糊测试技术揭

示故障的有效性; ② 平均查询率与产生一个对抗输入的时间反映了模糊测试技术揭示故障的效率; ③ 无效变异率与平均失真率分别反映了变异策略的有效性与变异后的数据与原始数据的差异程度. 度量指标选择需要考虑测试场景和研究目标. 如果研究目的旨在提高模糊测试的有效性, 对抗输入数量与成功率应该被选择作为度量指标; 如果研究目的旨在提高模糊测试的效率, 平均查询率与产生一个对抗输入的时间应该被选择作为度量指标; 当提出新的变异策略时, 无效变异率可以衡量变异策略产生有效变异数据的能力, 平均差异率可以衡量变异数据与原始数据的差异 (差异较小表明变异数据有较大的概率与原始数据保持语义一致).

(4) 待测模型测试程度方面: 测试充分性指标量化了模糊测试技术探索待测模型内部状态的程度, 是重要的量化模糊测试性能的指标, 被广泛用来衡量优化的模糊测试的有效性. 代码覆盖率、神经元覆盖率、强神经元覆盖率、神经元边界覆盖率、k 段神经元覆盖率和 Top-k 神经元覆盖率是常用的测试充分性指标. 其中, 神经元覆盖率是使用最广泛的测试充分性指标 (75% 的研究将神经元覆盖作为度量指标). 上述指标的可用性和有效性相关的讨论详见第 3.3 节.

6 总结与展望

深度学习系统的模糊测试是一个新兴的研究方向. 自 2017 年以来, 研究人员提出了一系列的测试技术, 开发了大量的自动化测试工具. 本文从模糊测试的种子队列构建、种子选择、种子变异、测试结果判定、覆盖分析、模型和数据集以及度量指标等方面对这一新兴领域进行了分析与总结. 表 5 进一步归纳了种子队列构建、种子变异、测试结果判定和覆盖分析等方面的研究现状. 我们认为深度学习系统的模糊测试研究尚处于早期阶段, 在工业界大规模推广应用还有很长的路. 为此, 我们进一步提出了该领域的主要挑战, 期望更多的研究工作围绕这些挑战展开, 提高深度学习系统的模糊测试技术的故障揭示效率与有效性.

表 5 面向深度学习的模糊测试技术概述

模糊测试技术	种子队列构建	种子选择	种子变异	测试结果判定	覆盖分析
DeepXplore[32]	随机选择	N/A	利用梯度值	差分测试/N-版本编程	神经元覆盖
DLFuzz[18]	随机选择	N/A	利用梯度值	输出一致性	神经元覆盖
DeepTest[10]	随机选择	N/A	基于领域知识	蜕变关系	神经元覆盖
DeepCruiser[16]	随机选择	N/A	基于领域知识	蜕变转换	基本状态、k步状态边界覆盖等
TensorFuzz[50]	随机选择	新近感知	基于领域知识	输出一致性	神经元激活向量
DeepHunter[49]	随机选择	频率感知/随机选择	基于领域知识	蜕变变异	神经元、k段神经元覆盖等
CAGTest[46]/CAGFuzz[1]	N/A	新近感知	利用生成对抗网络	输出一致性	神经元覆盖
DeepEvaluation[51]	随机选择	N/A	基于搜索方法	蜕变转换	神经元覆盖
DeepSmartFuzz[58]	N/A	随机选择	基于搜索方法	输出一致性	神经元覆盖、k段神经元覆盖
DeepStellar[52]	随机选择	N/A	基于领域知识	蜕变变异	基本状态、k步状态边界覆盖等
FDFuzz[59]	N/A	N/A	改变关键像素点	输出一致性	神经元覆盖
EFuzz[53]	随机选择	N/A	基于领域知识	输出一致性	神经元覆盖、k段神经元覆盖
DeepSearch[55]	随机选择	N/A	改变关键像素点	输出一致性	神经元覆盖
ADAPT[56]	随机选择	N/A	利用梯度值	输出一致性	神经元覆盖、Top-k神经元覆盖
SENSI[60]	N/A	N/A	基于搜索方法	输出一致性	神经元覆盖
FuzzE[61]	N/A	N/A	基于领域知识	输出一致性	与语言相关的标准
advFuzz[57]	随机选择	N/A	改变关键像素点	输出一致性	N/A

