智能软件可靠性的研究进展与趋势

CCF 软件工程专业委员会

陈雨亭[1]　谢晓园[2]　周　宇[3]　马　雷[4]

[1]上海交通大学，上海

[2]武汉大学，武汉

[3]南京航空航天大学，南京

[4]日本九州大学，福冈

摘　要

人工智能正在彻底改变人类的工作、学习、生活、发现和沟通的方式。智能软件是人工智能的重要载体，在不断为人工智能定义新的功能、效能与边界。但是，我们仍然频繁经历智能软件的缺陷和失败。一个重要的原因是人工智能和机器学习引起的软件开发模式的转变——使用机器学习和人工智能技术，大量规则会从训练数据中推理出来。编程范型、开发模式的转变使得对具有人工智能的软件的行为进行推理变得困难，对智能软件进行分析、理解、测试或验证也存在困难。

鉴于智能软件在我们的社会中发挥着越来越重要的作用，软件工程和人工智能领域都有必要研究和开发新技术以应对其可靠性的挑战——智能软件中智能算法的行为可能不正确，或者存在即使算法实现正确，而传统可靠性保障技术无效的情况。为求更好地保障智能软件可靠性，本文分析智能软件可靠性模型，并将其细分为数据可靠性、模型可靠性和平台可靠性等。进一步地，本文从保障技术和软件过程两方面出发，进一步厘清智能软件可靠性的保障机制和技术手段——国内外研究者提出大量针对智能软件的软件测试、调试、错误修复和形式化验证技术，解决人工智能引入的不确定性，提升智能软件产品的可靠性；工业界也定义机器学习工作流并将其集成至智能软件开发过程中，以从软件过程上保障智能软件可靠性。最后，本文将总结开发可靠的智能软件所面临的挑战、机遇以及发展方向。

关键词：智能软件，可靠性，软件过程，软件测试，软件验证

Abstract

Artificial intelligence is revolutionizing the means of working, learning, living, discovering and communicating. Intelligent software is important for artificial intelligence, which defines new functions, performance and boundaries for artificial intelligence. However, we are currently still facing defects and failures caused by intelligent software. One reason is that artificial intelligence and machine learning have changed programming paradigm, where rules are drawn out and automatically learnt from training data, rather than from human developers. This change makes it difficult to reason, understand, analyze, test, and verify software behaviors.

Since intelligent software is playing an important role in our society, it is necessary to study and

develop new techniques to alleviate some critical challenges——intelligent algorithms may behave incorrectly in intelligent software, or traditional testing/verification techniques may be ineffective even if the algorithms are implemented correctly. This report studies the reliability model of intelligent software and investigates the reasons why intelligent software has failures. Furthermore, this report studies and surveys the software process and reliability techniques——researchers have proposed a large number of techniques for improving the reliability of intelligent software products through testing, debugging, bug fixing, and formal verification; they have also developed machine learning workflows and studied how to integrate machine learning workflows into software development processes in order to guaranteeing the reliability of intelligent software. Finally, this report summarizes the challenges, opportunities, and directions of developing reliable intelligent software.

Keywords: intelligent software, reliability, software process, testing, verification

1 引言

人工智能是一种具有巨大经济和社会效益的革新性技术，正在革命性的改变人类的工作、学习、生活、发现和沟通方式。人工智能研发能够模拟、延伸和扩展人类智能的理论、方法、技术及系统，促使机器会听（语音识别、机器翻译等）、会看（图像识别、文字识别等）、会说（语音合成、人机对话等）、会学习（机器学习、知识表示等）、会思考（人机对弈、定理证明等）、会行动（机器人、自动驾驶汽车等）。

近些年来，人工智能领域的高速发展得益于大数据技术的发展、统计和概率方法的采用及计算机处理能力的提升。特别是，机器学习或深度学习支持计算机从经验或例子中学习[1,2]，已经表现出越来越精准的结果，引发人们对人工智能前景更多的兴趣。

智能软件是人工智能的重要载体。智能软件包含能产生人类智能行为的应用软件，也包含支持数据采集和人工智能算法（比如机器学习、深度学习等算法）的软件基础设施[237]。随着人工智能的发展，智能软件也表现出蓬勃发展的趋势，在社会各领域展现出广泛应用价值。

1.1 智能软件背景

人工智能技术体系通常包含 5 层：1）基础设施层，包括数据和算力；2）算法层，包含机器学习、深度学习等算法；3）技术方向层，如计算机视觉、语音工程、自然语言处理（NLP）等；4）具体技术层，如图像识别、图像理解等，提供包括推理、知识、规划、学习、感知、移动和操作对象能力；5）应用层，关注最为广泛的人工智能应用领域，如用户画像分析、基于信用评分的风险控制、欺诈检测、智能投顾、智能审核、智能客服机器人、机器翻译、人脸识别等。

智能软件承载着人工智能算法和实现，主要包括 1）为人工智能提供支持的软件基础设施和 2）产生智能行为的应用软件。软件基础设施主要指允许智能软件运行的软件

框架或人工智能平台。此外，人工智能依赖于数据。对智能算法而言，大数据很重要，且需要覆盖各种可能场景，才能训练出表现良好的模型。因此，软件基础设施需要集成大数据功能，包括数据采集、清洗、运算、存储、应用、展现等。

文献［5］列举了国外重要的人工智能平台/框架，包括谷歌人工智能平台、TensorFlow[3]、微软的 Azure[4]、Rainbird、Infosys Nia、Wipro HOLMES、Dialogflow、Premonition、Ayasdi、Mindmeld、Meya、KAI、Vital A. I、Wit、Receptiviti、Watson Studio、Lumiata、Infrrd 等。这些平台/框架模拟了人类思维的认知功能，提供解决问题、学习、推理等能力

国内的重要的人工智能平台/框架包括：

- **百度的飞桨深度学习平台**[6]。它于 2016 年正式开源，具有开发便捷的核心框架、支持超大规模深度学习模型训练、多端多平台部署的高性能推理引擎和产业级开源模型库等领先技术。
- **华为的 MindSpore 深度学习框架**[7]，是一款面向端边云全场景的全新深度学习训练/推理框架，旨在降低开发者学习成本，提高数据科学家的研发效率，以及通过提供一套统一框架加速面向全场景的 AI 新应用落地。
- **星环科技的 Sophon 人工智能平台**[8]，支持用户可以快速完成从特征工程、模型训练到模型上线的机器学习全生命周期开发工作。Sophon 平台还集成知识图谱工具、实体画像工具、报表工具、视频分析工具等。
- **旷视科技的天元 MegEngine 深度学习框架**[9]，是一个工业级的深度学习框架。它帮助开发者利用编程接口，进行大规模深度学习模型训练和部署。架构上具体分为计算接口、图表示、优化与编译、运行时管理和计算内核五层，可极大简化算法开发流程，实现了模型训练速度和精度的无损迁移，支持动静态的混合编程和模型导入，内置高性能计算机视觉算子，尤其适用于大模型算法训练。
- **腾讯的 Angel**[10]是基于参数服务器架构的分布式计算平台，致力于解决稀疏数据大模型训练以及大规模图数据分析问题。Angel 尝试打造一个全栈的机器学习平台，功能特性涵盖了机器学习的各个阶段：特征工程，模型训练，超参数调节和模型服务。
- **微众银行的 FedAI 联邦学习 FATE 开源平台**[11]，提供了一种基于数据隐私保护的分布式安全计算框架，为机器学习、深度学习、迁移学习算法提供高性能的安全计算支持，支持同态加密、SecretShare、DiffieHellman 等多种多方安全计算协议。
- **开放智能的边缘 AI 推理架构 Tengine**[12]是一个嵌入式的 AI 推理框架，其架构分为模型兼容层、基础工具链、模块化架构层、操作系统层、异构计算层五大框架层级，具有跨算法框架兼容、跨芯片适配、算力异构调度加速、轻量无依赖、一致性开发移植部署工具链几大核心能力。

目前，已有许多智能软件和技术，广泛应用于金融、医疗、安防、交通、工业、媒体等领域，具体包括：在城市管理领域，解决城市和社会治理难题，包括城市安防类、智慧水务管理等；在金融领域，主要用于风险监测、交易监管等；在工业制造领域，实

现优化流程、质量监控、良率预测等功能；在医疗领域，提供对影像进行快速检测，可帮助进行大规模病例筛查，并高效、准确地辅助医生快速进行病情分析和疗效评价；在媒体领域，集成视频、图像、语音处理技术，建立连接用户和媒体之间交互的桥梁；在交通出行领域，依靠大数据技术，将多元异构的数据融入平台，实现自动驾驶和车辆智能运维；在教育领域，为个性化教育提供支持，如通过声音、人脸识别等技术帮助教师掌握学员学习状态，同时 VR 等技术可以加强可视化教学服务等。值得注意的是，人工智能应用的领域非常多，本身又在不断发展过程中。不同领域智能软件具有强烈的领域特征，并需要不同的人工智能算法和技术的支持。

智能软件通过人工智能或者机器学习算法，模拟甚至超越人类感知、学习、推理和行为能力，实现对问题的求解。智能软件与传统软件存在如下区别：

- 基于问题求解。一个智能软件往往采用人工智能问题求解模式来获得结果。与传统软件相比，其问题往往具有指数型的计算复杂性，其问题求解在很大程度上依赖知识，问题求解算法往往是非确定的或启发式的。
- 基于知识处理。智能软件处理的对象，不仅有数据，而且还有知识。表示、获取、存取和处理知识的能力是智能软件与传统软件重要区别之一。因此，智能软件需要如下机制：表示知识的语言；知识组织工具；建立、维护与查询知识库的方法与环境；支持知识重用的机制。
- 基于现场感应。智能软件具有现场感知（环境适应）的能力，即它可与所处的现实世界进行交互。这种交互包括感知、学习、推理、判断并做出相应动作，即智能软件存在自适应性。

然而，智能软件不全是智能，其“智能”常常体现在数个搭载了人工智能或者机器学习算法的智能模块中，从而拥有比传统软件更强的感知、学习、推理、判断能力。智能软件仍需要更多的非智能模块。如图 1 所示，智能软件通常也不仅仅关于智能，还存在很多非智能的支撑模块、功能黏结代码、与大量常规软件模块，仍需要传统软件工程来解决的问题；开发团队需要同时具备软件工程技能和人工智能技能，且需要深刻认识到二者对于开发高可靠智能软件同样重要。

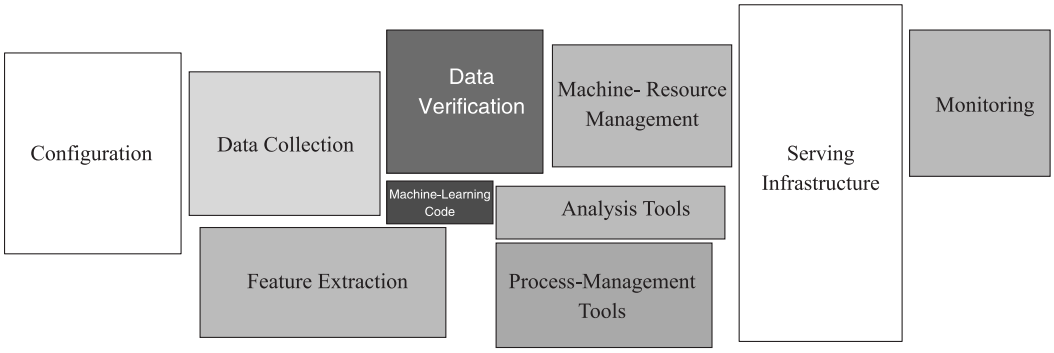


图 1　人工智能在智能软件中扮演的角色。在谷歌的智能软件中，人工智能模块仅占很小一部分，但却在整个软件决策中起到关键角色[13]

1.2 本文内容和组织安排

对更多自动化和智能的渴望推动了人工智能和机器学习进步，但我们仍然经历由此产生的智能软件的缺陷和失败，或者不可靠[14,236,242]。特别是，基于深度学习技术的智能软件系统正渗透到现代社会各个领域，例如图像分类、自然语言处理、双人博弈、自动驾驶系统、智能医疗诊断系统等，而软件缺陷可能会带来极高风险与损失。例如，谷歌自动驾驶系统的缺陷曾导致汽车向左变线时撞上公交车；优步和特斯拉的自动驾驶系统也均在路测时发生多起严重交通事故[15]。这些缺陷引发了全球范围对人工智能和深度学习技术可用性的激烈讨论，甚至一度导致自动驾驶技术的发展陷入了寒冬。

此外，2017 年 Deepfake 技术可以随意更改视频中人脸，使虚假视频、新闻泛滥[16]。2018 年“Facebook 数据泄露”事件中，美国剑桥分析公司利用广告定向、行为分析等智能算法，推送虚假政治广告，形成对选民政治观点的干预诱导，影响英国脱欧等政治事件走向[17]。2018 年 8 月，亚马逊智能音箱爆出后门，可实现远程窃听并录音[18]。2019 年 2 月，我国人脸识别公司深网视界曝出数据泄露事件，超过 250 万人数据、680 万条记录被泄露，包括身份证信息、人脸识别图像及 GPS 位置记录等[19]。智能软件的不可靠、不安全及对隐私的侵犯招致对智能软件的广泛担忧和批评声音。

鉴于智能软件在社会中发挥着愈来愈重要作用，软件工程和人工智能领域都有必要研究和开发技术以应对软件可靠性的挑战——智能软件中算法的行为可能不正确，或者存在即使算法实现正确，而传统可靠性保障技术无效的情况；软件可能缺少完整的规范，甚至不存在与某些智能行为对应的源代码；智能软件存在数据使用不当的情况。如何深入认识智能软件可靠性，如何开发可靠、真实和可信赖的智能软件，如何确保智能软件以受控的方式安全而可靠地运行，如何有效地测试、维护和演化智能软件，成为多领域共同关注的焦点。

本文主要包括五方面内容：

- 智能软件可靠性背景及报告基本内容概述。
- 从智能软件可靠性模型、保障技术和开发过程方面出发，介绍智能软件可靠性国际进展。
- 智能软件可靠性国内进展。
- 智能软件可靠性的挑战、机遇以及发展方向。
- 总结。

希望通过本文能为社会各界更了解智能软件可靠性有所帮助。在此，呼吁更多的中国企业、工程师与学者投身研究智能软件及其基础设施可靠性，推动可靠智能软件的发展。

2　国际进展分析

2.1　智能软件可靠性模型

软件系统中的可靠性属性被定义为工作产品或产品的属性，拥有者将通过它来判断其质量[20]。软件系统常常从满足其业务和任务目标，扩展到满足相应的可靠性需求。

智能软件可靠性是指在规定的条件下、规定的时间内，系统正确完成预期功能，且不引起系统失效或异常的能力[14]。总体而言，智能软件的可靠性可分为三大方面：软件/模型层面、数据层面、环境方面。软件/模型层面包括深度学习算法的正确性、代码正确性。数据层面包括训练数据集的影响和环境数据的影响：前者主要指数据的均衡性、数据集规模大小、数据集标注情况对可靠性的影响；后者包括对抗性数据、数据集分布变化以及野值数据对可靠性的影响。环境层面包括软硬件平台对智能软件可靠性的影响。

如图2所示，本文将智能软件可靠性细分为数据可靠性、模型可靠性、平台可靠性。

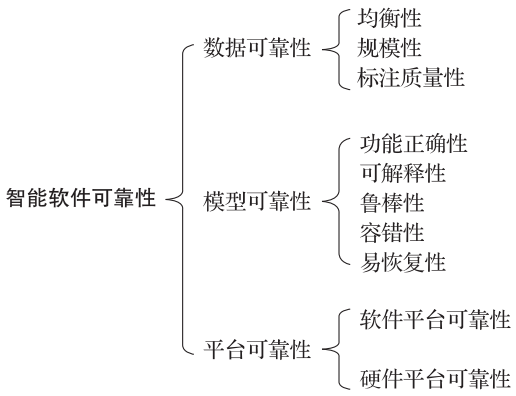


图2　智能软件可靠性维度

2.1.1　模型可靠性

模型可靠性评价智能软件中涉及的机器学习模型或深度学习模型可靠程度[22,23,24]，主要包括：

- 功能正确性，指模型对指定任务或目标完成结果的正确度。
- 可解释性，指模型对某一特定任务做出的决策的可靠程度。
- 鲁棒性，指模型面对异常情况是避免失效的能力。
- 容错性，指模型面对故障保持用户期望性能水平的能力。
- 易恢复性，指模型失效后在有限时间内恢复原始状态的能力。

最关键也是最难提升的是模型的可解释性。人工智能算法常常需要选择、训练及部署模型。工程师花费大量时间研究各种模型——他们花时间收集/生成数据，实验和尝试各种模型，以查看哪些模型适应他们的需求。为了提升智能软件可靠性，一个手段是使用可解释模型——了解模型何时以及什么情形下能做出准确的预测是一个热点研究问题，随着机器学习算法变得越来越复杂，模型可解释性正吸引着更多的关注。一些人工智能社区建议使用可解释模型，或开发可视化技术，使黑盒模型更易解释。另一方面，构建

或使用的模型需要具备可定制性和可扩展性，这要求开发团队不仅具备软件工程技能，且需要足够深厚的机器学习知识来解释、构建、评估和调整模型。

Sundarajan 等人认为好的解释应该满足两个属性：1）敏感性要求。对于每个只在一个特征上不同但预测结果不同的输入 x 和基础输入 x'，不同的特征应该给予非零属性。2）实现不变性。要求两个功能等价网络的属性始终相同。基于此，Sundarajan 等人提出梯度积分的方法[25]，并证明该方法是满足某些属性（包括灵敏度和实现不变性）的唯一方法。

Ribeiro 等人提出将决策规则表示成一组谓词，进行了局部预测[26]。该方法基于一种假设：虽然模型在全局上过于复杂而无法简洁地解释，但“放大”单个预测使解释任务变得可行。其形式化表示为：$A(x)$ 表示一个规则，$A(x)=1$ 当且仅当对于输入 x，所有的特征预测都是正确的。当且仅当 $E_{D(z|A)}[f(x)=f(z)]>1-\varepsilon$ 且 $A(x)=1$ 时 A 是一个有效的解释。

Shwartz-Ziv 等人提出一种基于信息论的信息流解释方法[27]。他们根据坐标（$I(X;T_K)$，（T_K，Y））引入信息平面的概念。这样 DNN 训练过程就可以通过可视化的信息平面来解释训练的 SGD 层动力学。训练过程可以分为两个阶段：1）在适应阶段，较高层次的神经元学习尽可能多的输入信息，并试图使其与输出相适应；2）在压缩阶段，更高层次的神经元压缩对其输入的表示，同时保留最相关的信息来预测输出。

Ribeiro 等人提出一种跨模型的局部解释：LIME[28]。该方法通过局部近似给定的预测来解释单个模型的预测。对于图像，LIME 将其视为一组超像素，通过最小化目标函数来计算其排序。

Kim 等人认为，大多数模型操作的特征（如像素值）不符合人类理解，现有的基于排名的方法不能产生一个人类容易接受的解释[29]。基于此，他们提出 TCAV 方法，该方法提供了高级概念的想法。TCAV 根据分类结果，使用方向导数对用户定义的概念进行排序，使最终的解释更接近人类可理解的解释。

2.1.2 数据可靠性

数据可靠性主要指用于训练机器学习或深度学习模型所使用的数据质量[30-31]。主要包括：

- 均衡性，指数据集分布是否均匀。
- 规模性，指用于训练模型的数据集的大小。
- 标注质量，指数据集中样本的标注是否恰当。

数据标注质量是数据可靠性的一个关注点。其中，MV[32]算法将绝大多数用户选择的结果视为最终结果。其基本思想是：假设有 m 个待标注的任务（t_1，…，t_m），每个任务 t_i 对应一个二元分类，在某众包平台上，管理者将这些数据分给员工。为了提高标注质量和标注的可靠性，将每个标注任务分给多个员工。每个工人 w_j 对 x_i 做出预测并创建一个标签，然后根据 x_i 的所有标签推断出 x_i 的最终标签。

MV 算法把大多数人认为正确的标签作为最终标签。在现实生活中，大多数人认为正确的并不总是正确。Dawid 等人提出 EM 算法[33]。EM 算法构建出一次标注任务中标注

者的标注错误率混淆矩阵，并与实际观测标注结果进行比较，二者比较结果的差异越大，代表标注的结果越差。RY 算法是在 MV 算法和 EM 算法基础上的改进算法[34-35]。