(1) 在构建多样性种子队列方面. 56% 的工作采用随机的方式构建种子队列, 44% 的工作没有报告构建种子队列的方式. 种子队列作为模糊测试的初始测试用例集, 对检测深度学习系统的故障具有决定性的影响. 传统的测试用例生成方法中, 通常利用距离的思想构建多样性的测试用例集. 针对深度学习系统复杂的输入格式, 如何构造具

有多样性的种子队列, 从而提高模糊测试探索深度学习系统内部状态的能力, 进一步保障深度学习系统的质量, 是一项尚未很好解决的难题.

(2) 在种子选择方面. 大多数研究者采用随机的方法 (顺序选择可以视为一种随机选择方式) 从种子队列中选择原始种子. 然而, 随机的方法难以识别不同种子揭示深度学习系统缺陷的概率. 在测试资源有限的情况下, 尽早揭示深度学习系统的缺陷, 是软件测试的主要目标之一. 如何尽早执行具有较高概率揭示故障的种子, 尽可能多的揭示深度学习系统的缺陷, 是一个值得研究的重要问题.

(3) 在种子变异方面. 人们已经利用领域知识、目标函数的梯度值、搜索算法等方法, 探索种子变异问题. 然而, 这些种子变异方法容易产生无效的变异种子、且依赖人工来检测无效的变异种子[52], 增加了测试资源的开销. 如何进一步减少无效的变异种子数目或者自动检测无效的变异种子是一个亟待解决的关键问题. 生成对抗网络能够产生与原始数据相似但不完全相同的数据, 并且可以利用神经网络提取的输入特征, 对比产生的数据与原始数据的差异, 避免产生无效数据. 因此, 利用生成对抗网络指导种子变异是一个可行的研究方向.

(4) 在测试结果判定方面. 61% 的研究工作将“输出一致性”作为测试预期, 所谓“输出一致性”是指在原始种子上做人眼无法识别的改变, 那么变异后的种子与原始种子的执行结果一致; 5% 的工作将差分测试/N-版本编程作为测试预期, 然而差分测试/N-版本编程存在开销大、难以找到功能相似的模型的缺点, 导致该方法在实际中难以应用. 34% 的研究工作利用蜕变关系/蜕变转换/蜕变变异判定测试结果并指导种子变异. 与此同时, 已有的蜕变关系获取方法均是依据领域知识, 泛化能力差. 由于深度神经网络本质是一个复杂的数学模型, 蕴含丰富的数学性质, 如何基于模型固有性质识别蜕变关系、增强蜕变关系的多样性与泛化性, 是一个非常有前景的研究方向.

(5) 在测试充分性指标与覆盖分析方面. 覆盖分析是面向深度学习系统模糊测试技术的关键步骤之一. 测试充分性指标可以定量地评估待测模型的测试充分程度, 有助于指导变异种子的执行过程. 现有研究工作较为深入地讨论了覆盖分析与测试充分性指标, 仍然存在如下问题: ① 覆盖信息没有得到充分利用. 覆盖信息反映了当前种子、使用的变异策略以及变异种子的好坏, 具有重要的利用价值. 现有工作中, 覆盖信息仅用来更新种子队列 (即如果某个变异种子触发了新覆盖, 则将其加入种子队列中), 而没有用来指导种子和变异策略选择; ② 覆盖分析采用的测试充分性指标较为单一. 覆盖分析试图回答不同的测试充分性准则对模糊测试效率的影响. 现有工作主要采用基于结构的测试充分性指标 (例如神经元覆盖、k 段神经元覆盖等) 作为覆盖分析的手段, 尚未考虑非结构的测试充分性指标; ③ 测试充分性指标的有效性有待商榷: 如何定义统一、有效的测试充分性指标集是一个亟待解决的重要问题. 人们对现有的测试充分性指标的有效性尚未达成共识. 特别地, 经验研究结果并不表明神经元覆盖、惊讶指标和变异分析等测试充分性指标的有效性.