Demartini 等人提出的 ZenCrowd 算法使用二元组 {good，bad} 来建模众包标注者的可靠性[36-37]。ZenCrowd 算法计算每个众包标注者的可靠度以及使用可靠度来更新每个样本属于特定类别的概率，标注者的可靠度越高，标注质量就越好。

此外，还存在针对特定数据进行标注的技术。在图像方面，Vedantam 等人提出的 CIDEr 算法是一种评估图像描述的算法[38]。CIDEr 算法将每个句子都看作“文档”，将其表示成 TF-IDF 向量的形式，通过对每个 n 元组进行权重计算，比较模型生成的描述与人工描述之间的余弦相似度．余弦相似度越高，图像描述的质量越好。Anderson 等人提出用于评估图像描述质量的 SPICE 算法[39]，使用基于图的语义表示来编码标注数据中的对象、属性和关系，将待评价标题和人工标注标题用概率上下文无关文法依赖解析树解析成语法依赖树，用基于规则的方法把依赖解析树映射成场景图，最后用 F-score 来评估两个场景图的相似性，分值越高，表示图像的描述质量越好[40]。

在文本方面，常用相似性来度量数据标注质量。Papineni 等人提出一种基于精确度相似性度量的方法：BLEU 算法[41]。该方法根据待评估数据和参考数据中 N 元组共同出现的程度来衡量标注数据的质量。Yew 提出一种基于召回率的相似性度量方法 ROUGE[42]，考察待评估标注的充分性和忠实性，并计算 N 元组在参考数据和待评估数据的共现率以评估文本标注的质量。Lavie 等人提出 METEOR 算法[43]，这是一种基于精确度和召回率的相似性度量方法，与单纯基于精度的 BLEU 算法相比，其结果和人工标注结果相关性较高。

在语音标注方面，语音标注质量评估算法主要有词错误率 WER 算法[44]和句子错误率 SER 算法[45]。词错误率表示为了让识别出来的词序列和标准的词序列之间保持一致，而需要进行替换、删除或者插入的某些词。SER 算法被用来识别句子中是否出现词识别错误，SER 的值越高，代表语音标注的质量越差；反之，代表语音标注的质量较好。

2.1.3 平台可靠性

平台可靠性指衡量智能软件所依赖的软硬件平台的可靠程度[46-48]。主要包括：

- 软件平台可靠性，指智能软件所依赖的软件平台（包括深度学习框架、大数据平台、操作系统等）的可靠程度[21,232]。
- 硬件平台可靠性，指智能软件所依赖的硬件（如传感器、人工智能芯片等）和平台的可靠程度。

Islam[72]等人选择当前最流行的五个深度学习框架：Caffe、Keras、Tensorflow、Theano、Torch，从 Stackflow 上收集了 2716 个高质量的问题并从 Github 上收集了 500 个错误修复提交来研究深度学习框架使用时出现错误的类型、错误根源、错误的影响以及易出现错误的阶段是否出现异常模式。研究发现，深度学习框架的错误主要是数据错误和逻辑错误，导致错误的主要根源是结构低效（SI）和模型参数错误（IPS），大多数错误发生在深度学习过程的数据准备阶段。他们还发现错误的分布和异常模式之间存在很

强的相关性。

此外，存在软硬件平台（如云平台甚至网络）的可靠性模型的进展。在此本文不予赘述。

2.2 软件的可靠性保障技术手段

为了实现主动的可靠性保障，通常手段包括验证与测试，其保障过程往往建立于程序分析的基础上：

- 测试：验证问题通常具有很高的计算复杂度，例如，当属性是简单的输入－输出约束时，就会遭遇 NP 难问题。此外，DNN 的高维度和高度非线性等特点也决定了现有的验证技术很难用于工业规模的 DNN。通过以可证明的保证为代价来考虑测试技术，可以部分缓解这种计算强度。相反，保证案例是按照现有的安全攸关系统的要求进行的。
- 验证：给定一个深度神经网络 N 和属性 C，验证即检查 N 是否满足 C 的一组技术。与其他技术（例如测试和对抗攻击）不同，验证技术需要为其结果提供可证明的保证（可以是布尔保证、近似保证、随时近似保证或统计保证）。布尔保证表示验证技术能够确定该属性何时成立，否则返回反例；近似保证可为属性提供过近似或欠近似；随时近似保证同时具有过近似和欠近似，且近似程度可以不断提高直到收敛。以上所有保证都需要数学证明，当难以获得数学证明时，统计保证可以为结果断言提供定量误差容限。

此外，对于智能软件而言，深度学习模型通常是其核心组件，基于对抗样本的可靠性保障技术近年来获得了较大关注。

2.2.1 测试用例生成技术

1）特定领域的测试输入合成技术。智能软件的测试用例可以被分为两类：对抗输入和天然输入。对抗输入是基于原始输入扰动得到的，它们可能不属于正常的数据分布（例如：可能现实中较少出现），但是可以暴露鲁棒性和安全性上的缺陷。与之对应的，天然输入是属于真实应用场景的数据分布的输入。

Pei 等人提出了首个基于白盒的差分测试技术 DeepXplore[49]，用于为深度学习系统生成测试输入。受到传统软件测试中的测试覆盖率的启发，作者提出“神经元覆盖率驱动测试输入生成”这一思想，指出所生成的测试输入应具有高神经元覆盖率。此外，输入需要揭示不同 DNN 模型之间的差异，并尽可能地与真实世界的数据相似。联合优化算法迭代地使用梯度搜索来找到满足所有这些目标的修改的输入。DeepXplore 的评估表明它比相同数量的随机选择输入和对抗输入分别多覆盖 34.4% 和 33.2% 的神经元。

针对智能驾驶系统 Tian 等人提出 DeepTest[50]，该方法使用九种不同的真实感图像变换进行贪婪搜索：改变亮度、改变对比度、平移、缩放、水平剪切、旋转、模糊、雾效果和雨效果生成测试用例。DeepTest 的评估使用了 Udacity 自驾车挑战数据集，该数据集

在 CNN 和 RNN 上检测到 1000 多个错误行为，假阳性率较低。

在自然语言处理（NLP）与基于大数据的代码分析（Big Code）领域，Rabin 等人[51]讨论了使用保留语义的程序转换结果作为测试输入，来测试一种代码嵌入方法 code2vec[52]的可能性。

2）基于模糊与搜索的测试用例生成技术。模糊测试是一种传统的自动测试技术，它生成随机数据作为程序输入来检测崩溃、内存泄漏、（内置）失败断言等，并成功地应用于系统安全和漏洞检测。作为另一种广泛应用的测试生成技术，基于搜索的测试生成通常使用元启发式搜索技术来指导模糊过程，以获得更高效、更有效的测试生成[53-55]。这两种技术在探索机器学习测试的输入空间方面被证明是有效的。

Odena 等人[56]提出 TensorFuzz，它使用最近邻爬山方法来探索在有效输入空间上 Tensorflow 计算图的可实现覆盖，并发现数值错误、神经网络与其量化版本之间的差异以及 RNN 中的不良行为。Wicker 等人[57]提出特征引导的测试生成。他们采用尺度不变特征变换（SIFT）来识别用高斯混合模型表示的图像特征，然后将寻找对抗性例子的问题转化为基于两人轮流的随机博弈。他们使用蒙特卡罗树搜索法来识别图像中最易受攻击的元素，以此作为产生对抗性例子的手段。实验表明，他们的黑盒方法与一些最先进的白盒方法相比具有更好的竞争力。Uesato 等人[58]提出用生成对抗数据样本的方式来评估强化学习。由于灾难性故障较少见，检测灾难性故障的代价非常昂贵。为了降低发现此类故障的代价，作者提出使用失效概率预测器来估计智能体失效的概率，并证明了该方法的有效性。

除了图像分类之外，还有一些针对特定应用场景的模糊器。Zhou 等人[59]结合模糊测试和蜕变测试，测试了现实生活中自动驾驶汽车的激光雷达障碍感知模块，并报告了之前未发现的软件故障。Jha 等人[60]则研究了如何通过将故障注入建模为贝叶斯网络，来生成最有效（最有可能导致破坏安全性的故障）的测试用例。这项评估基于 NVIDIA 和百度的两个生产级 AV 系统，揭示了许多故障导致安全性被破坏的情况。Udeshi 和 Chattopadhyay[61]为文本分类任务生成输入，他们提出一种考虑被测语法以及与输入之间距离的模糊化方法。Nie 等人[62]和 Wang 等人[63]分别各自对 NLI（自然语言推理）任务中的句子进行变异，从而生成测试输入以进行健壮性测试。Chan 等人[64]为 DNC 生成了对抗数据样本，以暴露其健壮性方面的不足。Udeshi 等人[65]关注个体公平性，生成的测试输入能够揭示被测模型的歧视性。Tuncali 等人[66]提出一个测试自动驾驶系统的框架。他们比较了三种测试数据生成策略：随机模糊测试数据生成，覆盖数组[67] + 模糊测试数据生成，和覆盖数组 + 基于搜索（使用模拟退火算法[68]）的测试数据生成。结果表明，采用基于搜索技术的测试数据生成策略在检测观察行为方面具有最好的性能。

3）基于符号执行的测试用例生成技术。符号执行是一种程序分析技术，用于测试被测试的软件是否会违反某些属性[69]。动态符号执行（DSE）是一种用于自动生成实现高代码覆盖率的测试数据的技术。DSE 将随机测试输入输入被测程序，同时并行进行符号执行，收集从执行路径上的分支语句中的断言获得符号约束。在智能软件中，模型的性能不仅取决于代码，还取决于数据。因此符号执行有两种应用场景：对数据进行和对代码进行。

Ramanathan 和 Pullum 等人[70]应用了符号分析以生成更有效的测试数据来暴露错误。他们提议将符号和统计方法结合起来以有效地寻找测试用例，想法是使用符号从距离理论上抽象数据，以帮助搜索那些对输入进行细微变化即可导致算法失效的测试输入。对 k-means 算法实现的评估表明，该方法能够检测到位翻转之类的细微错误。

在机器学习代码上应用符号执行时，存在许多挑战。Gopinath 等人在[71]中列出了在神经网络上应用符号执行所面临的三个挑战：1）网络没有明确的分支；2）网络可能是高度非线性的，没有完善的约束求解器；3）存在可扩展性问题，该问题源于机器学习模型的结构复杂程度常常超出了当前符号推理工具的能力。针对上述挑战，Gopinath 等引入了 DeepCheck[71]，它将深度神经网络（DNN）转换为程序，以使符号执行能够查找与原始图像具有相同激活模式的像素攻击。由于 DNN 中的激活功能遵循 If-Else 分支结构，该方法将其视为经过翻译程序的路径。DeepCheck 能够通过识别神经网络无法对相应修改图像进行分类的像素或像素对来创建 1 像素和 2 像素攻击。

类似地，Agarwal 等人[73]使用 LIME[28]（一种局部解释工具）来获得符号执行中使用的路径。基于 8 个开源公平性基准的评估表明，该算法生成的成功测试用例比随机测试用例生成方法 THEMIS[75]多 3.72 倍。Sun 等人[76]提出一种 DNN 的动态符号执行测试方法 DeepConcolic：通过具体评估 ML 模型的给定属性，具体执行可用于将符号分析引导到特定的 MC/DC 标准条件。DeepConcolic 明确地将覆盖需求作为输入。与 DeepXplore 相比，DeepConcolic 对评估模型的神经元覆盖率高出 10%。

2.2.2 对抗样本生成

对抗样本（Adversarial examples）是一种由人为添加扰动的特殊样本，其概念由 Szegedy 等人提出[78]。对抗样本可以使模型产生误判，使攻击者达到绕过模型检测的目的，甚至可导致基于此类模型的各种异常检测算法失效。现有的模型很容易受到“对抗样本”的攻击，因对抗样本等攻击噪声产生的深度学习模型错误而引发的可靠性问题是亟待解决的问题。基于对抗样本的智能软件深度学习模型可靠性保障技术是近年来关注热点，有大量相关研究成果产生。

1）计算机视觉领域研究进展。Szegedy 等人[78]给出了一个对抗样本生成方法，称为有边界约束的 L-BFGS。该方法通过简单的最优化过程，对一个能够正确分类的输入图像进行微小的扰动，使其错误分类。Szegedy 等人通过引入损失函数近似求解问题，在该问题优化目标中，对生成对抗样本与原样本相似性进行约束，并最小化对抗样本在某一目标标签上的损失函数，使分类器尽可能将对抗样本误分类为目标标签。该方法通过不断变化的约束常数的值，找到一个距离原样本最小对抗样本，同时使分类器误分类。然而，L-BFGS 要保证模型和待优化函数的梯度可求解，这降低了 L-BFGS 的可适用范围，并且该算法需要解决一系列代价昂贵的惩罚性优化问题，在产生对抗样本时的计算代价很高。

Goodfellow 等人[79]认为模型在面对对抗样本时的脆弱性来自深度学习模型的线性本质。他们提出能够快速产生对抗样本的方法 FGSM。FGSM 通过最小化对抗样本上的损失函数，使模型误分类生成样本。FGSM 通过值计算一次梯度，能够更快生成对抗样本，

适用于需要生成大量对抗样本的场景。

Moosavi Dezfooli 等人[80]在 L2 距离算法上提出一种非目标性对抗样本生成算法：Deepfool。该方法利用泰勒公式建立一个线性超平面，对每一个数据点 x，试图寻找该数据点越过决策边界的最短路径，使得模型对数据 x 误分类。该方法表明，对于深度神经网络分类器，绝大部分测试样本都非常靠近其决策边界，在 MNIST 数据集上超过 90% 的测试样本对于 L_∞ 范数小于 0.1 的扰动会分类错误。

Kurakin 等人[81]对快速梯度符号法进行改进，提出基于迭代的符号梯度算法 BIM。该算法在 FGSM 基础上引入迭代，使生成的对抗样本优于 FGSM 生成的对抗样本。Papernot 等人[82]提出基于求解目标函数 F 的雅可比矩阵的方法来生成对抗样本 JSMA。JSMA 是一种通过对模型输出影响最大的像素点进行迭代操作的贪婪攻击算法，其通过使用雅可比矩阵来模拟模型在输入改变时的变化程度——在目标攻击中，算法首先求解雅可比矩阵，选择对结果影响最大的像素，并对其进行修改，在最大程度上使模型误分类为目标类。循环该过程，直到目标标签的概率大于其他标签，或达到最大迭代次数。

Nicholas Carlini 和 David Wagner 提出著名的 C&W 攻击[83]，这种对抗样本生成方法可以成功使目前的防御网络（如蒸馏网络）失效，C&W 基于参数调节进行优化，通过调节置信度，生成具有攻击性且扰动最小的对抗样本。

Brendel 等人[84]提出一种基于决策的对抗样本生成方法，也被称为边界攻击。Wicker 等人[85]提出特征引导的对抗性样本的鲁棒性测试方法，借助尺度不变特征转换算法提取特征，通过双方博弈方式确定特征和待操作的像素点，并利用蒙特卡罗树搜索来生成对抗性样本。黑盒攻击方面，Papernot[86]等人提出首个黑盒攻击方法来攻击 DNN 分类器，黑盒攻击下攻击器对模型的参数、训练集等均未知，仅能获得对应输入的输出结果。Papernot 等人利用对抗样本的可迁移性，在对一个与目标模型结构功能相似的模型上生成对抗样本，再利用该样本攻击目标模型。Chen 等人[87]假设攻击者可以访问模型输出的预测置信度，即每个类别的概率大小，他们观察调整输入 x 的像素值时预测置信度 $F(x)$ 的变化来近似扰动样本周围的梯度信息，获得更高攻击成功率。

2）**自然语言处理领域研究进展**。Papernot 等人研究文本中对抗性样本问题，成功生成了对抗性序列[88]。作者基于 FGSM，利用计算图展开来评估前向导数，这与单词序列的嵌入输入有关；用 FGSM 计算结果来找出对抗扰动。由于文本的离散特性，修改词的对应向量可能不存在。为了解决这个映射问题，作者设置了一个特定字典，从中选择替换原始单词的单词。虽然它们的对抗性序列会使 LSTM 模型产生错误的预测，但输入序列中单词被随机选择替换，因此对抗性示例出现语法错误概率非常高。相似的，Samanta 等人[89]提出一种单词修改算法来生成对抗样本。该方法利用 FGSM 的思想对重要或突出的词进行评估，若修改这些词，会对分类结果产生很大影响。然后利用三种修改策略（插入、替换和删除）对最重要的 k 个单词进行修改。

Sato 等人[90]提出一种基于优化的非目标攻击方法 iAdv-Text。其核心是一个优化问题，该方法定义每个样本在方向向量上最坏情况的权重，通过最小化在整个数据集上的损失函数以及对权重进行优化。iAdv-Text 限制了查找替换的扰动方向，采用预定义词汇

表中单词，而不是未知单词完成替换。同时，该方法利用余弦相似度来选择更好的扰动，保持易读性和语义相似性。

Gao 等人提出一种黑盒攻击方法来生成对抗样本 WordBug[91]。整个过程分为两阶段，第一阶段确定要改变哪些重要单词，第二阶段对重要单词应用翻转、删除、插入等操作。

对于目标攻击，攻击者有目的地控制想要的输出类别，生成语义保留对抗的例子。目标攻击的难度要高于非目标攻击。Javid 等人提出针对字符级文本分类器的对抗样本生成器 HotFlip[92]。HotFlip 进行原子翻转操作以生成对抗样本——原子翻转操作指的是通过梯度计算，选择一个单词中的字符转换为另一字符，找出损失的最大增加量。除了字符级攻击外，HotFlip 还通过不同的修改用于单词级攻击。虽然 HotFlip 的效果很好，但在严格的约束条件下，只有少数成功的对抗性例子是通过一次或两次翻转生成的，不适合大规模的实验。Javid 等人又基于 HotFlip 提出一种基于优化的目标攻击方法[93]。除了 HotFlip 提供的交换、插入和删除之外，作者还提出一种控制攻击（从输出中删除特定的单词）以及一种目标攻击（用选定的单词替换特定的单词）。为了实现攻击，他们需要最大化原词的损失函数，最小化待替换词上的损失函数。

Gil 等人提出一种基于倒数的方法 DISTFLIP[94]，利用 HotFlip 获得的信息对黑盒模型进行训练，通过训练模型来生成黑盒攻击下的对抗样本。该方法生成对抗样本的速度是 HotFlip 的十倍。考虑到黑盒攻击下对梯度获取的限制，AlzAntot[95] 等人提出一种基于遗传算法的优化方法，从输入中随机选择替换单词，并利用 GloVe 嵌入空间中的欧几里得距离计算它们的最近邻并进行替换。对上述过程重复多次可以生成对抗样本。

Jia 等人提出针对阅读理解系统的对抗样本生成方法[96]。该方法针对段落中名词和形容词用其同义词进行替换并反复查询网络输出，并检测模型结果变化来获取具有对抗性的句子。在段落中插入这些句子但不改变段落的语义和问题的答案，以欺骗神经模型。结果表明，将分散注意力的内容附加到原始段落中，可以形成对攻击的 DNN 的敌对输入。

Belinkov 等人首次对机器翻译系统进行攻击，根据自然和合成的语言错误设计了对抗样本，包括打字、拼写错误或其他错误，通过随机或输入错误来生成对抗样本[97]。

3）**其他领域研究进展**。Eykholt 等人[98]提出对交通识别系统的攻击。他们对停止标志的交通路牌图像基于 L1 范数选择扰动区域，再基于 L2 范数来生成贴纸颜色。在扰动区域上添加选定颜色的贴纸，可以从任何距离和角度使自动驾驶系统产生误判。同样，Thys 等人[99]通过对行人检测系统进行攻击，将对抗图像贴在行人身上使系统无法检测到行人。Wu 等人[100]通过让行人穿上特殊的衣服成功使检测系统误判。

Vaidya 等人[101]利用人类和机器语音识别的差异对语音系统进行攻击。Song 等人[102]利用麦克风固有的非线性特性，通过向语音信号中注入被编辑过的超声波语音命令来攻击 Siri 等语音识别系统，达到操纵手机甚至汽车导航系统的目的。

2.2.3 测试谕言缺失问题解决方案

测试谕言（test oracle）指的是对于给定的测试输入，待测程序应给出的正确预期输出（或得到预期输出的计算方法）。在测试过程中，只有确定了测试谕言，才能够精确判断一