References:

[1] Zhang PC, Dai QY, Pelliccione P. CAGFuzz: Coverage-guided adversarial generative fuzzing testing for image-based deep learning systems. IEEE Trans. on Software Engineering, 2021: 1 [doi: 10.1109/TSE.2021.3124006]

[2] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Proc. of the 27th Int'l Conf. on Neural Information Processing Systems. Cambridge: MIT Press, 2014. 3104–3112.

[3] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego: ICLR, 2014.

[4] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587): 484–489. [doi: 10.1038/nature16961]

[5] Sermanet P, LeCun Y. Traffic sign recognition with multi-scale convolutional networks. In: Proc. of the 2011 Int'l Joint Conf. on Neural Networks. San Jose: IEEE, 2011. 2809–2813. [doi: 10.1109/IJCNN.2011.6033589]

[6] 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]

[7] 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. Sacramento: IEEE, 2016. 1–10. [doi: 10.1109/DASC.2016.7778091]

[8] 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]

[9] Bertolino A. Software testing research: Achievements, challenges, dreams. In: Proc. of the 2007 Future of Software Engineering (FOSE 2007). Minneapolis: IEEE, 2007. 85–103. [doi: 10.1109/FOSE.2007.25]

[10] Tian YC, Pei KX, Jana S, Ray B. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proc. of the 40th Int'l Conf. on Software Engineering. Gothenburg: ACM, 2018. 303–314. [doi: 10.1145/3180155.3180220]

[11] Deng J, Dong W, Socher R, Li LJ, Li K, Li FF. ImageNet: A large-scale hierarchical image database. In: Proc. of the 2009 Conf. on Computer Vision and Pattern Recognition. Miami: IEEE, 2009. 248–255. [doi: 10.1109/CVPR.2009.5206848]

[12] Merkel R. Software reliability growth models predict autonomous vehicle disengagement event. arXiv:1812.08901, 2018.

[13] Zhang JM, Harman M, Ma L, Liu Y. Machine learning testing: Survey, landscapes and horizons. IEEE Trans. on Software Engineering, 2022, 48(1): 1–36. [doi: 10.1109/TSE.2019.2962027]

[14] Gopinath D, Wang KY, Zhang MS, Pasareanu CS, Khurshid S. Symbolic execution for deep neural networks. arXiv:1807.10439, 2018.

[15] Chen TY, Cheung SC, Yiu SM. Metamorphic testing: A new approach for generating next test cases. arXiv:2002.12543, 2020.

[16] Du XN, Xie XF, Li Y, Ma L, Zhao JJ, Liu Y. DeepCruiser: Automated guided testing for stateful deep learning systems. arXiv:1812.05339, 2018.

[17] Miller BP, Fredriksen L, So B. An empirical study of the reliability of UNIX utilities. Communications of the ACM, 1990, 33(12): 32–44. [doi: 10.1145/96267.96279]

[18] Guo J, Jiang Y, Zhao Y, Chen Q, Sun JG. DLFuzz: Differential fuzzing testing of deep learning systems. In: Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Lake Buena: ACM, 2018. 739–743. [doi: 10.1145/3236024.3264835]

[19] Liang HL, Pei XX, Jia XD, Shen WW, Zhang J. Fuzzing: State of the art. IEEE Trans. on Reliability, 2018, 67(3): 1199–1218. [doi: 10.1109/TR.2018.2834476]

[20] Manès VJM, Han HS, Han C, Cha SK, Egele M, Schwartz EJ, Woo M. The art, science, and engineering of fuzzing: A survey. IEEE Trans. on Software Engineering, 2021, 47(11): 2312–2331. [doi: 10.1109/TSE.2019.2946563]

[21] Brereton F. Binspector: Evolving a security tool. 2014. http://binspector.github.io/