条测试用例是否成功。在现实测试实践中，存在大量待测程序，无法（或难以以较小开销）获得其测试谕言，对此类现象，我们统称为“谕言缺失问题”（oracle problem）[103]。报告针对智能软件测试中测试谕言缺失问题解决方案进行介绍。

1）基于蜕变测试的谕言缺失问题解决方案。针对这一问题研究最多的是基于蜕变测试的解决方案。蜕变关系是 Chen 等人为解决传统软件测试中的测试谕言问题而提出的[104]。蜕变关系是指在程序执行过程中，多个软件输入变化和输出变化之间的关系。例如，为了测试函数 $\sin(x)$ 的实现，可以检查当输入从 x 更改为 $\pi - x$ 时，函数输出的变化情况。如果 $\sin(x)$ 与 $\sin(\pi - x)$ 不同，可以基于观测结果发出错误信号，无须检查计算的特定值是否准确。因此，$\sin(x) = \sin(\pi - x)$ 是一种蜕变关系，起到测试谕言（也称为“伪谕言”）的作用，帮助缺陷检测。

在机器学习软件测试中，为解决 oracle 问题，众多工作对蜕变关系进行了广泛研究。许多蜕变关系都是基于训练或测试数据的转换，基于蜕变关系得到的数据在预测输出中应当产生与原始结果一致的或发生某些可预期的变化的结果。在研究相应的蜕变关系时，数据变化的粒度是不同的。有些转换会进行粗粒度的更改，如：扩展数据集或更改数据顺序，而不更改每个单独的数据实例。有些转换通过对每个数据实例进行较小的修改来进行数据更改，如：改变图像的属性、标签或像素。

对于粗粒度数据转换，2008 年，Murphy 等人讨论可以作为蜕变关系的机器学习算法的性质[105]。他们介绍六种输入变换，包括给数值添加一个常量；将数值乘以一个常数；改变输入数据的顺序；颠倒输入数据的顺序；删除部分输入数据；添加额外的数据。他们对 MartiRank、SVM Light 和 PAYL 进行了分析。这项工作为确定用于机器学习的蜕变测试的关系和变换方法提供了基础。Murphy 等[106]提出函数级蜕变关系。通过对 9 个机器学习应用程序的评估表明，功能级属性比应用程序级属性的效果强 170%。

关于细粒度的数据转换：Nakajima 分析了用于测试支持向量机与神经网络的蜕变关系的差异[107]，并讨论了在支持向量机和神经网络中使用不同粒度的蜕变关系来发现问题的可能性，涉及的蜕变关系包括操作实例顺序或属性顺序、反转标签和更改属性值、操作图像中的像素等[108-109]。Dwarakanath 等人将蜕变关系应用于支持向量机和深度学习系统的图像分类[110]。数据的变化方法包括：改变特征或实例顺序，测试数据特征的线性缩放，测试数据的归一化或放大，或者改变数据的卷积运算顺序。提出的蜕变关系能够发现 71% 的注入缺陷。Sharma 和 Wehrheim 在文献［111］中也考虑了细粒度的数据变换，例如：更改特性名称、重命名特性值，以测试软件公平性。他们研究了 14 个分类器，发现没有分类器对特征名称变换敏感。

Zhang 等人提出将蜕变关系和数据突变相结合的扰动模型验证方法（PMV）[112]，用来检测过拟合情况。PMV 通过在训练数据中注入噪声对训练数据进行变异，生成扰动的训练数据集，然后在噪声程度增加时检查训练精度下降率。训练精度下降越快，机器学习模型过拟合程度越低。Al-Azani 和 Hassine 研究了朴素贝叶斯、k 近邻以及它们的组合分类器的蜕变关系[113]。结果发现，朴素贝叶斯和 k 近邻所必需的蜕变关系可能不是它们的合集分类器所必需的。Tian 等人[50]和 Zhang 等人[115]各自指出，在不同天气条件下，

对于变换后的图像，自动驾驶车辆转向角不应发生显著变化或保持不变。Ramanagopal 等人使用相似图像的分类一致性作为测试自驾车的测试谕言[116]，该方法在未标记数据上检出错误的精度为 0.94。

此外，还有研究者根据不同数据集之间的一致性设计蜕变关系，用来检测数据中错误。例如，Kim 等人[117]和 Breck 等人[118]研究了训练数据与新数据之间的蜕变关系。如果训练数据和新数据具有不同的分布，则训练数据可能不够充分、准确。Breck 等[118]还研究了时间上相近的不同数据集之间的蜕变关系：由于数据生成代码很少发生频繁的剧烈变化，这些数据集应当存在一些共同特征。

在上述工作的基础上，出现了一系列蜕变测试的应用框架。Murphy 等人[120]实现了一个名为 Amsterdam 的框架来自动使用蜕变关系对机器学习软件进行错误检测。该框架通过在结果比较时设置阈值来减少误报。他们还开发了 Corduroy[106]，允许为机器学习软件指定蜕变属性和生成测试用例。

2）基于交叉参考的谕言缺失问题解决方案。另一种常用测试谕言是交叉参考。交叉参考是包括差分测试和 N 版本编程。差分测试是一种传统的软件测试技术，它通过观测相似应用程序是否对相同的输入产生不同的输出来检测错误[122-124]。根据 Nejadgholi 和 Yang[125]的研究，对于深度学习库进行测试的测试谕言中，5% ~27% 使用了差分测试。差分测试与 N 版本编程密切相关[126]：N 版本编程旨在基于同一个规范生成多个功能等价的程序（版本），使不同版本程序组合结果具有更强的容错性和健壮性。

Davis 和 Weyuker[123]讨论了“不可测试”程序的差分测试的可能性：如果一个算法的多个实现在一个相同的输入上产生不同的输出，那么至少有一个实现包含缺陷。Srisakaokul 等人[128]在机器学习上对这一思想进行了评估验证，成功从 7 个朴素贝叶斯的实现中发现 16 个错误，从 19 个 k 近邻实现中发现 13 个错误。

3）针对其他非功能性测试谕言解决方案。除了上述对于软件功能进行描述的测试谕言之外，还有一部分工作给出了智能软件非功能特性的定义或统计度量，如：鲁棒性[129]、公平性[130-132]以及可解释性[133-134]。这些度量不是直接谕言，但对于测试人员理解和评估被测属性，及提供一些可以与预期值进行比较的实际统计数据是必不可少的。例如，深度学习软件系统必须满足不同类型公平性[130-132]，公平性定义可以用于检测违反公平性的行为。

2.2.4 测试充分性相关研究

在传统的软件测试中，代码覆盖率是测试充分性的评估准则之一，它衡量测试套件对程序源代码的执行程度。测试套件的覆盖率越高，隐藏的错误就越有可能被发现。通常情况下，能够实现更高覆盖率的测试套件往往是更好的。与传统软件不同，由于机器学习软件的决策逻辑不是手动编写的，而是从训练数据中学习的，因而代码覆盖率很少情况下会成为机器学习软件测试的一个严格标准。例如，在 Pei 等人[49]的研究中，传统代码覆盖率很容易通过单个随机选择的测试输入达到 100%。针对上述问题，研究人员提出多种适用于智能软件深度学习模型的新型覆盖率准则。

Pei 等人[49]提出首个为深度学习测试而设计的覆盖标准——神经元覆盖率。神经元覆盖率指由所有测试输入激活的唯一神经元的数量与 DNN 中神经元总数的比率。具体而言，如果神经元的输出值大于用户指定的阈值，则神经元被认定为激活。Sun 等人在 MC/DC 覆盖准则的启发下，根据 DNN 的不同特点，观测神经元的符号、值或距离的变化，以捕捉测试输入中的因果变化[140]。该方法假设 DNN 是一个完全连通的网络，不考虑神经元在其自身层中的上下文以及同一层中不同的神经元组合[141]。Sekhon 和 Fleming[141]定义一个覆盖标准，寻找 1）同一层中的所有成对神经元具有所有可能的值组合；2）连续层中的所有成对神经元具有所有可能的值组合。

虽然上文的准则在一定程度上捕捉了前馈神经网络的行为，但它们并没有清楚地描述像递归神经网络（RNN）那样的带状态的机器学习系统。基于 RNN 的机器学习方法在处理顺序输入的应用中取得了显著的成功，例如：语音音频、自然语言、网络物理控制信号。为了分析这种带状态的机器学习系统，Du 等人[143]提出一套专门针对基于 RNN 的带状态的深度学习系统的测试准则。他们将有状态的深度学习系统抽象为一个概率转换系统；在建模的基础上，他们提出基于变化系统的状态和轨迹的准则，捕捉动态的变化行为。

而除了单神经元与多神经元组合覆盖率准则之外，也可以通过分析整个层神经网络特征空间对充分性评估方法。例如，Kim 等人[117]引入了 Surprise Adequacy 来度量深度学习系统离散输入在新颖方面上的覆盖率。与单个神经元不同，Surprise Adequacy 考虑了神经网络中一整层神经元的特征，进行统计特征提取。他们认为测试的多样性在根据训练数据进行测量时更有意义。与训练数据相比，“好”的测试输入应该是“充分但不过分新颖（surprise）”。具体而言，他们提出两种 Surprise Adequacy 的度量方法：一种基于 Keneral 密度估计（KDE）来估计系统在训练过程中出现相似输入的可能性，另一种基于表示给定输入的神经元激活轨迹的向量与训练数据之间的距离。目前的测试准则大部分基于深度神经网络的结构，故也称基于结构的覆盖准则（Structural Coverage）。与传统软件中代码覆盖率有一定的相似性。这些标准可用于检测对抗数据样本，然而其与语义与可解释性的关系仍是需要进一步研究的难题。

研究对抗数据样本、自然错误样本和基于新颖方面的标准之间的关系也是一个重要的研究工作。Breck 等人[145]采取了基于规则的测试充分性检查，提供了 28 个测试需要考虑的方向和一个谷歌在使用的评分系统，其重点在于评估给定机器学习系统的是否被有效地测试。28 个测试方向分为四类：1）机器学习模型本身的测试，2）用于构建模型的机器学习基础设施的测试，3）用于构建模型的机器学习数据的测试，及 4）检查随着时间变化机器学习系统是否正确工作的测试。它们可以应用于指导测试生成。例如，训练过程应该是可重复的；所有特性都应该是有益的；不存在比当前模型更简单但性能更好的其他模型。

虽然机器学习系统的测试是复杂的，但是当要构造一些基本测试用例来测试机器学习系统的基本功能时，设计上具备一些共同特性。智能软件的深度学习模型中的测试用例生成需要覆盖很大的输入空间；另一方面，需要对每个测试用例进行标记，以判断预测的准确性。上述两点将导致较高的测试用例生成成本。Byun 等人[146]使用交叉熵、意外性和贝叶斯不确定性等 DNN 指标对测试输入进行了优先排序。这些指标对于暴露那些

具有不可接受行为的输入具有良好效果，对再训练也很有帮助。

2.2.5 错误调试与错误修复

通过测试或形式化验证所检测出深度学习模型的错误，如何对其进行调试与修复是目前学术界与工业界面临的另一个难题。神经网络的行为由单个神经元与其之前的组合共同决定。因此，一个错误预测可以由一个或者多个神经元共同作用而完成。基于定位重要错误神经元的思想，Mode[77]通过对神经网络神经元进行分析，从而定位可能对神经网络产生错误结果相关神经元。Mode 进一步对训练数据进行选择，通过训练的方式修正可进一步能产生错误的行为。与 Mode 类似，Arachne[229]首先对与预测相关的神经元与权重进行敏感性分析，然后通过基于搜索的方法 Particle Swarm Optimisation（PSO）对神经网络行为进行局部调整，达到一定程度上的错误修复效果。与 Mode 和 Arachne 不同，Apricot[230]借鉴了集体学习的思想（Ensemble Learning），从多个训练数据的子集合训练多个参考神经网络，并用互相行为的关系逐步对错误行为进行修复，相关评测结果表明 Apricot 在提高模型预测精准性上有一定提高。

基于数据驱动开发的深度学习模型，其行为很大程度上由训练数据分布所定义。然而，学习所得的模型预测结果通常并不能很好描绘训练数据的分布。例如，一个只学习过猫和狗图像数据的二分类问题，当我们将熊猫作为输入，模型可能会以高置信度的将其误判为猫或者狗。目前，通过对输入数据数据与神经网络运行时行为关系，检测出不相关或者神经网络可能无法正确预测的数据，可以减少其在实际应用场景做出错误决定的风险。DeepMutation[158]通过在数据或者神经网络中进行相关扰动，达到对 DNN 决策边界的扰动变化。

然而，深度学习模型的错误的调试与修复仍面临重大困难与挑战。首先，模型行为本身的概率与不确定性。换而言之，深度学习模型通常无法保证 100% 行为正确。因此错误可能来源于深度学习本身内在局限性。其次，包括数据、训练程序、深度神经网络结构、动态学习算法以及支撑平台都可能是最终训练模型错误行为的原因。

2.2.6 基于确定性保证的验证方法

确定性保证的原理是将验证问题转换为一组约束（有或没有优化目标），进而可以使用约束求解器解决相关问题。之所以称为“确定性”，是因为求解器通常会对查询返回确定性答案（即可满足或不满足），这得益于近几年各种约束求解器（例如 SAT 求解器、线性规划 LP 求解器、混合整数线性规划 MILP 求解器、可满足性模理论 SMT 求解器）的成功应用。

Pulina 和 Tacchella 通过将网络抽象为一组线性算术约束的布尔组合，提出一种验证间隔属性的解决方案（可被轻松扩展以与 ReLU 激活函数的可达性一起使用）[147]。层之间的线性变换基本上可以直接编码，而非线性激活函数（如 Sigmoid）则通过分段线性函数来近似。结果表明，当抽象模型被证明安全时，具体模型也将同样被证明为安全。另一方面，虚假反例所触发的改进可以用来自动纠正错误行为。该方法在神经网络（拥有

10 + 个神经元且具有逻辑激活功能）上得到验证。

Katz 等人与 Ehlers 于 2017 年分别提出两个 SMT 求解器 Reluplex[148] 和 Planet[149]。用来验证具有 SMT 约束条件的神经网络的可表达性。在那些可以表示为对其他变量类型约束的布尔组合的问题上，SMT 求解器通常具有良好性能。通常情况下，SMT 求解器将 SAT 求解器与对其他理论的专用决策程序结合在一起。在网络验证中，它们可以使线性算法适应实数，原子（即最基本的表达式）的形式为 $\sum_{i=1}^{n} a_i x_i \leqslant b$，其中 a_i 和 b 是实数。

Narodytska 等人[150] 和 Narodytska[151] 提出通过简化布尔可满足性，来验证权重和激活均为二进制的一类神经网络（即二值神经网络）的属性。通过布尔编码，他们可以利用现代 SAT 求解器以及反例引导搜索程序来验证网络的各种属性。此外，该方法也适用于图像分类任务中使用的中等规模 DNN。

Lomuscio 和 Maganti[152] 用 MILP 编码全连接神经网络的行为。例如，隐藏层 $z_{i+1} = \mathrm{ReLU}(W_i z_i + b_i)$ 可以用以下 MILP 描述：

$$
\begin{aligned}
z_{i+1} &\geqslant W_i z_i + b_i \\
z_{i+1} &\leqslant W_i z_i + b_i + M t_{i+1} \\
z_{i+1} &\geqslant 0 \\
z_{i+1} &\leqslant M(1 - t_{i+1})
\end{aligned}
$$

其中，t_{i+1} 所含元素值为 0 或 1，且维度与 z_{i+1} 相同；$M > 0$ 是一个可被视为 ∞ 的常数。t_{i+1} 中的每一个 int 变量表示一个神经元被激活的可能性。优化目标可用于表示与边界有关的属性（例如 $\| z_1 - x \|_\infty$ 表示 z_1 到某个给定输入 x 的 L_∞ 距离），此方法同时具有可达性和区间性。

简单地使用 MILP 来验证网络或计算输出范围是非常低效的。Cheng 等人一方面开发了大量 MILP 启发式编码方法以加速求解过程，一方面还使用了并行的 MILP 求解器，这使得实验中计算核的数量（达到一定的极限）几乎呈线性加速[153]。在文献［154］中，Sherlock 交替进行局部和全局搜索以有效地计算输出范围。在局部搜索阶段，Sherlock 使用梯度下降法寻找局部最大值（或最小值），而在全局搜索阶段，其用 MILP 对问题进行编码，以检查局部最大值（或最小值）是否为全局输出范围。此外，Bunel 等人[155] 提出一种分支定界（B&B）算法，并认为基于 SAT/SMT 和基于 MILP 的方法都可以被视为它的特例。

2.2.7 基于收敛边界计算的验证方法

Kwiatkowska 团队基于可满足性模理论（SMT）开发了用于前馈多层神经网络的自动验证框架[156]。该框架的关键特征在于它可以确保找到错误分类（如果存在），并且可以逐层进行分析（先分析输入层，之后重点分析隐藏层，最后分析输出层）。在这项工作中，单个分类决策的安全性（即点态或局部鲁棒性）被定义为分类器结果对原始输入在较小邻域内扰动的不变性。形式化定义如下：

$$\mathcal{N}, \eta_k, \Delta_k \models x$$

其中 x 表示输入，$\mathcal{N}$ 表示神经网络，η_k 表示第 k 层中围绕输入 x 的激活区域，Δ_k 表示第 k 层中的一组操作。他们随后证明，Lipschitz 常数的存在可以保证 Δ 的最小操作[157,85]。具体地说，其验证算法使用了单 - 多径搜索技术，以全面覆盖与输入层或隐藏层有关的向量空间中的有限区域。同时，还使用 Z3 求解器进行逐层细化，确保较深层的局部鲁棒性可以包含较浅层的。尽管复杂度很高，但其可以被扩展到与 VGG16 等最先进的网络一起使用。此外，[157,85]通过 Monte-Carlo 树搜索有效缓解了搜索问题。

Ruan 等人[161]开发的软件工具 DeepGO 表明，深度神经网络中大多数已知的层都是 Lipschitz 连续的。基于此，他们提出一种基于全局优化的验证方法。具体而言，在一维情况下，提出一种利用 Lipschitz 常数计算下界并最终收敛于最优值的算法；而多维算法则在一维算法的基础上穷举搜索最优组合。该算法能够应用于最先进的神经网络中，但受到扰动维度的限制。此外，Ruan 等人在文献［162］中研究了如何基于汉明距离对已训练 DNN 的全局鲁棒性进行量化。他们提出一种迭代生成网络鲁棒性上下限的方法。该方法是随时可用的，它返回中间边界和鲁棒性估计值（该值在计算过程中逐渐但严格地得以改进）；该方法是基于张量的，即计算过程在一组输入上同时展开而非逐个进行，以实现高效的 GPU 运算；该方法保证无论是边界还是鲁棒性估计值都能收敛到最优。

2.3 可靠的智能软件的开发过程

软件过程是利用过程来保障软件可靠性的重要手段。软件过程指为一个为建造高可靠软件所需完成的一系列步骤、中间产品、资源、角色及过程中采取的方法、工具等。存在众多软件开发模型，包括瀑布模型、快速原型模型、增量模型、螺旋模型等。

为了提升软件开发效率，许多团队使用反馈密集的敏捷方法来开发软件——他们需要通过更快开发周期，在满足不断变化的客户需求方面做出响应。敏捷开发是一种以人为核心，以迭代方式循序渐进开发的方法，其软件开发过程称为“敏捷过程”。在这一过程中，软件项目被切分成多个子项目，各个子项目都经过测试，具备集成和可运行的特征。另一方面，敏捷过程有助于支持进一步的适应。

近些年，以支持可靠的软件开发为目的的 DevOps 的概念在软件开发行业中逐渐流行起来。DevOps 是一组过程、方法与系统的统称，用于促进开发（应用程序/软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合。例如，许多团队围绕 DevOps[163-166]重新组织软件过程，更好地满足构建和支持云计算应用程序和平台的需求。DevOps 在软件开发中具有如下优势：1）DevOps 是把人员、流程、产品进行结合，给用户提供持续价值的一个过程，既涉及人员、流程、工具，也涉及产品；2）通常把 DevOps 这些流程通过一个流水线的方式串联起来，我们称为一个 DevOps 流水线，包括产品的规划跟踪、软件开发、构建测试、产品部署、运维、监控和优化等，其核心目标就是持续给用户交付有价值的产品。