[22] Cisco secure development lifecycle. 2021. https://www.cisco.com/c/en/us/about/security-center/security-programs/secure-development-lifecycle/sdl-process/validate.html

[23] Google chromium security. 2021. https://www.chromium.org/Home/chromium-security/bugs

[24] Announcing OSS-Fuzz: Continuous fuzzing for open source software. 2016. https://opensource.googleblog.com/2016/12/announcing-oss-fuzz-continuous-fuzzing.html

[25] Clusterfuzz. https://code.google.com/p/clusterfuzz/

[26] Microsoft Security Development Lifecycle, Verification Phase. https://docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307418(v=msdn.10)

[27] Bounimova E, Godefroid P, Molnar D. Billions and billions of constraints: Whitebox fuzz testing in production. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). San Francisco: IEEE, 2013. 122–131. [doi: 10.1109/ICSE.2013.6606558]

[28] Howard M, Lipner S. The Security Development Lifecycle. Redmond: Microsoft Press, 2006.

[29] 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]

[30] Tahir Z, Alexander R. Coverage based testing for V&V and safety assurance of self-driving autonomous vehicles: A systematic literature review. In: Proc. of the 2020 IEEE Int'l Conf. on Artificial Intelligence Testing (AITest). Oxford: IEEE, 2020. 23–30. [doi: 10.1109/AITEST49225.2020.00011]

[31] Riccio V, Jahangirova G, Stocco A, Humbatova N, Weiss M, Tonella P. Testing machine learning based systems: A systematic mapping. Empirical Software Engineering, 2020, 25(6): 5193–5254. [doi: 10.1007/s10664-020-09881-0]

[32] Pei KX, Cao YZ, Yang JF, Jana S. DeepXplore: Automated whitebox testing of deep learning systems. In: Proc. of the 26th Symp. on Operating Systems Principles. Shanghai: ACM, 2017. 1–18. [doi: 10.1145/3132747.3132785]

[33] Jouppi NP, Young C, Patil N, *et al*. In-datacenter performance analysis of a tensor processing unit. In: Proc. of the 44th ACM/IEEE Annual Int'l Symp. on Computer Architecture (ISCA). Toronto: IEEE, 2017. 1–12. [doi: 10.1145/3079856.3080246]

[34] Lindholm E, Nickolls J, Oberman S, Montrym J. NVIDIA Tesla: A unified graphics and computing architecture. IEEE Micro, 2008,

28(2): 39–55. [doi: 10.1109/MM.2008.31]

[35] Luebke D. CUDA: Scalable parallel programming for high-performance scientific computing. In: Proc. of the 5th IEEE Int'l Symp. on Biomedical Imaging: From Nano to Macro. Paris: IEEE, 2008. 836–838. [doi: 10.1109/ISBI.2008.4541126]

[36] He KM, Zhang XY, Ren SQ, Sun J. Deep residual learning for image recognition. In: Proc. of the 25th IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016. 770–778. [doi: 10.1109/CVPR.2016.90]

[37] 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]

[38] Szegedy C, Liu W, Jia YQ, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proc. of the 2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Boston: IEEE, 2015. 1–9. [doi: 10.1109/CVPR.2015.7298594]

[39] Krizhevsky A. Learning multiple layers of features from tiny images [MS. Thesis]. Toronto: University of Toronto, 2009.

[40] Miller GA. WordNet: A lexical database for English. Communications of the ACM, 1995, 38(11): 39–41. [doi: 10.1145/219717.219748]

[41] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: Proc. of the 27th Int'l Conf. on Machine Learning. Haifa: Omnipress, 2010. 807–814.

[42] Yosinski J, Clune J, Nguyen A, Fuchs T, Lipson H. Understanding neural networks through deep visualization. arXiv:1506.06579, 2015.

[43] Monniaux D. A survey of satisfiability modulo theory. In: Proc. of the 18th Int'l Workshop on Computer Algebra in Scientific Computing. Bucharest: Springer, 2016. 401–425. [doi: 10.1007/978-3-319-45641-6_26]

[44] Oehlert P. Violating assumptions with fuzzing. IEEE Security & Privacy, 2005, 3(2): 58–62. [doi: 10.1109/MSP.2005.55]

[45] Woo M, Cha SK, Gottlieb S, Brumley D. Scheduling black-box mutational fuzzing. In: Proc. of the 20th IGSAC Conf. on Computer & Communications Security. Berlin: ACM, 2013. 511–522. [doi: 10.1145/2508859.2516736]

[46] Zhang PC, Dai QY, Ji SH. Condition-guided adversarial generative testing for deep learning systems. In: Proc. of the 2019 IEEE Int'l Conf. on Artificial Intelligence Testing (AITest). Newark: IEEE, 2019. 71–72. [doi: 10.1109/AITest.2019.000-5]

[47] Segura S, Fraser G, Sanchez AB, Ruiz-Cortés A. A survey on metamorphic testing. IEEE Trans. on Software Engineering, 2016, 42(9): 805–824. [doi: 10.1109/TSE.2016.2532875]

[48] Chen TY, Kuo FC, Liu H, Poon PL, Towey D, Tse TH, Zhou ZQ. Metamorphic testing: A review of challenges and opportunities. ACM Computing Surveys, 2018, 51(1): 4. [doi: 10.1145/3143561]

[49] Xie XF, Ma L, Xu FJ, 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. Beijing: ACM, 2019. 146–157. [doi: 10.1145/3293882.3330579]

[50] Odena A, Olsson C, Andersen D, Goodfellow I. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: ICML, 2019. 4901–4911.

[51] Braiek HB, Khomh F. DeepEvolution: A search-based testing approach for deep neural networks. In: Proc. of the 35th IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Cleveland: IEEE, 2019. 454–458. [doi: 10.1109/ICSME.2019.00078]

[52] Du XN, Xie XF, Li Y, Ma L, Liu Y, Zhao JJ. DeepStellar: Model-based quantitative analysis of stateful deep learning systems. In: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Tallinn: ACM, 2019. 477–487. [doi: 10.1145/3338906.3338954]

[53] Park LH, Oh S, Kim J, Chung S, Kwon T. Poster: Effective layers in coverage metrics for deep neural networks. In: Proc. of the 27th ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 2681–2683. [doi: 10.1145/3319535.3363286]

[54] Xie XF, Chen HX, Li Y, Ma L, Liu Y, Zhao JJ. Coverage-guided fuzzing for feedforward neural networks. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 1162–1165. [doi: 10.1109/ASE.2019.00127]

[55] Zhang F, Chowdhury SP, Christakis M. DeepSearch: A simple and effective blackbox attack for deep neural networks. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and the Symp. on the Foundations of Software Engineering. ACM, 2020. 800–812. [doi: 10.1145/3368089.3409750]

[56] Lee S, Cha S, Lee D, Oh H. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In: Proc. of the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2020. 165–176. [doi: 10.1145/3395363.3397346]

[57] Qin Y, Yue C. Fuzzing-based hard-label black-box attacks against machine learning models. Computers & Security, 2022, 117: 102694. [doi: 10.1016/j.cose.2022.102694]

[58] Demir S, Eniser HF, Sen A. DeepSmartFuzzer: Reward guided test generation for deep learning. In: Proc. of the 2020 Workshop on Artificial Intelligence Safety Co-located with the 29th Int'l Joint Conf. on Artificial Intelligence and the 17th Pacific Rim Int'l Conf. on

Artificial Intelligence. Yokohama: CEUR-WS.org, 2021. 1–7.

[59] Wang J, Cao KF, Fang CR, Chen JX. FDFuzz: Applying feature detection to fuzz deep learning systems. Int'l Journal of Performability Engineering, 2019, 15(10): 2675. [doi: 10.23940/ijpe.19.10.p13.26752682]

[60] Gao X, Saha RK, Prasad MR, Roychoudhury A. Fuzz testing based data augmentation to improve robustness of deep neural networks. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Seoul: IEEE, 2020. 1147–1158.