智能软件是一个软件密集型系统。从开发过程上看，智能软件的开发需要遵循软件开发过程。设计和部署质量软件系统，及时满足其使命目标，这一既定原则亦适用于智

能软件。开发团队应遵循现代软件工程实践以及准则，努力按时地提供高质量的功能，设计架构上的可靠性需求，维护智能软件全生命周期。

另一方面，正如其他复杂软件系统，智能软件面临若干可靠性挑战，且其可靠性表现得更不可控。一个重要原因是人工智能和机器学习引起的编程范型和软件开发模式的转变。传统上，软件系统是采用程序代码来编写系统行为的规则。但是，使用机器学习和人工智能技术，大量规则会从训练数据中推理出来。这种编程范型的转变使得智能软件开发存在挑战——智能软件模块之间的交互存在不确定性，对具有智能组件的软件系统行为进行推理变得困难，对智能软件进行测试或验证也存在挑战[167]。

针对上述转变，智能软件开发存在几个重要问题：

1）需要遵循什么过程，才能开发高可靠的人工智能算法/模型？

2）软件开发团队应如何将人工智能模型生命周期（培训、测试、部署、演化等）集成到智能软件的软件过程中？

3）存在哪些新角色、组件和活动，它们如何结合至现有敏捷过程或 DevOps 流程？

需要同时具备软件工程和人工智能两个领域的知识才能回答上述问题。然而，这两个领域存在着认知差异：人工智能社区关注算法设计及其性能特征，而软件工程社区则更关注算法实现和部署。迄今为止，这两个社区还没有得以真正合作。两个领域应共同努力，构建适应于智能软件的软件过程，保障智能软件可靠性。

2.3.1　数据驱动的机器学习工作流

智能模块的开发，常常遵循一个数据驱动的工作流。图 3 展现了微软公司使用的机器学习工作流[168-169]。它与数据科学和数据挖掘中定义的工作流具有共性，其某些阶段面向数据（包括数据收集、清洗和标记），其他阶段面向模型（例如模型需求、特征工程、培训、评估、部署和监视）。

值得注意的是，图 3 中过程是线性的，但是，机器学习工作流是非线性的，包含多个反馈循环：较大反馈箭头表示模型评估和监视可能会循环回任一阶段，较小反馈箭头说明模型训练可以循环回特征工程。工程师需要对所选模型、超参数和数据集频繁迭代和优化。例如，训练数据和实际数据之间存在较大的分布差异，工程师会收集更具代表性的数据并重新执行工作流；如果问题变化或存在更好的算法，工程师会重新审查在第一阶段选择的模型。如果系统是集成的（如包含多个机器学习组件且组件交互复杂），上述工作流会更加复杂。

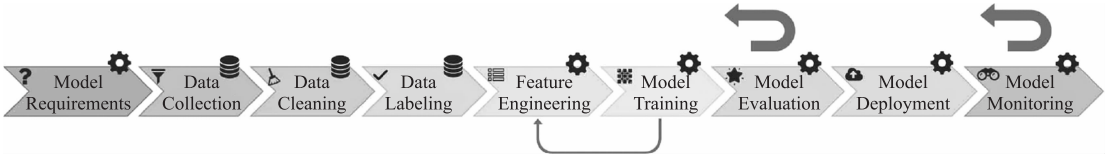


图 3　机器学习工作流程。这个工作流以数据为中心，不同阶段之间存在反馈循环

（1）与数据相关的阶段

流程中与数据相关的阶段包括：1）**数据收集**：团队查找可用数据集（例如，内部或开源）或收集自己数据集。有时候，他们使用通用数据集（例如，ImageNet）训练部分模型，使用迁移学习与专业数据训练来训练出更准确的模型（例如行人检测）。2）**数据清洗**：工程师从数据集中删除不准确或噪声数据。3）**数据标记**：工程师、领域专家或在线众包平台中的众包工作者可以为数据加上标签，大多数受监督的学习技术利用标签来导引模型训练，其他技术（如强化学习）使用演示数据或环境奖励机制来调整模型训练策略。

人工智能算法或机器学习是否成功，一个关键是为算法或学习模型提供足够且可靠的训练数据。数据是智能软件开发具有挑战性的关键方面，影响着系统设计的各个方面。软件元素需要根据数据的结构和行为来进行架构，并且需要根据数据的不确定性、可用性和可伸缩性来明确地进行架构。人工智能系统以在数据上进行训练的模型为中心，如果提供给训练模型的数据发生变化，那么系统的性能可能会下降。这就是数据中心化问题。例如模型在对抗攻击面前性能会非常脆弱，模型的结果可以通过误导性的或对抗的数据来操纵。

现实中，智能算法中需要使用大量数据，这些数据可能是异构的、与上下文相关，而且非常杂乱。搜集、清洗、管理数据和数据版本所需的工作量常常比开发算法、实现功能更为复杂。保障数据可靠性常常需要考虑如下因素：

- **数据规模**。在模型训练中通常需要大数据，大数据可以提升模型的准确性，且可以提高对实时数据预测的准确性。
- **数据质量**。如果数据质量差，则拥有大数据集是没有用的。找到高质量数据对于保障模型精度及确保模型没有偏见至关重要。为此，需要对搜集到的数据进行模型偏差和公平性验证。例如，Raman 等人提出一种数据清洗工具 Potter' s Wheel[170]，支持异常检测和数据不一致清洗所用到的数据变换功能。
- **数据可用**。由于许多机器学习技术要从大数据中学习，因此以机器学习为中心的软件项目能否成功，在很大程度上取决于数据（特别是高质量数据）的可用性。标记数据需要大量人力，因此，需要尽可能重用数据以减少重复性工作。
- **数据安全**。数据的收集管理能力和数据挖掘能力不断提升，如果这些数据被不当或恶意利用，将导致高风险。智能软件面临的数据安全风险可大致分为：1）由于训练数据污染而导致的智能软件失效，例如通过将恶意制造的样本混入训练数据集破坏原数据集的分布和完整性从而导致训练的算法模型出现错误。2）系统运行阶段的异常数据致使智能软件运行错误，例如利用人为构造对抗样本攻击元模型可以使系统产生错误的决策结果或者出现系统异常。3）模型和数据窃取，攻击者对算法模型的输入和输出信息映射关系进行分析，构造出与目标模型相似度非常高的模型，实现算法模型窃取，进一步可以分析出在训练过程中所使用的数据信息。
- **数据融合**。数据集可能包含来自多个不同系统的数据，而所有数据都必须被存储、跟踪和版本控制。当工程师收集和处理这些数据时，他们可以跟踪详细数据信息。

但当项目规模扩大时，保存这些数据及相关来源会成为负担。为此，可以使用电子数据表，以更透明、可靠地跟踪数据集的元数据；为了便于比较数据集，Datadiff工具支持开发人员根据数据示例制定转换函数[171]。

- **数据传输**。数据集规模可能较大，且可能位于不同物理位置。这带来了一些独特的挑战，如数据不容易移动，传输成本高且耗时。由于这些原因，需要使用邻近的数据进行模型训练，避免远程传输或使用大数据集。
- **其他**。上述要求之外，智能软件相关的数据还需要满足以下性质：准确性、权威性、时效性、结构化等。

（2）与模型相关的阶段

流程中与模型相关的阶段包括：1）**模型需求**阶段，设计人员决定可使用机器学习或者人工智能算法来实现智能软件的哪些功能或者问题，他们还决定哪些模型最适合给定功能/问题。2）**特征工程**指为提取模型信息和选择机器学习模型而执行的活动。对于某些模型（如卷积神经网络）而言，此阶段通常与模型训练合并。3）在**模型训练**阶段，需要在收集、清洗的数据集上训练和调整模型（及相关特征）、超参数调优。4）在**模型评估**阶段，工程师使用预定义的指标评估输出模型。这一阶段还可能涉及人工评估。5）在**模型部署**阶段，将模型的推理代码部署在目标设备上。6）在**模型监视**阶段，在实际运行中持续监视智能软件或者模型，发现误差或错误。

为保障模型可靠性，需要开展如下活动：

- **跟踪运行**。工程师跟踪智能软件使用的数据集、模型、代码和超参数的更改，并发现不同运行之间的变化。
- **代码测试与调试**。在初始阶段，程序员所编写的许多代码通常质量不高，无法满足模型训练要求。工程师花费大量时间调试代码，并进行测试。
- **训练和故障排除**。通常需要数小时或数天才能完成模型训练。例如，完整构建特斯拉自动驾驶仪的神经网络需要在GPU上进行7万个小时训练。如果训练过程出错，则需要进行故障排除及继续/重新启动模型训练。
- **模型精度评估**。训练后需要评估模型精度，以查看其是否符合所需的指标。在达到精度要求之前，仍要不断提高准确性。值得注意的是，人工智能的目标不是实现100%的准确性，因为人类在日常任务中也不能达到100%的准确性。人工智能的优势来自从环境不同变化因素中抽象出来的能力，以获得高可靠模型并解决前所未有的情况。
- **再训练**。以机器学习为中心的智能软件会经历由模型更改、参数调整和数据更新启动而导致的频繁更改，这些修订对系统可靠性和性能有显著影响。在数据更新、模型出错或需求变化的情况下，需要重新训练模型。因此，需要设计高效的模型再训练的方法。
- **模型部署**。在模型经过训练并达到精度要求后，它将被部署到生产环境中，并根据模型及其在生产中使用实时数据的方式，进行脱机（批处理）预测或在线（实时）预测。需要在更广泛的设备（如通用设备或智能芯片，甚至IoT、移动设备

等）上测试模型，并在实际部署、运行阶段监控模型，如果发现问题，需要向团队发出警报，以便及时解决问题。

2.3.2 将机器学习工作流集成至智能软件开发过程

随着机器学习组件变得成熟，将机器学习集成到更大的软件系统变得愈来愈重要。但是，由于机器学习模块与传统软件模块具有不同特性，实现系统集成具有很高挑战性。例如，数据驱动学习算法固有的不确定性及隐藏反馈循环回引起复杂组件纠缠，带来大量流程上的变化。在软件工程中已经有一套完整可靠性保障手段（例如，规约、测试、调试等）。然而，机器学习开发具有实验性和迭代性，如何建立统一的、适应智能软件的软件过程，成为难点。

总体而言，设计、部署和维持人工智能系统时，需要工程实践来管理智能算法中存在的不确定性，此外还需要考虑不断涌现的更新。敏捷过程和 DevOps 原则和实践（包括人员、流程和技术）成功地缓解了传统软件可靠性挑战，这些原则和实践及一些机器学习特定实践也成功应用于开发和部署机器学习系统[172]。微软将机器学习的 DevOps 称为 MLOps[173]。

长期以来，软件工程学科一直将软件过程改进视为其重要使命之一。软件工程领域的研究人员和实践者开发了几个著名的评估指标，包括能力成熟度模型（CMM）[174]和六西格玛[175]。CMM 对软件过程进行五个级别的评分——初始级，已管理级，已定义级，量化管理级，优化级。在 CMM 的启发下，微软公司为构建集成机器学习组件的团队设计一个新的成熟度模型[168]。该模型具有六个维度，评估每个工作流阶段是否 1）定义目标，2）一致实现，3）记录，（4）自动化，5）测量和跟踪，6）持续改进。

用于设计智能软件的程序语言、工具和平台在不断发展，为开发可靠和可扩展的系统奠定了基础。当构造智能软件时，我们可以使用更好的工程和数据管理工具。例如，工程师可以利用开源工具，或者构建自己的工具链，以发现、收集、使用、理解和转换数据。工具提供高度自动化，并支持训练、部署模型及产品集成。业界还需要去设计更丰富的机器学习工具、库、模型和框架，满足对于智能软件的开发需求。

此外，软件过程的变化不仅改变了开发团队日常的开发，也影响了团队成员角色。许多软件开发团队包括三个重要角色，包括项目经理、开发人员和测试人员。在 DevOps 敏捷开发中，团队可以将开发人员和测试人员相结合，将运维的角色也集成到软件团队中。近年来，很多软件开发团队添加数据科学家，以提升团队分析应用程序行为、确定 Bug 优先级、估计故障率等能力，数据科学家也承担数据搜集和分析的角色[177-178]。这些数据科学家成为将统计和机器学习工作流集成到软件开发流程中的先行者。随着智能软件发展，数据科学家的角色与深入理解业务问题的领域专家、开发预测模型的建模者以及创建基于云的基础架构的平台构建者等角色相结合。

不同领域专家之间的协作对于开发智能软件非常重要，因此，具有传统软件工程背景的工程师需要学习如何与机器学习专家一起工作。例如，许多团队定期举办关于机器学习和深度学习的开放式论坛，使从业者能够聚在一起，了解有关人工智能的信息，任何人都可以提出和回答有关人工智能和机器学习的技术和实用问题，分享最新学术成果。

2.3.3 高可靠的智能软件开发的建议

尽管众多软件开发准则、实践均适用于智能软件开发，但智能软件仍具有特殊性。为此，文献［179］指出适用于智能软件（及智能组件）开发的11条建议：

1）确保有一个可以，且应该由人工智能来解决的问题。

2）在软件开发团队中需要包括领域专家、数据科学家、数据架构师和人工智能专家。

3）认真对待数据，防止其过多占用项目资源、时间和注意力。

4）基于需要做什么，而不是其受欢迎程度来选择算法模型。

5）通过高强度监测，确保人工智能系统的安全可靠。

6）定义检查点，以满足可恢复、可追溯等的潜在需求。

7）整合用户体验和交互，不断验证和演化模型和架构。

8）解释输出中存在的模糊性。

9）实施松散耦合的解决方案，支持对系统进行扩展或更新，以适应数据、模型和算法更新。

10）为系统生命周期的持久变化投入足够的时间和专门知识。

11）软件设计时考虑道德和政策因素。

3 国内进展分析

3.1 智能软件可靠性模型

3.1.1 模型可靠性

南京大学Feng等人[180]对DeepRED工作进行了改进。DeepRED是一种将具有较好解释性的决策树与解释性差的深度神经网络结合，利用决策树可解释性的优点对神经网络决策进行过程简化，使深度学习网络具有信息提供性的特征。该方法的缺点是成本巨大，需要大量的时间和内存。为解决此问题，Feng等人提出gcForest，通过深度树集成方法，利用级联结构让gcForest进行表征学习，从而解决了DeepRED的成本问题，并同时获得了更好的解释性。

浙江大学的纪守领、李进锋等人[181]回顾了机器学习中的可解释性问题，分析了可解释性与可解释机器学习安全性之间的关系。他们提出，可解释性研究当前面临的两大挑战，第一是如何设计更精确、更友好的解释方法，以消除解释结果与模型真实行为之间的不一致；第二是如何设计更科学、更统一的可解释性评估指标，以评估可解释方法解释性能和安全性。江苏大学的成科扬等人[182]从模型代理、逻辑推理、网络节点关联

分析、传统机器学习模型改进4方面剖析了可解释性深度学习模型构建研究。

南京大学的綦小龙、高阳等人[183]针对基于约束的方法所存在的序依赖、高阶检验等问题，提出通过互信息排序的贝叶斯网络结构学习方法。该方法包括度量信息矩阵学习和“偷懒”启发式策略两部分。其中度量信息矩阵刻画了变量间的依赖程度而且暗含了程度强弱的比较，有效地解决了检验过程中由于变量序导致的误判问题；而“偷懒”启发式策略在度量信息矩阵的指导下有选择地将变量加入条件集中，有效地降低了高阶检验而且减少了检验次数。在不丢失学习结构质量的条件下，该方法的搜索比其他搜索过程显著快而且易扩展到样本量小且稀疏的数据集上。

在人工智能安全理论方面，北京大学的研究者发现，没有进行任何对抗防御的普通模型更加关注局部纹理特征，而经过对抗训练的鲁棒的神经网络则依赖于物体形状和远距离依赖关系来进行分类[114]；清华大学以及北京航空航天大学的研究者则从模型网络中神经元的角度出发来解释深度学习对抗脆弱性的机理[121,176]。

3.1.2 数据可靠性

大规模数据收集令个人隐私保护面临更大的风险与挑战。中国人民大学刘俊旭、孟小峰等人[185]对现有隐私保护研究工作进行梳理和总结，首先分别针对传统机器学习和深度学习两类情况，探讨了集中学习下差分隐私保护的算法设计；在此基础上进一步概述了联邦学习中存在的隐私问题及保护方法；最后总结了目前隐私保护中面临的主要挑战，并着重指出隐私保护与模型可解释性研究、数据透明之间的问题与联系。

数据集内数据可能高度冗余，重复数据删除技术能够极大地缩减数据存储容量需求，提高网络带宽利用率。国防科技大学付印金、肖侬、刘芳等人在文献［186］中针对重复数据删除技术，包括数据划分方法、I/O优化技术、高可靠数据配置策略以及系统可扩展性进行了调研。

北京大学的陈立玮、冯岩松等人[187]针对海量网络数据的信息抽取与应用，提出利用n-gram特征来缓解传统词法特征稀疏性的问题。该方法针对弱监督学习中标注数据不完全可靠的情况，使用基于bootstrapping思想的协同训练方法来对弱监督关系抽取模型进行强化，并且对预测关系时的协同策略进行了详细分析。

数据交换是将符合源模式的数据转换为符合目标模式数据的问题，该过程要求目标模式尽可能准确并且以与各种依赖性一致的方式反映源数据的问题。Xiao等人[188]设计了一种新颖的后端计算架构——数据隐私保护自动化架构（DPA），以促进在线隐私保护处理的自动化，并能够以无中断的方式与公司的主要应用系统无缝集成，同时允许适应灵活的模型和交叉的服务质量保证实体数据交换。随着云计算和Web服务的快速发展，Wu等人[189]将基于特征的数据交换应用于基于云的设计与制造的协作产品开发上，提出一种面向服务的基于云的设计和制造数据交换架构。

3.1.3 平台可靠性

北京大学的张路团队和香港科技大学Shing-Chi Cheung等人[190]针对深度学习系统的

真实程序缺陷开展了首个实证研究。他们通过挖掘 StackOverflow 和 Github 页面中与 Tensorflow 程序缺陷报告相关的信息，对 175 个缺陷进行了深入细致的分析探讨，包括缺陷的症状、根本原因，以及现有检测与处理此类软件缺陷所面临的挑战等。该研究还对 175 个缺陷进行分类，包括了异常或崩溃、低正确率、低效率、未知几大类。其中低正确率问题引起了研究人员的关注，他们发现，175 个缺陷中有 40 个缺陷报告与此有关。而与低效率相关的缺陷则为偶发，175 个缺陷中仅有 9 个报告与此有关。此外，研究将出现缺陷的根本原因进行归纳，包括 API 误用、未对齐张量以及错误的模型参数或结构。Zhang 等人[191]从微软深度学习平台上收集 4960 个失败案例并对深度学习工作程序失败进行实证研究，其中有 2.8% 的工作由于深度学习框架问题导致失败。北京大学与香港科技大学进一步提出针对神经网络架构中由于数值精度导致的错误[199]。与此同时，天津大学王赞、陈俊洁等[198]以模型变换为基础，对智能软件平台进行相关分析，可以有效检测深度学习框架的质量问题。

此外，有学者针对各类软硬件平台构造了可靠性模型。湖州师范大学蒋云良等人[192]针对软件失效时间及其之前的 m 个失效时间数据使用相关向量机进行学习，建立了失效时间之间的内在依赖关系，由此构建新的基于相关向量机的软件可靠性预测模型。在 4 个数据集上的实验结果表明，新模型在预测能力上较基于支持向量机或人工神经网络的软件可靠性预测模型有明显的提高，同时也表明现时失效数据的预测能力比很久之前观测的失效数据更强。

越来越多的智能软件依赖于云。华东师范大学的段文雪、陈铭松等人[193]针对云计算可靠性问题，概述了云计算系统中常见的各种故障，描述了目前云计算中提高可靠性关键的故障管理技术；鉴于故障管理技术的应用会不可避免地增加系统的能耗，该工作还介绍了云计算中可靠性与能耗权衡问题。此外，云也产生了数据损坏和丢失等方面的数据完整性问题。针对这一问题，徐光伟等人[194]提出一种数据验证结果的检测算法以抵御来自不可信验证结果的伪造欺骗攻击，算法通过建立完整性验证证据和不可信检测证据的双证据模式来执行交叉验证，通过完整性验证证据来检测数据的完整性，利用不可信检测证据判定数据验证结果的正确性。