[61] Rios A. FuzzE: Fuzzy fairness evaluation of offensive language classifiers on African-American English. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence. Palo Alto: AAAI Press, 2020. 881–889.

[62] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proc. of the 16th IEEE Int'l Conf. on Computer Vision (ICCV). Venice: IEEE, 2017. 2223–2232. [doi: 10.1109/ICCV.2017.244]

[63] Lowe DG. Distinctive image features from scale-invariant keypoints. Int'l Journal of Computer Vision, 2004, 60(2): 91–110. [doi: 10.1023/B:VISI.0000029664.99615.94]

[64] Davis MD, Weyuker EJ. Pseudo-oracles for non-testable programs. In: Proc. of the 1981 ACM Conf. New York: ACM, 1981. 254–257. [doi: 10.1145/800175.809889]

[65] McKeeman WM. Differential testing for software. Digital Technical Journal, 1998, 10(1): 100–107.

[66] Avizienis A. The methodology of N-version programming. In: Lyu M, ed. Software Fault Tolerance. New York: Wiley, 1995. 23–46.

[67] Zhang J, Zhang LM, Harman M, Hao D, Jia Y, Zhang L. Predictive mutation testing. IEEE Trans. on Software Engineering, 2018, 45(9): 898–918. [doi: 10.1109/TSE.2018.2809496]

[68] Ma L, Xu FJ, Zhang FY, Sun JY, Xue MH, Li B, Chen CY, Su T, Li L, Liu Y, Zhao JJ, Wang YD. DeepGauge: Multi-granularity testing criteria for deep learning systems. In: Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Montpellier: IEEE, 2018. 120–131. [doi: 10.1145/3238147.3238202]

[69] Sun YC, Huang XW, Kroening D, Sharp J, Hill M, Ashmore R. Structural test coverage criteria for deep neural networks. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE-Companion). Montreal: IEEE, 2019. 320–321. [doi: 10.1109/ICSE-Companion.2019.00134]

[70] Ma L, Zhang FJ, Xue MH, Li B, Liu Y, Zhao JJ, Wang YD. DeepCT: Combinatorial testing for deep learning systems. In: Proc. of the 26th IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering. Hangzhou: IEEE, 2019. 614–618. [doi: 10.1109/SANER.2019.8668044]

[71] Chen JJ, Yan M, Wang Z, Kang YN, Wu Z. Deep neural network test coverage: How far are we? arXiv:2010.04946, 2020.

[72] Gerasimou S, Eniser HF, Sen A, Cakan A. Importance-driven deep learning system testing. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Seoul: IEEE, 2020. 702–713.

[73] Sekhon J, Fleming C. Towards improved testing for deep learning. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE-NIER). Montreal: IEEE, 2019. 85–88. [doi: 10.1109/ICSE-NIER.2019.00030]

[74] Wang D, Wang ZY, Fang CR, Chen YS, Chen ZY. DeepPath: Path-driven testing criteria for deep neural networks. In: Proc. of the 2019 IEEE Int'l Conf. on Artificial Intelligence Testing (AITest). Newark: IEEE, 2019. 119–120. [doi: 10.1109/AITest.2019.00013]

[75] Sun YC, Huang XW, Kroening D, Sharp J, Hill M, Ashmore R. Testing deep neural networks. arXiv:1803.04792, 2018.

[76] Papadakis M, Kintis M, Zhang J, Jia Y, Le Traon Y, Harman M. Mutation testing advances: An analysis and survey. Advances in Computers, 2019, 112: 275–378. [doi: 10.1016/bs.adcom.2018.03.015]

[77] Ma L, Zhang FY, Sun JY, Xue MH, Li B, Xu JF F, Xie C, Li L, Liu Y, Zhao JJ, Wang YD. DeepMutation: Mutation testing of deep learning systems. In: Proc. of the 29th IEEE Int'l Symp. on Software Reliability Engineering (ISSRE). Memphis: IEEE, 2018. 100–111. [doi: 10.1109/ISSRE.2018.00021]

[78] Hu Q, Ma L, Xie XF, Yu B, Liu Y, Zhao JJ. DeepMutation++: A mutation testing framework for deep learning systems. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 1158–1161. [doi: 10.1109/ASE.2019.00126]

[79] Shen WJ, Wan J, Chen ZY. MuNN: Mutation analysis of neural networks. In: Proc. of the 18th IEEE Int'l Conf. on Software Quality, Reliability and Security Companion (QRS-C). Lisbon: IEEE, 2018. 108–115. [doi: 10.1109/QRS-C.2018.00032]

[80] Feldt R, Poulding S, Clark D, Yoo S. Test set diameter: Quantifying the diversity of sets of test cases. In: Proc. of the 2016 IEEE Int'l Conf. on Software Testing, Verification and Validation (ICST). Chicago: IEEE, 2016. 223–233. [doi: 10.1109/ICST.2016.33]

[81] Poulding S, Feldt R. Generating controllably invalid and a typical inputs for robustness testing. In: Proc. of the 10th IEEE Int'l Conf. on Software Testing, Verification and Validation Workshops (ICSTW). Tokyo: IEEE, 2017. 81–84. [doi: 10.1109/ICSTW.2017.21]

[82] Kim J, Feldt R, Yoo S. Guiding deep learning system testing using surprise adequacy. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Montreal: IEEE, 2019. 1039–1049. [doi: 10.1109/ICSE.2019.00108]

[83] Trujillo M, Linares-Vásquez M, Escobar-Velásquez C, Dusparic I, Cardozo N. Does neuron coverage matter for deep reinforcement learning? A preliminary study. In: Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering Workshops. Seoul: ACM, 2020. 215–220. [doi: 10.1145/3387940.3391462]

[84] Li ZN, Ma XX, Chang X, Cao C. Structural coverage criteria for neural networks could be misleading. In: Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering (ICSE-NIER). Montreal: IEEE, 2019. 89–92. [doi: 10.1109/ICSE-NIER.2019.00031]

[85] Jahangirova G, Tonella P. An empirical evaluation of mutation operators for deep learning systems. In: Proc. of the 13th IEEE Int'l Conf. on Software Testing, Validation and Verification (ICST). Porto: IEEE, 2020. 74–84. [doi: 10.1109/ICST46399.2020.00018]

[86] LeCun Y, Cortes C, Burges C. MNIST handwritten digit database. AT&T Labs. 2010. http://yann.lecun.com/exdb/mnist

[87] 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]

[88] Variant of lenet. 2022. https://github.com/tensorflow/models/blob/master/research/slim/nets/lenet.py

[89] 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]

[90] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. In: Proc. of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8). Doha: Association for Computational Linguistics, 2014. 103–111. [doi: 10.3115/v1/W14-4012]

[91] Krizhevsky A. The CIFAR-10 dataset. 2022. http://www.cs.toronto.edu/~kriz/cifar.html

[92] Zagoruyko S, Komodakis N. Wide residual networks. arXiv:1605.07146, 2016.

[93] CIFARNet. 2022. https://github.com/tensorflow/models/blob/master/research/slim/nets/cifarnet.py

[94] Xiao CW, Li B, Zhu JY, He W, Liu MY, Song D. Generating adversarial examples with adversarial networks. In: Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence. Stockholm: IJCAI.org, 2018. 3905–3911.

[95] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proc. of the 25th IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE, 2016. 2818–2826. [doi: 10.1109/CVPR.2016.308]

[96] Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning. In: Proc. of the 2011 Deep Learning and Unsupervised Feature Learning Workshop, Co-located with the 25th Annual Conf. on Neural Information Processing Systems (NIPS), 2011. 23–32.