副本技术成为提高系统可靠性的主流技术之一。清华大学郑纬民院士团队基于 Markov 模型，针对多副本存储系统建立了度量系统可靠性的理论模型[195]。该模型能够反应失效检测延迟对系统可靠性的影响。通过该模型可以度量存储系统关键参数如系统规模、副本阶数、单节点容量、单节点平均失效时间、数据对象平均大小、平均修复带宽等对系统可靠性的影响。

3.2 软件的可靠性保障技术手段

3.2.1 测试用例生成技术

在特定领域的测试输入合成方面，为了测试机器翻译系统，北京大学的张路团

队[196]通过改变翻译输入中的单词，来自动生成测试输入。为了生成应该产生一致翻译的输入句子对，他们的方法基于单词嵌入相似度进行了单词替换。该生成方法在生成具有一致翻译的输入对时具有可达99%的较高精度。

针对智能驾驶系统，Zhang等人与南方科技大学张煜群合作[115]将GAN应用于各种天气条件下驾驶场景的测试输入的生成，他们从Udacity挑战数据集及雪景和雨景的YouTube视频中采集图像，并将其输入一个基于DNN的方法来执行图像到图像的转换的模型UNIT进行训练。训练得到的模型可读入整张Udacity提供的图像作为种子，产生转换了天气场景的图像，作为生成的测试输入。Zhou等人与南方科技大学张煜群、香港中文大学Bei Yu合作[201]，提出DeepBillboard用于生成真实世界中的广告牌的对抗性图像，这些广告牌图像可以导致自动驾驶系统发生潜在的转向错误。

为了测试基于音频的深度学习系统，Du等人[143]考虑音频的背景噪声和音量变化，提出专门针对基于RNN的带状态的深度学习系统的测试准则。他们首先将有状态的深度学习系统抽象为一个概率转换系统。在建模的基础上，他们提出基于变化系统的状态和轨迹的准则，以捕捉动态的变化行为，并用于指导带状态的机器学习系统的测试生成。Chan等人[64]为DNC生成了对抗数据样本，以暴露其健壮性方面的不足。

而为了测试图像分类平台在对生物细胞图像进行分类时的表现，Ding等人与中国地质大学张冬梅建立一个生物细胞分类器的测试框架[204]。该框架迭代地生成新图像，并使用蜕变关系进行测试。例如，他们通过增加生物细胞图像中的人工线粒体的数量或变化形状来生成新的图像，这一操作便导致分类结果发生了变化。

在更加通用的测试用例生成技术方面，由清华大学的孙家广院士团队提出的DLFuzz[205]是另一种基于DeepXplore实现的以神经元覆盖率为指导的模糊测试生成工具，DLFuzz旨在生成对抗数据样本。因此，其测试数据生成过程不需要像DeepXplore和TensorFuzz一样使用功能相似的深度学习系统进行交叉引用检查。相反，它只需要对原始输入进行足以改善神经覆盖率但获得与原始输入具有不同的预测结果的最小的变化，就可以找到那些新的输入。对MNIST和ImageNet的初步评估表明，与DeepXplore相比，DLFuzz可以产生135%至584.62%的输入，所需时间减少20.11%。河海大学的Zhang等人[206]发现现有的DNN测试用例生成方法总是生成大量的测试用例，但其中大多数都不能满足测试需求或实际情况。因此，他们提出一个条件引导的对抗生成测试工具CAGTest，用于为DNN生成测试输入，以发现DNN所存在的潜在缺陷。南京邮电大学的Chen等人[207]发现对于不同的输入（例如不同的对抗攻击技术生成的测试用例），DNN神经元的行为模式是不同的。为此，他们提取了DNN在面对不同的对抗攻击技术时的神经元行为模式，作为测试用例选择的指导，以找到比随机选择的方式更有效的DNN测试用例选择方法。

3.2.2 测试用例集优化相关研究

考虑到生成测试输入的计算成本十分昂贵，中国科学院大学张震宇团队建议对那些表示更有效的对抗样本的测试实例进行识别，以达到降低成本的目的[208]。该方法其实

是一种测试优先级排序技术，可根据对噪声的敏感性对测试实例进行排序（对噪声越敏感的实例越有可能产生对抗样本）。

南京大学吕建院士团队则聚焦在 DNN 操作测试中减少测试数据[209]。他们使用基于交叉熵最小化的分布近似技术，提出一种在 DNN 的最后一个隐藏层中由神经元引导的采样技术。该评估是在三个图像数据集（MNIST、Udacity challenge 和 ImageNet）的预训练模型上进行的。结果表明，该方法仅对一半的测试输入进行抽样，就达到了与随机采样相似的精度水平。

北京大学的金芝团队[210]基于 Metropolis-Hastings 抽样的标识符重命名技术提出一种 Metropolis-Hastings-Modifier（MHM）方法，专用于为处理程序代码的深度学习模型生成对抗数据样本以提高模型鲁棒性。在一套程序功能分类基准上的评估结果表明 MHM 在生成对抗源代码样本方面是有效的，将 MHM 生成的对抗样本用于模型训练可以提升被测模型的鲁棒性和性能。

3.2.3 对抗样本生成

1）计算机视觉领域研究进展。Zhao 等人[211]提出一种生成更加自然的对抗样本的方法。他们使用 GAN 学习样本在输入空间中的分布，然后在该分布中寻找相应的对抗样本，以此生成更加自然的对抗样本。作者在图片输入和文本输入上展开了实验，结果表明，该方法生成的对抗样本更加有效。

Xiao 等人[212]提出一种空域变换的对抗样本生成方法 stAdv。传统的对抗样本生成方式都是加扰动，是一种像素值变换，而该文中作者提出一种空间变换生成对抗样本的方法，他们通过做平移、旋转和扭曲局部图像特征等微弱的空间变换来扰动图像。这种方式扰动很小，可以逃避人类的检查，但仍旧可以使分类器失效。与其他对抗样本生成方法相比，该方法有较大的 L_p 距离，但变换方式更真实，且不容易被现有对抗攻击防御方法检测出来。

Dong 等人[213]提出基于 Momentum 的对抗样本生成方法。该方法借鉴物理学的动量思想，在基于迭代的 FGSM 基础上上引入 Momentum，既保证了对抗样本的攻击能力，又提升了对抗样本的迁移性。

目前对抗样本生成方法大多基于 L2 和 L∞ 范数，很少有方法用到 L1 范数。Chen 等人[214]将对抗样本攻击 DNN 的过程转化为使用弹性网络正则化（elastic-net regularized）的优化问题。当前最佳的 L2 范数攻击算法成为该方法的一个特例（在不考虑 L1 范数的情况下）。在 MNIST、CIFAR10 和 ImageNet 上的实验结果表明该方法可以生成具有很小 L1 失真的对抗样本，并且生成的对抗样本有显著增强的攻击可迁移性，在一些针对 L2 和 L∞ 范数防御性很强的模型面前，该方法仍具有较高的攻击性。

Xiao 等人[215]提出利用对抗生成网络（GAN）来生成对抗样本。他们使用生成对抗网络构造对抗样本，一旦训练了生成器，可以为任何测试数据产生有效的对抗性扰动。他们在半白盒和黑盒攻击设置中应用 AdvGAN，与传统的白盒攻击相比，半白盒攻击在生成器训练之后不需要访问原始目标模型。在黑盒攻击中，AdvGAN 动态训练黑盒模型

相应的蒸馏模型并优化其生成器。与其他攻击相比，AdvGAN 在有防御情况下具有较高的攻击成功率。

重庆大学李俊杰等人提出 GAPE 模型[238]，该模型解决基于迭代的生成算法需要多次反向梯度计算的问题。在模型中，原始图像作为生成器输入，通过在原图像上进行图像增强，生成器一次输出对抗样本。该方法使得对原图的修改尽可能被隐藏在原图的边缘等人眼不易察觉的区域。GPAE 模型中还加入了判别器损失、感知损失、分类损失等多损失函数，使对抗样本可以攻击多个分类器。

北京工业大学马玉琨等人提出一种面向人脸活体检测的对抗样本生成算法[239]。该算法基于迭代生成算法，将最小扰动维度和扰动间距约束相结合。算法首先计算给定输入的梯度矩阵，根据梯度矩阵选择幅值最大的维度进行扰动，并不断迭代直至模型输出错误。在该方法中加入扰动点的间距约束，使生成的对抗样本更不易被察觉。

西安邮电大学的王曙燕等人提出了基于 GAN 的黑盒攻击的图像对抗样本生成方法[240]。该方法通过利用原始样本集训练样本生成器，然后利用模型蒸馏方法获得目标模型的复制；再以对抗生成网络的输出作为输入，以获得蒸馏模型作为目标模型，训练新的生成对抗网络，网络输出为加入对抗样本的扰动，最后将样本与扰动相加并以像素灰度值区间进行规范化，得到对抗样本。该方法提升对抗样本多样性，更容易暴露模型缺陷。

阿里巴巴联合美国 UIUC 大学开发了对抗性测试平台 DeepSec[74]，利用对抗样本技术对原始图像实行针对性干扰，降低人工智能模型识别率。

2）自然语言处理领域研究进展。Lei 等人[216]提出一种对抗样本攻击的通用框架，该方法在句子和单词粒度上生成对抗样本。他们首先建立一个包含许多单词和句子释义的语料库，结合梯度信息引导贪婪搜索以寻找最优的有效释义。该方法相比传统的贪婪算法耗时更少，并使用联合句和单词释义技术保证了文本的原始语义和语法。

Liang 等人[217]提出一种基于 FGSM 的文本攻击方法 Textfool。TextFool 采用了 FGSM 的思想，在文本级别上对文本分类任务进行攻击。不同于原始 FGSM，TextFool 考虑代价梯度的大小，找出对结果贡献最大的字符，将其命名为热字符，包含足够多的热字符并且出现频率最高的短语被选为热训练短语（HTPs），再对热短语进行插入、修改删除等操作。

Cheng 等人[218]提出另一中基于梯度的对抗样本生成算法：AdvGen。AdvGen 基于梯度对神经机器翻译（NMT）模型进行攻击，它首先考虑损失函数的梯度之间的相似性，以及单词与替换单词之间的距离，使用语言模型来识别给定单词中最有可能替换的单词并保留语义信息，该方法能够实现非重叠攻击和有针对性的关键词攻击。

Zhang 等人[219]提出一种基于编辑的方法 MHA，旨在提供流畅有效的对敌攻击。MHA 基于语言模型和 Metropolis-Hastings（M-H）抽样。使用 M-H 抽样生成替换旧单词（替换操作）和随机单词（插入操作）的单词。语言模型用于在替换/插入/删除操作之后加强句子的流畅性。他们提出黑盒和白盒攻击两种方法。

Ren 等人[220]提出一种基于概率和单词显著性的方法选择要替换的单词来攻击文本分

类模型。它首先收集现有语料库中单词的所有同义词，然后通过衡量替代词对分类概率的影响，从同义词中选择替代词。该方法可使那些将导致分类结果发生最重大变化的词被选择。

武汉大学王文琦等人提出面向中文文本的情感倾向性分类的对抗样本生成方法 WordHanding[241]。该方法在黑盒模式下，使用一种计算词语重要性的算法，利用同音词替换的方法来生成对抗样本。该方法在黑盒情况下生成对抗样本，对长短记忆网络和卷积神经网络两种 DNN 模型进行攻击。生成的对抗样本能够误导中文文本的倾向性检测系统。

3）其他领域研究进展。Liu 等人[221]提出 PS-GAN 方法，该方法将贴纸张贴张交通路牌上来模拟对抗样本，使自动驾驶系统发生误判。Huang 等人[222]通过在人穿着的衣服上张贴具有攻击性的贴图，使得目标检测系统无法正确识别。

Yuan 等人[223]提出一种实用且系统的对抗性攻击方法 CommanderSong，该方法将一组命令嵌入到一首歌曲中，从而合成包含命令信息的并且可远程传播歌曲。其能够对 iFLYTEK2，Kaldi 等主流 ASR 系统进行黑盒攻击。Xu 等人[224]指出，ASR 系统可能会对用户对话内容进行监控，从而导致用户隐私泄露等安全问题。基于此，他们利用 ASR 系统对于对抗样本的敏感性，在 MFCC 生成的阶段注入扰动，提出针对移动设备恶意 ASR 监控的高性能自适应安全增强方案 HASP。

针对语音识别系统，浙江大学的徐文渊团队[197]发现可以通过调制生成人类无法听到的超声波载波上的语音命令，利用麦克风电路的非线性特性使语音识别系统能够解调、恢复和解释调制后的低频音频命令，实现对 Siri 等语音识别系统的诱骗和攻击。他们同时也提出针对该情况的分别基于硬件和软件防御方案。

山东大学的甘滨与郭山清针对对抗样本脆弱性的问题，设计了面向深度学习模型的安全测试平台[246]。该平台可提供深度学习对抗样本攻击、防御和评估功能。

3.2.4 测试谕言缺失问题解决方案

在近期，武汉大学的谢晓园与南京大学的徐宝文等人[225]针对无监督学习的聚类程序提出一种蜕变验证方法 METTLE。METTLE 从无监督学习软件的用户预期出发，设计了包含六种不同粒度的蜕变关系。这些蜕变关系对实例顺序、显著性、密度、属性进行操作，或者注入数据的异常值。评估中使用 Scikit learn 生成数据，评估结果表明 METTLE 在不同应用领域中对于无监督学习软件的验证是切实有效的。这项研究是该团队对其 2009 和 2011 年[226-227]所提出的针对有监督学习软件进行测试工作的重要延伸与补充。文献［226-227］中提出五种类型的蜕变关系，根据对输入所执行的具体特定变化预测对应输出的预期变化（如：类、标签、属性的变化）。对 Weka 中 KNN 和朴素贝叶斯的实现进行的人工证明表明，并非所有的蜕变关系都是每个分类器所必要服从的。此外，文献［226］明确验证了机器学习软件传统验证方法（如交叉验证）具有很大局限性，而（蜕变）测试则可以在程序已获得极低错误率的情况下，仍然揭示出程序在重要属性上的失效和错误，是传统度量指标的关键补充。

Ding 等人与中国地质大学的 Kang[231] 提出 11 个蜕变关系来测试生物细胞图像的分类的深度学习系统。在数据集级别，蜕变关系为基于不影响真实分类结果的训练数据或测试数据转换，如在训练数据集的每个类别中添加 10% 的训练图像或从数据集中删除一类数据。Pham 等人与中国科学技术大学的 Qi[232] 采用了交叉参考来测试机器学习软件的实现，但其主要集中在深度学习库的实现上。他们提出一个专注于在深度学习软件库中发现和定位缺陷的方法 CRADLE。在三个库（TensorFlow、CNTK 和 Theano）、11 个数据集（包括 ImageNet、MNIST 和 KGS Go game）和 30 个预训练模型上，CRADLE 检测到 104 个独特的不一致的情况和 12 个错误。

大多数的差分测试依赖于多个实现或多个版本，南京大学的吕建院士团队[233] 提出基于程序合成的机器学习系统的功能评估方法 SynEva。为了回答“在机器学习软件训练过程中抽取的知识是否适用于预测过程”这一关键问题，该方法首先基于现有知识构造镜像（类 oracle 结构），确保镜像和现有知识在对机器学习软件的训练过程中有相似的表现。然后，SynEva 对镜像和现有知识在预测过程中的行为进行比较，如果二者相似，则可以对上述关键问题进行肯定回答，反之，则说明训练过程中抽取的知识并不适用于实际环境。该方法通过学习场景的程序合成对机器学习系统进行评估，其具有三个特点：不依赖于测试谕言；无须在相同规范下多次执行；自动执行。

南京大学的陈振宇团队[234] 针对深度学习软件系统测试时面临的谕言缺失问题提出一种基于 DNN 统计视角的测试优先排序技术 DeepGini，将测量误分类概率的问题简化为测量集合杂质的问题，从而在不需要花费大量时间和人力进行样本标注的情况下快速识别可能的误分类测试，对深度学习软件进行测试。同时，他们指出 DeepGini 在效率和效果两方面的表现，在区分测试优先级上都优于现有的基于覆盖率的技术。

3.2.5 测试充分性相关研究

马雷等人[235] 通过扩充神经元覆盖概念，提出综合的 DeepGaunge 度量指标。他们首先根据训练数据描绘一个 DNN，并在此基础上根据训练数据获得每个神经元激活的统计行为。基于此，他们提出更细粒度的标准（k 多段神经元覆盖率、神经元边界覆盖率和强神经元激活覆盖率）来表示 DNN 的主要功能行为和角点行为。此外，他们还提出层次覆盖准则，该准则考虑了最活跃的神经元及其组合（或序列）来表征 DNN 的行为。他们结合基于 MNIST 和 ImageNet 数据集上得到的情况，评价该覆盖的性能，并发现该覆盖具有更好的性能。在后续工作中，他们进一步提出组合测试覆盖率[228]。他们通过检查每层中的神经元激活交互的比例来检查每层神经元的组合激活状态。在 DeepGaunge 基础上，Xie 等人[203] 提出一种基于蜕变变换的覆盖引导模糊化技术 DeepHunter。该技术使用一种更细粒度的蜕变突变策略来生成数据，在降低假阳性情况方面显示出优势，并在实现高覆盖率和缺陷检测能力方面也显示出优势。

南京大学的陈振宇团队[200] 提出一系列路径驱动的测试准则 DeepPath，以全面计算深度神经网络的覆盖率。工作中，他们采用了四种 DNN 模型和四种对抗性攻击技术对 DeepPath 的效能进行了评估，发现使用 DeepPath 测试的方法在充分性方面具有更好的评

估能力，对于识别对抗性攻击测试输入也更有帮助。

针对目前流行的基于 DNN 神经元的覆盖准则，南京大学的马晓星、许畅、曹春等人[202]指出，由于神经网络与人类编写的程序存在根本差异，深度神经网络的结构覆盖准则存在局限性。他们对自然输入的初步实验发现，测试集中错误分类的输入数量与其结构覆盖率之间没有强相关性。由于机器学习系统的黑盒特性，目前对于这些标准如何直接与系统的决策逻辑相关暂时还并不清晰。在此基础上，文献［209］将对深度学习软件测试视作通过统计抽样进行可靠性估计，基于传统结构覆盖的支撑思想是方差减少的条件的认识，提出一种基于被测 DNN 模型学习的表征条件的 DNN 测试方法。该方法利用 DNN 网络训练学习得到的最后一个隐藏层中神经元输出的分布表示来简约测试输入控件，减少测试数据的标注代价。该方法相比于随机选择测试集，能够在大幅减少测试样本数量的情况下达到同样 DNN 测试精确程度。

而另一些学者采用变异测试的方法设计测试充分性准则。与结构覆盖准则相比，基于变异测试的准则更直接地与 DNN 的决策边界相关。例如，在 DNN 的决策边界附近的输入数据可以更容易地检测到 DNN 与其突变体之间的不一致性。马雷等人[158]提出一种在源级或模型级对 DNN 进行变异的 DeepMutation，以对 DNN 的决策边界进行小扰动。在此基础上，突变分数被定义为结果发生变化的测试实例与总实例数的比率。南京大学陈振宇团队[159]提出 DNN 的五种变异算子，并在 MINST 数据集上评价了变异的性质。他们指出针对不同应用领域的机器学习软件需要引入领域特定的变异算子来加强突变分析。此外，他们还提出一种工具 MuSC 对智能合约进行变异测试[160]，MuSC 实现了一系列关于智能合约编程语言 Solidity 的新变异算子，可以快速生成大量突变体，并支持创建测试网、配置和执行测试等自动化操作。

3.2.6　运行时不相关与错误数据分析检测

新加坡孙军与浙江大学王新宇等人[144]利用边界扰动产生的决策一致性，对模型可能进行误判的输入（如对抗样本）进行相关检测。南京大学王慧妍等人[137]提出采用神经网络各个层行为的一致性，对输入预测的不确定性进行分析，可以检测并过滤掉可能的错误输入。

3.2.7　形式化验证技术

随着深度学习技术的快速发展，分类深度神经网络（CDNN）被嵌入到许多安全攸关的软件系统中，因此，对其鲁棒性进行验证显得必要且迫切。为此，何积丰院士团队[127]提出一种使用 Sigmoid 激活函数验证 CDNN 鲁棒性的新方法。基本思路是，将鲁棒性验证问题转化为检测输入区域中最可疑点的等价问题，从而构成非线性优化问题；然后通过将非线性约束转换为线性包含，将其进一步细化为线性规划问题。