[97] Task3. 2022. https://github.com/yh1008/deepLearning

[98] Udacity self-driving car challenge dataset. 2022. https://github.com/udacity/self-driving-car/tree/master/datasets/CH2

[99] Chauffeur model. 2022. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur

[100] Epoch model. 2022. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/cg23

[101] Rambo model. 2022. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo

[102] Houben S, Stallkamp J, Salmen J, Schlipsing M, Igel C. Detection of traffic signs in real-world images: The German traffic sign detection benchmark. In: Proc. of the 23rd Int'l Joint Conf. on Neural Networks (IJCNN). Dallas: IEEE, 2013. 1–8. [doi: 10.1109/IJCNN.2013.6706807]

[103] Traffic-Sign-Recognition. 2022. https://github.com/chsasank/Traffic-Sign-Classification.keras

[104] Traffic-Sign-Classification. 2022. https://github.com/xitizzz/Traffic-Sign-Recognition-using-Deep-Neural-Network

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

[106] Fashion-MNIST-CNN-Keras. 2022. https://github.com/umbertogriffo/Fashion-mnistcnn-keras

[107] Fashion-MNIST-with-Keras. 2022. https://github.com/markjay4k/Fashion-MNIST-with-Keras

[108] Panayotov V, Chen GG, Povey D, Khundanpur S. LibriSpeech: An ASR corpus based on public domain audio books. In: Proc. of the 40th IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). Brisbane: IEEE, 2015. 5206–5210. [doi: 10.1109/ICASSP.2015.7178964]

[109] Mozilla/DeepSpeech. 2022. https://github.com/mozilla/DeepSpeech

[110] Cieri C, Miller D, Walker K. The fisher corpus: A resource for the next generations of speech-to-text. In: Proc. of the 4th Int'l Conf. on Language Resources and Evaluation (LREC 2004). Lisbon: European Language Resources Association (ELRA), 2004. 69–71.

[111] Godfrey JJ, Holliman EC, McDaniel J. SWITCHBOARD: Telephone speech corpus for research and development. In: Proc. of the 17th IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing. San Francisco: IEEE, 1992. 517–520. [doi: 10.1109/ICASSP.1992.225858]

[112] CommomVoice dataset. 2022. https://voice.mozilla.org/en/datasets
[113] PDF malware dump. 2022. http://contagiodump.blogspot.de/2010/08/malicious-documents-archive-for.html
[114] Virustotal. 2022. https://www.virustotal.com/
[115] IMDB. 2022. https://www.imdb.com/interfaces/
[116] Blodgett SL, Green L, O'Connor B. Demographic dialectal variation in social media: A case study of African-American English. In: Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing. Austin: Association for Computational Linguistics, 2016. 1119–1130. [doi: 10.18653/v1/D16-1120]
[117] Zampieri M, Malmasi S, Nakov P, Rosenthal S, Farra N, Kumar R. Predicting the type and target of offensive posts in social media. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers). Minneapolis: Association for Computational Linguistics, 2019. 1415–1420. [doi: 10.18653/v1/N19-1144]
[118] Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K. DREBIN: Effective and explainable detection of Android malware in your pocket. In: Proc. of the 21st Annual Network and Distributed System Security Symp. San Diego: The Internet Society, 2014. 23–26.
[119] Spreitzenbarth M, Freiling F, Echtler F, Schreck T, Hoffmann J. Mobile-sandbox: Having a deeper look into Android applications. In: Proc. of the 28th Annual ACM Symp. on Applied Computing. Coimbra: ACM, 2013. 1808–1815. [doi: 10.1145/2480362.2480701]
[120] Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P. Adversarial perturbations against deep neural networks for malware classification. arXiv:1606.04435, 2016.
[121] Self-driving-car. 2022. https://github.com/udacity/self-driving-car
[122] Nvidia-Autopilot-Keras. 2022. https://github.com/0bserver07/Nvidia-Autopilot-Keras
[123] Keras-steering-angle-visualizations. 2022. https://github.com/jacobgil/keras-steering-angle-visualizations
[124] Behavioral-cloning. 2022. https://github.com/navoshta/behavioral-cloning

代贺鹏(1993－), 男, 博士生, 主要研究领域为软件测试.

金慧(1997－), 女, 硕士生, 主要研究领域为软件测试.

孙昌爱(1974－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件测试, 故障定位, 服务计算.

肖明俊(1997－), 男, 硕士生, 主要研究领域为软件测试.