相较于其他鲁棒性验证技术普遍采用的计算上近似值的思路，该方法着眼于寻找输入域中比剩余其他点更容易被赋予不同标签的点，并检查该点是否会被错误分类。具体而言，该方法构建了一个等效优化问题，其中，网络里神经元之间的关系被编码为了约

束，目标函数的目的则是找出期望标签输出值与其他标签输出值之间的最小差异。由于期望标签应该具有最大值，因此，当且仅当鲁棒性保持不变时，优化问题的最优值才为正。随后，通过将约束中的非线性 Sigmoid 函数替换为线性包含，优化问题可得以进一步简化，线性规划求解器也由此可以应用到验证过程中。与现有方法相比，该方法主要具有两个优点：一是通过检查输入域中的最差点是否会被予以不同分类来验证系统鲁棒性，从而对 CDNN 鲁棒性进行更加精确的估计；二是可以在确保安全的前提下进行扩展，以适用于现实应用中的大规模网络。

3.3 可靠的智能软件的开发过程

国内针对智能软件的开发过程的研究较少。但是，智能软件开发过程中产生的大数据，可以指导智能软件可靠性提升和发现软件缺陷。梅宏院士指出，软件自动化可能会借助大数据实现突破[184]：首先，可以自动生成各种日常软件开发任务的源代码；其次，自动生成的代码应具有与人工编写的代码相比具有可比性甚至更高的质量。开发大数据为智能软件提供可靠性支持。

另一方面，由北京大学梅宏院士和南京大学吕建院士提出的网构软件（Internetware）这一概念[119]。网构软件能感知外部网络环境的动态变化，并随着这种变化按照功能指标、性能指标和可信性指标等进行静态的调整和动态的演化。存在大量关于网构软件可靠构造的研究成果，包括文献［138-139］。可以预计到，针对引入智能构件的网构软件的可靠性研究，将成为一个引人入胜的研究问题。

此外，有学者开始研发基于人工智能全生命周期的对抗安全评测体系，设计面向人工智能系统的对抗及评测集成框架，实现异构的人工智能模型的自动化评测流程；结合模型对抗安全理论基础，以及对抗安全防护策略，集成对抗训练智能算法加固、对抗样本智能甄别与人工智能缺陷定位技术，构建面向人工智能模型对抗安全的保障框架，提升模型的对抗安全性并给出可解释性建议[243]。其中，清华大学开展了面向人工智能算法和模型的对抗测试技术的研究，提出了 RealSafe 人工智能安全平台，支持多种 AI 算法的漏洞检测与修复。北京航空航天大学搭建了人工智能模型安全评测平台“重明”；融合不同场景、不同任务下的对抗攻击算法并构建评测数据集，形成了面向人工智能系统对抗攻防与评测的资源库。

4 智能软件可靠性的挑战、机遇以及发展方向

近年来，智能软件的可靠性问题日渐成为人们的关注焦点，因为相关软件的缺陷与低可靠性可能会带来极高的风险与损失。目前，已有越来越多的学者指出，软件工程领域中的可靠性保障技术可在智能软件中发挥重要应用，未来也应对此展开持续性的深入研究。本节，我们将对智能软件可靠性研究的未来趋势进行展望。

4.1　智能软件可靠性模型

在智能软件可靠性方面，一些热点的研究问题包括：人工智能系统故障机制；机器学习攻击和威胁模型；人工智能系统可靠性评价方法和指标；人工智能系统行为的正确性和可预测性；机器学习和深度学习的可靠性、鲁棒性和脆弱性；神经网络中的鲁棒性和对抗性扰动。此外，针对领域内智能软件，如自动驾驶仪软件、物联网（IoT）和自然语言界面（NLI）的可靠性、安全性和鲁棒性，也将成为热门的研究主题。针对智能软件可靠性的实证研究和试验平台也将得以进一步发展。

模型可靠性。影响模型可靠性的因素很多，其中最重要因素就是由于模型缺陷导致的模型可靠性下降。智能软件中的缺陷不同于传统软件中的缺陷。在智能软件系统中缺陷可以分为环境缺陷、模型缺陷、数据缺陷，其中模型缺陷会严重影响模型可靠性。模型缺陷指智能软件系统中涉及的算法的缺陷如深度学习梯度下降算法应用时易出现梯度不稳定的现象、代码缺陷如开发者编写程序的缺陷（异常和错误、API 误调用缺陷等）、功能缺陷如模型的准确性、模型功能覆盖程度缺陷等。对应于模型可靠性来说，可能会导致模型功能性变差如准确度降低、覆盖范围变窄；模型可解释性降低，无法对模型预测结果做出合适解释；鲁棒性变差，对于异常情况无法处理；面对故障难以恢复等。目前在有一些工作对各种模型可靠性进行了研究，例如深度神经网络的测试、验证、对抗鲁棒性的研究。然而，对其他可靠性属性还缺少充分的研究。

数据可靠性。数据安全是人工智能安全的关键与核心，也是智能软件可靠性的保障。数据的质量和安全直接影响人工智能算法和模型的准确性。在此方面的研究趋势包括：

1）更加广泛应用领域的可靠性保障技术研究。人工智能已经涉及实际应用中的各方各面，但目前的大多数技术都是基于图像文本，其他数据类型的研究工作非常少，在以后的研究中，研究者可以扩展到其他数据方面。

2）人工智能系统的数据质量与数据安全问题的研究。大数据和算力的提升是人工智能再次全面兴起的重要因素，然而数据的质量与安全性问题也日益凸显。如何应对相关的风险，已成为全世界共同的议题。对于软件工程领域，需考虑如何高效自动地实现数据完整性、准确性、有效性、时效性、一致性、安全性等一系列属性的评估与分析，结合相关法律法规，推进形成更加完善且有针对性的智能软件开发与数据收集框架流程。

3）针对数据泄露问题进行研究。在部署智能软件时常常需要把访问接口公开给用户，恶意攻击者可以通过访问接口对模型进行攻击获得隐私信息，或智能软件使用不当可能导致数据泄露等。如何结合软件工程技术和数据安全技术、网络安全技术以保障数据安全，是智能软件可靠性方面的重要研究问题。

人工智能平台可靠性。平台可靠性包括软件平台可靠性和硬件平台可靠性。软件平台以深度学习框架为代表，是人工智能实现飞跃式发展的重要突破口，是人工智能深度学习框架为开发者提供了进行深度学习的底层架构和接口，封装了大量的神经网络模型，可以在很大程度上减少开发者编程的时间和精力，提高深度学习效率，为编程人员提供

开发便利，减轻编程人员的负担。如何开发出 API 友好、调试简单、执行高效（包括计算效率、数据预处理效率和分布式训练效率等）、全场景支持的深度学习框架目前仍是一个挑战性的问题。

另一个值得注意的研究趋势是对人工智能硬件平台可靠性进行研究。人工智能需要大量计算，对算力提出更高要求。例如，卷积神经网络（CNN）需要运算大量卷积运算。为此，业界常常设计出自己的人工智能芯片，进行计算加速。人工智能芯片主要包括 NVidia GPU、Google 的 TPU、Intel 的 Nervana[142]、IBM 的 TreueNorth[135]、微软的 DPU 和 BrainWave、百度的 XPU、Xilinx 的 xDNN、寒武纪芯片、地平线以及深鉴科技的 AI 芯片等。深度学习硬件计算平台被广泛应用，但这些硬件平台都存在着安全隐患，例如硬件生成制造过程中，攻击者可利用不可信的第三方 IP、工具、人员进行恶意攻击等。保障平台可靠性是保障智能软件可靠的基础，迄今为止对人工智能平台的可靠性较为不足，需要更多关注。

为缩小智能算法与硬件后端的差距，人工智能或机器学习框架中还提供诸多优化技术，例如：1）TensorFlow 的 XLA 会把高级代码抽象成 XLA HLO 表示，做目标无关优化之后再用对应后端来生成更深层的代码；2）NVIDIA 的 TensorRT 是在图转化之后的统一表示上进行优化，如根据设定好的规则来对一些相邻计算单元合并（Kernel Fusion）等；3）TVM 是一个用于 CPU、GPU 和专用加速器的开放式深度学习编译器堆栈[136]，它将 Keras、MXNet、PyTorch、Tensorflow、CoreML、DarkNet 中的深度学习模型汇编成各种硬件后端的最小可部署模块，并实现计算图优化，而基础架构可在更多后端自动生成和优化张量运算符。保障人工智能算法的部署或优化的可靠性成为一个重要的挑战——如何确保上层模型能被正确地编译到底层的代码指令，如何保障优化的正确性，需要开展更多研究工作。

可靠性度量与评测体系。可靠性度量、评测体系与评测工具的相关研究与开发仍处于探索阶段。比如，在数据方面，需要提供可靠性度量手段，以检测训练智能软件的数据集满足质量标准。类似的，在模型方面，已经提出多种基于神经元的测试充分性度量指标，然而目前方法仍存在诸多不足与缺陷，需要建立有效的可靠性度量方法以及评测体系。此外，智能软件平台通常设计与实现较为复杂，如何评估并量化其可靠性也是未来重要研究点。

4.2 智能软件可靠性保障技术手段

智能软件通常依赖于深度神经网络学习算法，具有鲜明的黑箱属性，即不可解释性。其训练与测试依赖于海量数据，但仍然具有泛化性差、鲁棒性差、结果不稳定等问题。此外，大量对抗样本的出现也揭示了此类软件在安全性方面的诸多隐患。对人工智能算法和模型进行质量与安全性评估，尽早发现算法中潜在的问题，是保障智能软件可靠性的关键所在。现阶段的智能软件测试仍然处于起步阶段，有诸多亟待解决的问题。在当前已有的研究工作之上，研究者们可以做进一步的研究，具体包括：

1）更高隐蔽性的对抗样本生成技术研究。当前的对抗样本生成技术都是基于原模型

的，因此，对原模型来说，很容易暴露对抗样本的性质，而在实际中原模型对异样样本的性质是未知的。因此，需要研究隐蔽性更高的对抗样本生成技术，用来更好地模拟实际中的异常。

2）验证与测试的有机结合。在智能软件质量保障中，也要重视对深度神经网络的验证，如何将验证和测试有效结合，进一步保证系统可靠性，是目前急需解决的问题。

3）对于可靠性保障技术泛化性的研究。目前基于对抗样本的系统安全性检测方法都是基于本模型，对新模型的检测需要重新设计，不具备实际应用意义。目前迫切需要一套通用检测机制，可以对新出训练好模型进行检测。

4）出台人工智能可靠性测试标准，用以指导商业级别的智能软件测试活动。标准应涵盖（但不局限于）准确性、复杂度、可信度、可用性、泛化能力、性能、可解释性、不确定性、鲁棒性等一系列标准，此外，需进一步完善人工智能测试的工作流程，形成系统性的与传统测试量级相当的测试充分性准则及变异测试算子。

5）当检测出可能存在问题时，如何进行自动调试与错误定位变得十分关键。智能软件通常比较复杂，且难于解释。需要进一步研究自动分析、调试手段，以极大提升智能软件质量与开发效率。

6）开发测试、攻击、防御和评估算法的集成支撑平台。虽然存在一些研究提供可靠性评估和保障手段[244-245,247]，但还需要更多平台方面的工作，包括：在知识资源层融合对抗攻击算法、对抗防御算法、评测数据集、多种人工智能模型及开源代码库资源；利用高复用性和高可扩展性的封装方法，实现平台中算法、模型、数据集资源的保存、扩展和关联；开发相应工具集[243]，如人工智能模型对抗安全性评测工具、对抗样本检测与甄别工具、模型缺陷分析和定位工具，模型安全防护与结构优化工具。

我们认为，未来相关测试平台的发展趋势将包括以下几方面：

1）高度自动化：不局限于使用工具执行测试，更应涵盖测试目标定位、测试分析、测试设计、质量度量、测试过程监控与评估和测试结果评估等全流程的自动化，让自动化无处不在，覆盖测试的各个方面。而且，应借助自动化测试流程的改进、自动化基础设施的夯实，持续提高自动化测试的成熟度。

2）云化：测试的基础设施采用当今的虚拟机、容器技术，这使测试环境更容易维护、系统更容易部署，从而更好地支持自动化测试，有利于整合测试资源、提高资源使用的效率，降低企业成本。而且，可以更好地融合开发、测试和运维，收集更多的研发数据，更好地支持测试服务化、智能化。

3）服务化：让软件测试成为一种服务，就是让所有的测试能力可以通过 API 来实现，构建测试中台。例如腾讯公司 WeTest 事业部已经建立较完整的测试中台服务，任何研发人员均可以按需自动获取测试的能力，这样也使研发人员愿意做更多的测试，实现测试左移。

4）智能化：让人工智能技术服务于智能软件测试，通过构建统一的代码库，把所有的代码放在一起更有利于机器学习，使人工智能能更好地发挥作用，包括测试数据的自动生成、自主操控软件、缺陷和日志的智能分析、优化测试分析与设计等。

4.3 可靠的智能软件开发过程

对可靠的智能软件开发过程，存在如下几个研究趋势：

1）**可靠智能软件的架构与设计**。对于大型软件，应对其进行分层并使用模块化思想予以开发。首先，模块被分离和隔离，模块交互由编程接口（API）控制，可以确保当前开发的组件不会干扰其他正在开发中组件的行为。其次，模块化开发使软件开发团队能够高效地被组织——单独的模块被分配给单独的团队/个人；API 使软件模块保持分离，以最大限度减少开发人员之间交互协作。

在智能软件开发中，模块化开发变得困难：在机器学习模块之间保持严格边界更加困难；软件模块与机器学习模型以复杂的方式“纠缠”，它们在训练和调优期间相互影响，即使构建这些模块/模型的软件团队难以将它们彼此隔离。这种纠缠现象可能导致错误传播，意味着工程人员对系统某些某块的改进甚至会降低整体系统可靠性。因此，即使团队单独构建了每个模块，团队之间还是必须密切合作，以便正确训练模型和提升整个软件系统可靠性。因此，需要提供可靠的智能软件的架构与设计方法和技术。

2）**高可靠模型的定制和重用**。软件工程师需要花费大量精力定制和重用软件代码，在软件中，可重用的主要单元是函数、算法、库和模块。软件工程师可以从 Github 上搜索到库的源代码，分叉它，并轻松地对代码进行更改，以开发自己软件的类似功能。

在机器学习中，定制和重用需要更多工作量。一个主要原因是机器学习模型不易扩展。如果工程师将模型应用于与最初训练数据类似的领域中，重用模型非常简单。但是，当需要在不同领域上运行模型或使用不同输入格式时，则需要更多的工作量，而不是简单地更改机器学习模型参数。例如，不能将英文的 NLP 模型直接应用到中文上。可能需要再训练，或者重新选择智能软件中使用的模型。如何实现高可靠模型定制和重用成为智能软件的软件过程中的关键问题。

3）**数据更新和碎片化**。为获取高可靠的数据集以用于模型训练，在机器学习的工作流中，需要解决数据更新中出现的碎片化问题：

- **数据更新**。由于机器学习中涉及的快速迭代，数据框架（和数据）经常得以更改，甚至每天会被多次更改。数据的持续变更可能来自工程师自己发起的操作，也可能来自传入的新数据（例如传感器、用户交互）。然而，即使对数据集进行微调，也会对模型/智能算法可靠性产生重大影响。数据版本控制是一个技术手段——对训练运行期间使用的数据集建立版本并进行版本控制，支持数据科学家能够灵活地回到数据集的早期版本并进行分析。
- **数据碎片化**。有必要将数据管理工具与机器学习框架混合，以避免数据和模型管理活动的碎片化。可以采用数据版本控制和共享技术解决数据碎片化问题。例如，每个数据集都标有其来源；每个模型都标有来源，标记该模型版本已训练了哪些数据。

4）**模型跨平台部署**。模型的训练过程通常是在有 GPU 等计算加速硬件的高性能的服务器上进行的。训练好的模型在实际的应用部署过程中，通常会面临硬件与软件平台

等多方面的差异。现有的深度学习模型甚至在多训练框架下（比如：Tensorflow 模型转为 PyTorch 模型）都不能够完全兼容。因此，设计相应的通用方法可以使得模型在多个学习平台迁移是一个研究方向。此外，随着越来越多小型系统如 IoT，手机移动端等对深度学习模型的需求，如何将复杂而高精度的模型经过优化转换为在有限硬件资源支持的小型设备上可以高效运行将是重要的一个方向。

5）**自动化**。自动化是另一个至关重要的问题，支持团队能够高效地聚合、提取、标记数据，支持实时数据处理且加快模型训练速度。又如，机器学习工作负载具有某些特殊的硬件设备要求，如 GPU 和高密度内核。与 CPU 相比，GPU 同时运行数千个处理内核，训练和预测运行速度要快得多。由于这些硬件设备成本高昂，并且经常在训练中被使用，因此利用云来支持扩展以及配置的自动化成为一个可行方案。需要加强开发过程的自动化。

6）**可解释**。在软件开发的多个阶段，提供有效的可解释性方法与工具支持，提升智能软件的开发效率，并在开发中实现质量反馈。目前研究虽然在可解释性上有一些进展，然而还需要对智能软件开发过程中智能行为可解释性进行系统化研究。

5 结束语

近年来，随着数据爆炸式地增长以及硬件算力的大幅增强，人工智能研究取得了重大进展，其相关技术通过智能软件的方式得到了广泛的应用和推广。基于深度学习技术的智能软件系统正渗透到现代社会的各个领域，例如图像分类、自然语言处理、双人博弈、自动驾驶系统、智能医疗诊断系统等。在这样的背景下，智能软件的质量与可靠性问题日渐成为人们的关注焦点。

智能软件需要一个强有力的可靠性保障体系与工具支撑。从软件过程上来看，智能软件需要从项目开始时采取不同的心态——由于 AI 是一个软件组件，我们需要将相关的软件可靠性属性应用于它；AI 具有一些特殊的质量属性（公平性、可解释性等），这些属性也必须集成到软件开发和可靠性管理中。另一方面，尽管全球范围内对智能软件的测试和验证做出很多的努力，但相关的可靠性技术依旧是零散和不完整的。为了满足实际工业软件生产需求，工业界和学术界需要建立一个满足智能软件的可靠性系统框架与方法。在此，更希望呼吁更多的中国企业、工程师与学者投身研究智能软件及其基础设施可靠性，推动可靠智能软件的发展。

参考文献

[1] Arthur L. Samuel: Some Studies in Machine Learning Using the Game of Checkers[J]. IBM J. Res. Dev. 3(3): 210-229(1959).

[2] Yann LeCun, Yoshua Bengio, Geoffrey E. Hinton: Deep learning[J]. Nat. 521(7553): 436-444 (2015).

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng: TensorFlow: A System for Large-Scale Machine Learning [C]. OSDI 2016: 265-283.

[4] M. Salvaris, D. Dean, and W. H. Tok, Microsoft AI Platform, in Deep Learning with Azure[M]. Springer, 2018: 79-98.

[5] https://www.predictiveanalyticstoday.com/artificial-intelligence-platforms/.

[6] https://www.paddlepaddle.org.cn/.

[7] https://www.mindspore.cn/.

[8] https://www.transwarp.cn/transwarp/product-sophon.html? categoryId=19.

[9] https://megengine.org.cn/.

[10] https://github.com/Angel-ML/angel.

[11] https://cn.fedai.org/.

[12] http://www.tengine.org.cn/.

[13] Tim Menzies: The Five Laws of SE for AI[C]. IEEE Softw. 2020, 37(1): 81-85.

[14] Lena Pons, Ipek Ozkaya: Priority Quality Attributes for Engineering AI-enabled Systems[C]. CoRR abs/1911.02912(2019).

[15] http://tech.sina.com.cn/roll/2017-03-26/doc-ifycspxn9900655.shtml.

[16] Samuel Greengard: Will deepfakes do deep damage? Commun[C]. ACM 63(1): 17-19(2020).

[17] Jonathan Heawood: Pseudo-public political speech: Democratic implications of the Cambridge Analytica scandal[C]. Information Polity, 2018, 23(4): 429-434.

[18] https://new.qq.com/omn/20181221/20181221A0EHUZ.html.

[19] https://www.freebuf.com/column/225216.html.

[20] Debra S. Herrmann: Software safety and reliability-techniques, approaches, and standards of key industrial sectors[C]. IEEE 1999, ISBN 978-0-7695-0299-1, pp. I-XVI, 1-503.

[21] Qianyu Guo, Sen Chen, Xiaofei Xie, Lei Ma, Qiang Hu, Hongtao Liu, Yang Liu, Jianjun Zhao, Xiaohong Li: An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms[C]. ASE 2019: 810-822.

[22] Brunno F Goldstein, Sudarshan Srinivasan, Dipankar Das, Kunal Banerjee, Leandro Santiago de Araújo, Victor C Ferreira, Alexandre Solon Nery, Sandip Kundu, Felipe M G França: Reliability Evaluation of Compressed Deep Learning Models[C]. LASCAS 2020: 1-5.

[23] Gustav Mårtensson, Daniel Ferreira, Tobias Granberg, Lena Cavallin, Ketil Oppedal, Alessandro Padovani, Irena Rektorová, Laura Bonanni, Matteo Pardini, Milica Kramberger, John-Paul Taylor, Jakub Hort, Jón Snædal, Jaime Kulisevsky, Frédéric Blanc, Angelo Antonini, Patrizia Mecocci, Bruno Vellas, Magda Tsolaki, Iwona Kloszewska, Hilkka Soininen, Simon Lovestone, Andrew Simmons, Dag Aarsland, Eric Westman: The reliability of a deep learning model in clinical out-of-distribution MRI data: a multicohort study[C]. CoRR abs/1911.00515(2019).

[24] Muhammad Abdullah Hanif, Faiq Khalid, Rachmad Vidya Wicaksana Putra, Semeen Rehman, Muhammad

Shafique: Robust Machine Learning Systems: Reliability and Security for Deep Neural Networks[C]. IOLTS 2018: 257-260.

[25] Mukund Sundararajan, Ankur Taly, Qiqi Yan: Axiomatic Attribution for Deep Networks[C]. ICML 2017: 3319-3328.

[26] Marco Túlio Ribeiro, Sameer Singh, Carlos Guestrin: Anchors: High-Precision Model-Agnostic Explanations[C]. AAAI 2018: 1527-1535.

[27] Ravid Shwartz-Ziv, Naftali Tishby: Opening the Black Box of Deep Neural Networks via Information[C]. CoRR abs/1703.00810(2017).

[28] Marco Túlio Ribeiro, Sameer Singh, Carlos Guestrin: Why Should I Trust You?: Explaining the Predictions of Any Classifier[C]. HLT-NAACL Demos 2016: 97-101.

[29] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, Rory Sayres: Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)[C]. ICML 2018: 2673-2682.

[30] Ranjan Kumar, Subhash Kumar, Sanjay K Tiwari: A study of software reliability on big data open source software[C]. Int. J. Syst. Assur. Eng. Manag. 10(2): 242-250(2019).

[31] Yang Xiang, Ivan Stojmenovic, Peter Mueller, Jun Zhang: Security and reliability in big data[C]. Concurr. Comput. Pract. Exp. 28(3): 581-582(2016).

[32] Fangna Tao, Liangxiao Jiang, Chaoqun Li: Label similarity-based weighted soft majority voting and pairing for crowdsourcing[C]. Knowl. Inf. Syst. 62(7): 2521-2538(2020).

[33] Rahul Gupta, Kartik Audhkhasi, Zach Jacokes, Agata Rozga, Shrikanth S. Narayanan: Modeling Multiple Time Series Annotations as Noisy Distortions of the Ground Truth: An Expectation-Maximization Approach[J]. IEEE Trans. Affect. Comput. 9(1): 76-89(2018).

[34] Vikas C Raykar, Shipeng Yu, Linda H. Zhao, Anna K. Jerebko, Charles Florin, Gerardo Hermosillo Valadez, Luca Bogoni, Linda Moy: Supervised learning from multiple experts: whom to trust when everyone lies a bit[C]. ICML 2009: 889-896.

[35] Vikas C Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, Linda Moy: Learning From Crowds[C]. J. Mach. Learn. Res. 11: 1297-1322(2010).

[36] Gianluca Demartini, Djellel Eddine Difallah, Philippe Cudré-Mauroux: ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking[C]. WWW 2012: 469-478.

[37] Jing Zhang, Victor S Sheng, Jian Wu, Xindong Wu: Multi-Class Ground Truth Inference in Crowdsourcing with Clustering[J]. IEEE Trans. Knowl. Data Eng. 28(4): 1080-1085(2016).

[38] Ramakrishna Vedantam, C Lawrence Zitnick, Devi Parikh: CIDEr: Consensus-based image description evaluation[C]. CVPR 2015: 4566-4575.

[39] Peter Anderson, Basura Fernando, Mark Johnson, Stephen Gould: SPICE: Semantic Propositional Image Caption Evaluation[C]. ECCV (5) 2016: 382-398.

[40] Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, Serge J. Belongie: Learning to Evaluate Image Captioning[C]. CVPR 2018: 5804-5812.

[41] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu: Bleu: a Method for Automatic Evaluation of Machine Translation[C]. ACL 2002: 311-318.

[42] Alon Lavie, Abhaya Agarwal: METEOR: An Automatic Metric for MT Evaluation with High Levels of

Correlation with Human Judgments[C]. WMT@ ACL 2007: 228-231.

[43] Lin C ROUGE: A package for automatic evaluation of summaries. In: Proc. of the Meeting of the Association for Computational Linguistics[C]. 2004. 74-81.

[44] Nicholas Ruiz, Marcello Federico: Phonetically-oriented word error alignment for speech recognition error analysis in speech translation[C]. ASRU 2015: 296-302.

[45] Agnese Augello, Alfredo Cuzzocrea, Giovanni Pilato, Carmelo Spiccia, Giorgio Vassallo: An Innovative Similarity Measure for Sentence Plagiarism Detection[J]. ICCSA (5) 2016: 552-566.

[46] Basemah Alshemali, Jugal Kalita: Improving the Reliability of Deep Neural Networks in NLP: A Review. Knowl[C]. Based Syst. 191: 105210(2020).

[47] Fernando Fernandes dos Santos, Pedro Foletto Pimenta, Caio B. Lunardi, Lucas Draghetti, Luigi Carro, David R. Kaeli, Paolo Rech: Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs[J]. IEEE Trans. Reliab. 68(2): 663-677(2019).

[48] Boyang Du, Sarah Azimi, Corrado De Sio, Ludovica Bozzoli, Luca Sterpone: On the Reliability of Convolutional Neural Network Implementation on SRAM-based FPGA[C]. DFT 2019: 1-6.

[49] Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana: DeepXplore: Automated Whitebox Testing of Deep Learning Systems[C]. SOSP 2017: 1-18.

[50] Yuchi Tian, Kexin Pei, Suman Jana, Baishakhi Ray: DeepTest: automated testing of deep-neural-network-driven autonomous cars[C]. ICSE 2018: 303-314.

[51] M R I Rabin, K Wang, and M A Alipour, Testing neural program analyzers[C], in ASE (poster track), 2019.

[52] Uri Alon, Meital Zilberstein, Omer Levy, Eran Yahav: code2vec: learning distributed representations of code. Proc. ACM Program. Lang[C]. 3(POPL): 40: 1-40: 29(2019).

[53] Phil McMinn: Search-based software test data generation: a survey[C]. Softw. Test. Verification Reliab. 14(2): 105-156(2004).

[54] Mark Harman, Bryan F Jones: Search-based software engineering[C]. Inf. Softw. Technol. 43(14): 833-839(2001).

[55] Kiran Lakhotia, Mark Harman, Phil McMinn: A multi-objective approach to search-based test data generation[C]. GECCO 2007: 1098-1105.

[56] Augustus Odena, Ian J Goodfellow: TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing[C]. CoRR abs/1807.10875(2018).

[57] Matthew Wicker, Xiaowei Huang, Marta Kwiatkowska: Feature-Guided Black-Box Safety Testing of Deep Neural Networks[J]. TACAS (1) 2018: 408-426.

[58] Jonathan Uesato, Ananya Kumar, Csaba Szepesvári, Tom Erez, Avraham Ruderman, Keith Anderson, Krishnamurthy (Dj) Dvijotham, Nicolas Heess, Pushmeet Kohli: Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures[C]. ICLR (Poster) 2019.

[59] Zhi Quan Zhou, Liqun Sun: Metamorphic testing of driverless cars. Commun[J]. ACM 62(3): 61-67 (2019).

[60] Saurabh Jha, Subho S Banerjee, Timothy Tsai, Siva Kumar Sastry Hari, Michael B. Sullivan, Zbigniew T Kalbarczyk, Stephen W Keckler, Ravishankar K Iyer: ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection[C]. DSN 2019: 112-124.

[61] Sakshi Udeshi, Sudipta Chattopadhyay: Grammar Based Directed Testing of Machine Learning Systems

[C]. CoRR abs/1902.10027(2019).

[62] Yixin Nie, Yicheng Wang, Mohit Bansal: Analyzing Compositionality-Sensitivity of NLI Models[C]. CoRR abs/1811.07033(2018).

[63] Haohan Wang, Da Sun, Eric P. Xing: What If We Simply Swap the Two Text Fragments? A Straightforward yet Effective Way to Test the Robustness of Methods to Confounding Signals in Nature Language Inference Tasks[C]. CoRR abs/1809.02719(2018).

[64] Alvin Chan, Lei Ma, Felix Juefei-Xu, Xiaofei Xie, Yang Liu, Yew-Soon Ong: Metamorphic Relation Based Adversarial Attacks on Differentiable Neural Computer[C]. CoRR abs/1809.02444(2018).

[65] Sakshi Udeshi, Pryanshu Arora, Sudipta Chattopadhyay: Automated directed fairness testing[C]. ASE 2018: 98-108.

[66] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, James Kapinski: Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components[C]. Intelligent Vehicles Symposium 2018: 1555-1562.

[67] Hartman, Software and hardware testing using combinatorial covering suites, in Graph theory, combinatorics and algorithms[M], Springer, 2005, pp.237-266.

[68] S Kirkpatrick, C D Gelatt, and M P Vecchi, Optimization by simulated annealing[C], science, vol.220, no.4598, pp.671-680, 1983.

[69] James C King: Symbolic Execution and Program Testing[J]. Commun. ACM 19(7): 385-394(1976).

[70] Arvind Ramanathan, Laura L. Pullum, Faraz Hussain, Dwaipayan Chakrabarty, Sumit Kumar Jha: Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision[C]. DATE 2016: 786-791.

[71] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S. Pasareanu, Sarfraz Khurshid: Symbolic Execution for Deep Neural Networks[C]. CoRR abs/1807.10439(2018).

[72] Md Johirul Islam, Giang Nguyen, Rangeet Pan, Hridesh Rajan: A comprehensive study on deep learning bug characteristics[C]. ESEC/SIGSOFT FSE 2019: 510-520.

[73] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, Diptikalyan Saha: Automated Test Generation to Detect Individual Discrimination in AI Models[C]. CoRR abs/1809.03260(2018).

[74] Ling X, Ji S, Zou J, Wang J, Wu C, Li B, Wang T Deepsec: A uniform platform for security analysis of deep learning model[C]. 2019 IEEE Symposium on Security and Privacy(SP). IEEE, 2019: 673-690.

[75] Sainyam Galhotra, Yuriy Brun, Alexandra Meliou: Fairness testing: testing software for discrimination [C]. ESEC/SIGSOFT FSE 2017: 498-510.

[76] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, Daniel Kroening: Concolic testing for deep neural networks[C]. ASE 2018: 109-119.

[77] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, Ananth Grama: MODE: automated neural network model debugging via state differential analysis and input selection[C]. ESEC/SIGSOFT FSE 2018: 175-186.

[78] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, Rob Fergus: Intriguing properties of neural networks[C]. ICLR (Poster) 2014.

[79] Ian J Goodfellow, Jonathon Shlens, Christian Szegedy: Explaining and Harnessing Adversarial Examples [C]. ICLR (Poster) 2015.

[80] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard: DeepFool: A Simple and Accurate

Method to Fool Deep Neural Networks[C]. CVPR 2016: 2574-2582.

[81] A Kurakin, I J Goodfellow, S Bengio, Adversarial examples in the physical world, CoRR abs/1607.02533 [J]. arXiv: 1607.02533.

[82] N Papernot, P D McDaniel, S Jha, M Fredrikson, Z B Celik, A Swami, The limitations of deep learning in adversarial settings[C], in: IEEE, 2016, pp. 372-387.

[83] N Carlini, D A Wagner, Towards evaluating the robustness of neural networks[C], in: IEEE, IEEE Computer Society, 2017, pp. 39-57.

[84] W Brendel, J Rauber, M Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models[C], CoRR abs/1712.04248. arXiv: 1712.04248.

[85] M Wicker, X Huang, M Kwiatkowska, Feature-guided black-box safety testing of deep neural networks [C], in TACAS, Vol. 10805 of Lecture Notes in Computer Science, Springer, 2018, pp. 408-426.

[86] N Papernot, P D McDaniel, I J Goodfellow, S Jha, Z B Celik, A Swami, Practicalblack-box attacks against machine learning[C], in AsiaCCS, ACM, 2017, pp. 506-519.

[87] P Chen, H Zhang, Y Sharma, J Yi, C Hsieh, ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models[C], in AISec@CCS, ACM, 2017, pp. 15-26.

[88] N Papernot, P D McDaniel, A Swami, R E Harang, Crafting adversarial input sequences for recurrent neural networks[C], in MILCOM, IEEE, 2016, pp. 49-54.

[89] S Samanta, S Mehta, Towards crafting text adversarial samples, CoRR abs/1707.02812[J]. arXiv: 1707.02812.

[90] M Sato, J Suzuki, H Shindo, Y Matsumoto, Interpretable adversarial perturbation in input embedding space for text[C], in: J. Lang (Ed.), IJCAI, ijcai.org, 2018, pp. 4323-4330.

[91] J Gao, J Lanchantin, M L Soffa, Y. Qi, Black-box generation of adversarial text sequences to evade deep learning classifiers[C], in: 2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018, IEEE Computer Society, 2018, pp. 50-56.

[92] J Ebrahimi, A Rao, D Lowd, D Dou, Hotflip: White-box adversarial examples for text classification, in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics[C], ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers, Association for Computational Linguistics, 2018, pp. 31-36.

[93] J Ebrahimi, D Lowd, D Dou, On adversarial examples for character-level neural machine translation, in Proceedings of the 27th International Conference on Computational Linguistics[C], COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018, Association for Computational Linguistics, 2018, pp. 653-663.

[94] Y Gil, Y Chai, O Gorodissky, J. Berant, White-to-black: Efficient distillation of black-box adversarial attacks, in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies[C], NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 1373-1379.

[95] M Alzantot, Y Sharma, A Elgohary, B Ho, M B Srivastava, K Chang, Generating natural language adversarial examples, in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing[C], Brussels, Belgium, October 31-November 4, 2018, Association for Computational Linguistics, 2018, pp. 2890-2896.

[96] R Jia, P Liang, Adversarial examples for evaluating reading comprehension systems, in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing[C], EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, Association for Computational Linguistics, 2017, pp. 2021-2031.

[97] Y Belinkov, Y Bisk, Synthetic and natural noise both break neural machine translation, in: 6th International Conference on Learning Representations[C], ICLR 2018, Vancouver, BC, Canada, April 30-May 3, 2018, Conference Track Proceedings, OpenReview. net, 2018.

[98] K Eykholt, I Evtimov, E Fernandes, B Li, A Rahmati, C Xiao, A Prakash, T Kohno, D Song, Robust physical-world attacks on deep learning models[C]. 2017.

[99] S Thys, W V Ranst, T. Goedemé, Fooling automated surveillance cameras: Adversarial patches to attack person detection[C], in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019, Computer Vision Foundation/IEEE, 2019, pp. 49-55.

[100] Z Wu, S Lim, L Davis, T Goldstein, Making an invisibility cloak: Real world adversarial attacks on object detectors, CoRR abs/1910.14667[J]. arXiv: 1910.14667.

[101] T Vaidya, Y Zhang, M Sherr, C. Shields, Cocaine noodles: Exploiting the gap between human and machine speech recognition, in 9th USENIX Workshop on Offensive Technologies[C], WOOT '15, Washington, DC, USA, August 10-11, 2015, USENIX Association, 2015.

[102] L Song, P Mittal, Inaudible voice commands, CoRR abs/1708.07238[J]. arXiv: 1708.07238.

[103] E T Barr, M Harman, P McMinn, M Shahbaz, S Yoo, The oracle problem in software testing: A survey [J], IEEE Trans. Software Eng. 41(5)(2015) 507-525.

[104] T Y Chen, S C Cheung, S Yiu, Metamorphic testing: A new approach for generating next test cases [C], CoRR abs/2002.12543.

[105] C Murphy, G E Kaiser, L Hu, L Wu, Properties of machine learning applications for use in metamorphic testing, in: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008)[C], San Francisco, CA, USA, July 1-3, 2008, Knowledge Systems Institute Graduate School, 2008, pp. 867-872.

[106] C Murphy, K Shen, G E Kaiser, Using JML runtime assertion checking to automate metamorphic testing in applications without test oracles, in: Second International Conference on Software Testing Verification and Validation[C], ICST 2009, Denver, Colorado, USA, April 1-4, 2009, IEEE Computer Society, 2009, pp. 436-445.

[107] S Nakajima, Generalized oracle for testing machine learning computer programs, in Software Engineering and Formal Methods-SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, September 4-5, 2017, Revised Selected Papers, Vol[C]. 10729 of Lecture Notes in Computer Science, Springer, 2017, pp. 174-179.

[108] S Nakajima, H N Bui, Dataset coverage for testing machine learning computer programs, in 23rd Asia-Pacific Software Engineering Conference, APSEC 2016, Hamilton, New Zealand, December 6-9, 2016 [C], IEEE Computer Society, 2016, pp. 297-304.

[109] S Nakajima, Dataset diversity for metamorphic testing of machine learning software, in Structured Object-Oriented Formal Language and Method-8th International Workshop, SOFL + MSVL2018, Gold Coast, QLD, Australia, November 16, 2018, Revised Selected Papers, Vol. 11392 of Lecture Notes in Computer Science[M], Springer, 2018, pp. 21-38.

[110] A Dwarakanath, M Ahuja, S Sikand, R M Rao, R P J C Bose, N Dubash, S Podder, Identifying implementation bugs in machine learning based image classifiers using metamorphic testing, in Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, Amsterdam[C], The Netherlands, July 16-21, 2018, ACM, 2018, pp. 118-128.

[111] A Sharma, H Wehrheim, Testing machine learning algorithms for balanced data usage, in: 12th IEEE Conference on Software Testing, Validation and Verification[C], ICST 2019, Xi'an, China, April 22-27, 2019, IEEE, 2019: 125-135.

[112] J M Zhang, E T Barr, B Guedj, M Harman, J Shawe-Taylor, Perturbed model validation: A new framework to validate model relevance[C], CoRR abs/1905.10201. arXiv: 1905.10201.

[113] S Al-Azani, J Hassine, Validation of machine learning classifiers using metamorphic testing and feature selection techniques, in Multi-disciplinary Trends in Artificial Intelligence-11th International Workshop [C], MIWAI 2017, Gadong, Brunei, November 20-22, 2017, Proceedings, Vol. 10607 of Lecture Notes in Computer Science, Springer, 2017: 77-91.

[114] Zhang T, Zhu Z. Interpreting adversarially trained convolutional neural networks[C]. International Conference on Machine Learning, 2019.

[115] M Zhang, Y Zhang, L Zhang, C Liu, S Khurshid, Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems, in Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering[C], ASE 2018, Montpellier, France, September 3-7, 2018, ACM, 2018: 132-142.

[116] M S Ramanagopal, C Anderson, R Vasudevan, M Johnson-Roberson, Failing to learn: Autonomously identifying perception failures for self-driving cars[J], IEEE Robotics Autom. Lett. 2018, 3(4): 3860-3867.

[117] J Kim, R Feldt, S Yoo, Guiding deep learning system testing using surprise adequacy, in Proceedings of the 41st International Conference on Software Engineering[C], ICSE 2019, Montreal, QC, Canada, May 25-31, 2019, IEEE/ACM, 2019: 1039-1049.

[118] E. Breck, N. Polyzotis, S. Roy, S. Whang, M. Zinkevich, Data validation for machine learning, in Proceedings of Machine Learning and Systems 2019[C], MLSys 2019, Stanford, CA, USA, March 31-April 2, 2019, mlsys. org, 2019.

[119] H Mei, J Lü, Internetware-A New Software Paradigm for Internet Computing[M], Springer, 2016. pp. 3-442.

[120] C Murphy, K Shen, G E Kaiser, Automatic system testing of programs without test oracles, in Proceedings of the Eighteenth International Symposium on Software Testing and Analysis[C], ISSTA 2009, Chicago, IL, USA, July 19-23, 2009, ACM, 2009: 189-200.

[121] Dong Y, Su H, Zhu J, Bao F. Towards interpretable deep neural networks by leveraging adversarial examples[J]. arXiv preprint arXiv: 1708.05493, 2017.

[122] W M McKeeman, Differential testing for software[J], Digital Technical Journal, 1998, 10(1): 100-107.

[123] M D Davis, E J Weyuker, Pseudo-oracles for non-testable programs, in Proceedings of the ACM 1981 Annual Conference[C], Los Angeles, CA, USA, November 9-11, 1981, ACM, 1981, pp. 254-257.

[124] V Le, M Afshari, Z Su, Compiler validation via equivalence modulo inputs, in ACM SIGPLAN Conference on Programming Language Design and Implementation[C], PLDI '14, Edinburgh, United

Kingdom-June 09-11, 2014, ACM, 2014, pp. 216-226.

[125] M Nejadgholi, J Yang, A study of oracle approximations in testing deep learning libraries, in: 34th IEEE/ACM International Conference on Automated Software Engineering[C], ASE 2019, San Diego, CA, USA, November 11-15, 2019, IEEE, 2019, pp. 785-796.

[126] A Aviziens, The Methodology of N-Version Programming, vol. 3, 1995, pp. 23-46.

[127] Wang Lin, Zhengfeng Yang, Xin Chen, Qingye Zhao, Xiangkun Li, Zhiming Liu, Jifeng He: Robustness Verification of Classification Deep Neural Networks via Linear Programming[C]. CVPR 2019: 11418-11427.

[128] S Srisakaokul, Z Wu, A Astorga, O Alebiosu, T Xie, Multiple implementation testing of supervised learning software, in: The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence[C], New Orleans, Louisiana, USA, February 2-7, 2018, Vol. WS-18 of AAAI Workshops, AAAI Press, 2018, pp. 384-391.

[129] V Tjeng, K Y Xiao, R. Tedrake, Evaluating robustness of neural networks with mixed integer programming, in: 7th International Conference on Learning Representations[C], ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview. net, 2019.

[130] C Dwork, M Hardt, T Pitassi, O Reingold, R S Zemel, Fairness through awareness, in: S. Goldwasser (Ed.), Innovations in Theoretical Computer Science 2012[C], Cambridge, MA, USA, January 8-10, 2012, ACM, 2012, pp. 214-226.

[131] M Hardt, E Price, N Srebro, Equality of opportunity in supervised learning, in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016 [C], December 5-10, 2016, Barcelona, Spain, 2016, pp. 3315-3323.

[132] I Zliobaite, Fairness-aware machine learning: a perspective[C], CoRR abs/1708.00754. arXiv: 1708.00754.

[133] F Doshi-Velez, B Kim, Towards a rigorous science of interpretable machine learning[J], arXiv: 1702.08608.

[134] B Herman, The promise and peril of human evaluation for model interpretability[C], CoRR abs/1711.07414, withdrawn.

[135] Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, Peter J Carlson, Andrew S Cassidy, Pallab Datta, Steven K Esser, Guillaume Garreau, Kevin L Holland, Scott Lekuch, Michael Mastro, Jeffrey L McKinstry, Carmelo di Nolfo, Brent Paulovicks, Jun Sawada, Kai Schleupen, Benjamin Shaw, Jennifer L Klamo, Myron D Flickner, John V Arthur, Dharmendra S Modha: TrueNorth: Accelerating From Zero to 64 Million Neurons in 10 Years[J]. IEEE Computer 52(5): 20-29(2019).

[136] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Q. Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy: TVM: An Automated End-to-End Optimizing Compiler for Deep Learning[C]. OSDI 2018: 578-594.

[137] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, Jian Lu. DISSECTOR: Input Validation for Deep Learning Applications by Crossing-layer Dissection[C]. ICSE 2020.

[138] J Zhang, H Lei, A pre-distribution algorithm of component reliability in internetware system, Comput [J]. 97(7)(2015) 755-768.

[139] X Zhang, H Chen, X Li, Z Qian, S Zhang, S Lu, Guarantee high reliability and effectiveness for softwares in internetware[C], in Proceedings of the 6th Asia-Pacific Symposium on Internetware, Internetware 2014, Hong Kong, China, November 17, 2014, ACM, 2014, pp. 107-115.

[140] Y Sun, X Huang, D Kroening, Testing deep neural networks, CoRR abs/1803.04792[J]. arXiv: 1803.04792.

[141] J Sekhon, C Fleming, Towards improved testing for deep learning, in Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results[C], ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019, IEEE/ACM, 2019, pp. 85-88.

[142] Andrew Yang: Deep Learning Training At Scale Spring Crest Deep Learning Accelerator (Intel® Nervana™ NNP-T)[C]. Hot Chips Symposium 2019: 1-20.

[143] X Du, X Xie, Y Li, L Ma, J Zhao, Y Liu, Deepcruiser: Automated guided testing for stateful deep learning systems, CoRR abs/1812.05339[J]. arXiv: 1812.05339.

[144] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, Peixin Zhang: Adversarial sample detection for deep neural network through model mutation testing[C]. ICSE 2019: 1245-1256.

[145] E Breck, S Cai, E Nielsen, M Salib, D Sculley, The ML test score: A rubric for ML production readiness and technical debt reduction, in 2017 IEEE International Conference on Big Data, Big Data 2017, Boston, MA, USA, December 11-14, 2017[C], IEEE Computer Society, 2017, pp. 1123-1132.

[146] T Byun, V Sharma, A Vijayakumar, S Rayadurgam, D D Cofer, Input prioritization for testing neural networks[C], in: IEEE International Conference On Artificial Intelligence Testing, AITest 2019, Newark, CA, USA, April 4-9, 2019, IEEE, 2019: 63-70.

[147] L Pulina, A Tacchella, An abstraction-refinement approach to verification of artificial neural networks, in Computer Aided Verification, 22nd International Conference, CAV 2010[C], Edinburgh, UK, July 15-19, 2010. Proceedings, Vol. 6174 of Lecture Notes in Computer Science, Springer, 2010: 243-257.

[148] G Katz, C W Barrett, D L Dill, K Julian, M J Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, in Computer Aided Verification-29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, Vol. 10426 of Lecture Notes in Computer Science[C], Springer, 2017: 97-117.

[149] R Ehlers, Formal verification of piece-wise linear feed-forward neural networks, in Automated Technology for Verification and Analysis-15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings, Vol. 10482 of Lecture Notes in Computer Science[C], Springer, 2017: 269-286.

[150] N Narodytska, S P Kasiviswanathan, L Ryzhyk, M Sagiv, T Walsh, Verifying properties of binarized deep neural networks, in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)[C], New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018: 6615-6624.

[151] N. Narodytska, Formal analysis of deep binarized neural networks, in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence[C], IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018: 5692-5696.

[152] A Lomuscio, L Maganti, An approach to reachability analysis for feed-forward relu neural networks, CoRR abs/1706.07351[J]. arXiv: 1706.07351.

[153] C Cheng, G Nührenberg, H Ruess, Maximum resilience of artificial neural networks, in Automated Technology for Verification and Analysis-15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings, Vol. 10482 of Lecture Notes in Computer Science[M], Springer, 2017: 251-268.

[154] S Dutta, S Jha, S Sankaranarayanan, A Tiwari, Output range analysis for deep neural networks, CoRR abs/1709.09130[J]. arXiv: 1709.09130.

[155] R Bunel, I Turkaslan, P H S Torr, P Kohli, M P Kumar, Piecewise linear neural network verification: A comparative study[C]. 2018.

[156] X Huang, M Kwiatkowska, S Wang, M Wu, Safety verification of deep neural networks, in Computer Aided Verification-29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I, Vol. 10426 of Lecture Notes in Computer Science[M], Springer, 2017, pp. 3-29.

[157] M Wu, M Wicker, W Ruan, X Huang, M Kwiatkowska, A game-based approximate verificationof deep neural networks with provable guarantees, Theor. Comput[C]. Sci. 807(2020) 298-329.

[158] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, Yadong Wang: DeepMutation: Mutation Testing of Deep Learning Systems[C]. ISSRE 2018: 100-111.

[159] Weijun Shen, Jun Wan, Zhenyu Chen: MuNN: Mutation Analysis of Neural Networks[C]. QRS Companion 2018: 108-115.

[160] Zixin Li, Haoran Wu, Jiehui Xu, Xingya Wang, Lingming Zhang, Zhenyu Chen: MuSC: A Tool for Mutation Testing of Ethereum Smart Contract[C]. ASE 2019: 1198-1201.

[161] Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska: Reachability Analysis of Deep Neural Networks with Provable Guarantees[C]. IJCAI 2018: 2651-2659.

[162] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, Marta Kwiatkowska: Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance [C]. IJCAI 2019: 5944-5952.

[163] Andrew Begel, Nachiappan Nagappan: Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study[C]. ESEM 2007: 255-264.

[164] Andrew Begel, Nachiappan Nagappan: Pair programming: what's in it for me? [C]. ESEM 2008: 120-128.

[165] Brendan Murphy, Christian Bird, Thomas Zimmermann, Laurie Williams, Nachiappan Nagappan, Andrew Begel: Have Agile Techniques been the Silver Bullet for Software Development at Microsoft? [C]. ESEM 2013: 75-84.

[166] Mali Senapathi, Jim Buchan, Hady Osman: DevOps Capabilities, Practices, and Challenges: Insights from a Case Study[C]. EASE 2018: 57-67.

[167] Ipek Ozkaya: What Is Really Different in Engineering AI-Enabled Systems? [J]. IEEE Softw. 37(4): 3-6(2020).

[168] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald C. Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, Thomas Zimmermann: Software engineering for machine

learning: a case study[C]. ICSE (SEIP) 2019: 291-300.

[169] Machine learning workflow, https://cloud. google. com/mlengine/docs/tensorflow/ml- solutions-overview.

[170] Vijayshankar Raman, Joseph M. Hellerstein: Potter's Wheel: An Interactive Data Cleaning System[C]. VLDB 2001: 381-390.

[171] Charles A Sutton, Timothy Hobson, James Geddes, Rich Caruana: Data Diff: Interpretable, Executable Summaries of Changes in Distributions for Data Wrangling[C]. KDD 2018: 2279-2288.

[172] Mirko Perkusich, Lenardo Chaves e Silva, Antonio Alexandre Moura Costa, Felipe Barbosa Araújo Ramos, Renata M Saraiva, Arthur Silva Freire, Ednaldo Dilorenzo, Emanuel Dantas, Danilo F S Santos, Kyller Costa Gorgônio, Hyggo O. Almeida, Angelo Perkusich: Intelligent softwareengineering in the context of agile software development: A systematic literature review[C]. Inf. Softw. Technol. 119 (2020).

[173] https://docs. microsoft. com/en-us/azure/devops/learn/devops-at-microsoft/.

[174] C Weber, B Curtis, and M B Chrissis: The capability maturity model, guidelines for improving the software process[C]. Addison Wesley, 1994.

[175] M Alexander, Six Sigma: The breakthrough management strategy revolutionizing the world's top corporations[C]. Taylor & Francis, 2001.

[176] C Zhang, A Liu, X Liu, H Yu, Y Ma and T Li. Interpreting and Improving Adversarial Robustness of Deep Neural Networks with Neuron Sensitivity[J]. arXiv preprint arXiv.

[177] Miryung Kim, Thomas Zimmermann, Robert DeLine, Andrew Begel: The emerging role of data scientists on software development teams[C]. ICSE 2016: 96-107.

[178] Miryung Kim, Thomas Zimmermann, Robert DeLine, Andrew Begel: Data Scientists in Software Teams: State of the Art and Challenges[J]. IEEE Trans. Software Eng. 44(11): 1024-1038(2018).

[179] Angela Horneman, Andrew Mellinger, and Ipek Ozkaya: AI Engineering: 11 Foundational Practices. https://resources. sei. cmu. edu/asset_files/WhitePaper/2019_019_001_634648. pdf.

[180] Ji Feng, Zhi-Hua Zhou: AutoEncoder by Forest[C]. AAAI 2018: 2967-2973.

[181] 纪守领, 李进锋, 杜天宇, 李博. 机器学习模型可解释性方法、应用与安全研究综述[C]. 计算机研究与发展, 2019, 56(10): 2071-2096.

[182] 成科扬, 王宁, 师文喜, 詹永照. 深度学习可解释性研究进展[J]. 计算机研究与发展, 2020, 57(6): 1208-1217.

[183] 綦小龙, 高阳, 王皓, 宋蓓, 周春蕾, 张友卫. 一种可度量的贝叶斯网络结构学习方法[J]. 计算机研究与发展, 2018, 55(8): 1717-1725.

[184] Hong Mei, Lu Zhang: Can big data bring a breakthrough for software automation[C]? Sci. China Inf. Sci. 61(5): 056101: 1-056101: 3(2018).

[185] 刘俊旭, 孟小峰. 机器学习的隐私保护研究综述[C]. 计算机研究与发展, 2020, 57(2): 346-362.

[186] 付印金, 肖 侬, 刘芳. 重复数据删除关键技术研究进展[C]. 计算机研究与发展, 2012, 49(1): 12-20.

[187] 陈立玮, 冯岩松, 赵东岩. 基于弱监督学习的海量网络数据关系抽取[C]. 计算机研究与发展, 2013, 50(9): 1825-1835.

[188] Zhe Xiao, Xiuju Fu, Rick Siow Mong Goh: Data Privacy- Preserving Automation Architecture for

Industrial Data Exchange in Smart Cities[J]. IEEE Trans. Ind. Informatics 14(6): 2780-2791 (2018).

[189] Yiqi Wu, Fazhi He, Dejun Zhang, Xiaoxia Li: Service-Oriented Feature-Based Data Exchange for Cloud-Based Design and Manufacturing[J]. IEEE Trans. Serv. Comput. 11(2): 341-353(2018).

[190] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, Lu Zhang: An empirical study on TensorFlow program bugs[C]. ISSTA 2018: 129-140.

[191] Ru Zhang, Wencong Xiao, Hongyu Zhang, Yu Liu, Haoxiang Lin, Mao Yang: An Empirical Study on Program Failures of Deep Learning Jobs[C]. ICSE 2020.

[192] 楼俊钢, 江建慧, 沈张果, 蒋云良. 软件可靠性预测的相关向量机模型[J]. 计算机研究与发展, 2013, 50(7): 1542-1550.

[193] 段文雪, 胡铭, 周琼, 吴庭明, 周俊龙, 刘晓, 魏同权, 陈铭松. 云计算系统可靠性研究综述. 计算机研究与发展, 2020, 57(1): 102-123.

[194] 徐光伟, 白艳珂, 燕彩蓉, 杨延彬, 黄永锋. 大数据存储中数据完整性验证结果的检测算法. 计算机研究与发展, 2017, 54(11): 2487-2496.

[195] 穆飞, 薛巍, 舒继武, 郑纬民. 一种面向大规模副本存储系统的可靠性模型. 计算机研究与发展, 2009, 46(5): 756-761.

[196] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, Lu Zhang: Automatic Testing and Improvement of Machine Translation[C]. ICSE 2020.

[197] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, Wenyuan Xu: DolphinAttack: Inaudible Voice Commands[C]. ACM Conference on Computer and Communications Security 2017: 103-117.

[198] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, Dongdi Zhang In: The 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering[C], November 8-13, 2020.

[199] Yuhao Zhang, Luyao Ren, Liqian Chen, Yingfei Xiong, Shing-Chi Cheung, Tao Xie. Detecting Numerical Bugs in Neural Network Architectures[C]. ESEC/FSE'20: ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, November 2020.

[200] Dong Wang, Ziyuan Wang, Chunrong Fang, Yanshan Chen, Zhenyu Chen: DeepPath: Path-Driven Testing Criteria for Deep Neural Networks[C]. AITest 2019: 119-120.

[201] Husheng Zhou, Wei Li, Yuankun Zhu, Yuqun Zhang, Bei Yu, Lingming Zhang, Cong Liu: DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems[C]. CoRR abs/1812.10812(2018).

[202] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao: Structural coverage criteria for neural networks could be misleading[C]. ICSE (NIER) 2019: 89-92.

[203] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Hongxu Chen, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, Simon See: Coverage-Guided Fuzzing for Deep Neural Networks[C]. CoRR abs/1809.01266(2018).

[204] Junhua Ding, Dongmei Zhang, Xin-Hua Hu: A Framework for Ensuring the Quality of a Big Data Service[C]. SCC 2016: 82-89.

[205] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, Jiaguang Sun: DLFuzz: differential fuzzing testing of deep learning systems[C]. ESEC/SIGSOFT FSE 2018: 739-743.

[206] Pengcheng Zhang, Qiyin Dai, Shunhui Ji: Condition-Guided Adversarial Generative Testingfor Deep Learning Systems[C]. AITest 2019: 71-72.

[207] Yanshan Chen, Ziyuan Wang, Dong Wang, Yongming Yao, Zhenyu Chen: Behavior Pattern-Driven Test Case Selection for Deep Neural Networks[C]. AITest 2019: 89-90.

[208] Long Zhang, Xuechao Sun, Yong Li, Zhenyu Zhang: A Noise-Sensitivity-Analysis-Based Test Prioritization Technique for Deep Neural Networks[C]. CoRR abs/1901.00054(2019).

[209] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, Jian Lü: Boosting operational DNN testing efficiency through conditioning[C]. ESEC/SIGSOFT FSE 2019: 499-509.

[210] Huangzhao Zhang, Zhuo Li, Ge Li, Lei Ma, Yang Liu, Zhi Jin: Generating Adversarial Examples for Holding Robustness of Source Code Processing Models[C]. AAAI 2020: 1169-1176.

[211] Zhengli Zhao, Dheeru Dua, Sameer Singh: Generating Natural Adversarial Examples[C]. ICLR (Poster) 2018.

[212] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, Dawn Song: Spatially Transformed Adversarial Examples[C]. ICLR (Poster) 2018.

[213] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, Jianguo Li: Boosting Adversarial Attacks With Momentum[C]. CVPR 2018: 9185-9193.

[214] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh: EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples[C]. AAAI 2018: 10-17.

[215] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, Dawn Song: Generating Adversarial Examples with Adversarial Networks[C]. IJCAI 2018: 3905-3911.

[216] Qi Lei, Lingfei Wu, Pin-Yu Chen, Alex Dimakis, Inderjit S. Dhillon, Michael J. Witbrock: Discrete Adversarial Attacks and Submodular Optimization with Applications to Text Classification[C]. MLSys 2019.

[217] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, Wenchang Shi: Deep Text Classification Can be Fooled[C]. IJCAI 2018: 4208-4215.

[218] Yong Cheng, Lu Jiang, Wolfgang Macherey: Robust Neural Machine Translation with Doubly Adversarial Inputs[J]. ACL (1) 2019: 4324-4333.

[219] Huangzhao Zhang, Hao Zhou, Ning Miao, Lei Li: Generating Fluent Adversarial Examples for Natural Languages[J]. ACL (1) 2019: 5564-5569.

[220] Shuhuai Ren, Yihe Deng, Kun He, Wanxiang Che: Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency[J]. ACL (1) 2019: 1085-1097.

[221] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, Dacheng Tao: Perceptual-Sensitive GAN for Generating Adversarial Patches[C]. AAAI 2019: 1028-1035.

[222] Lifeng Huang, Chengying Gao, Yuyin Zhou, Changqing Zou, Cihang Xie, Alan L. Yuille, Ning Liu: UPC: Learning Universal Physical Camouflage Attacks on Object Detectors[C]. CoRR abs/1909.04326 (2019).

[223] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, Carl A. Gunter: CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition[C]. USENIX Security Symposium 2018: 49-64.

[224] Zirui Xu, Fuxun Yu, Chenchen Liu, Xiang Chen: HASP: A High-Performance Adaptive Mobile Security Enhancement Against Malicious Speech Recognition[C]. CoRR abs/1809.01697(2018).

[225] Xiaoyuan Xie, Zhiyi Zhang, Tsong Yueh Chen, Yang Liu, Pak-Lok Poon, Baowen Xu: METTLE: A metamorphic testing approach to validating unsupervised machine learning methods [C]. IEEE Transactions on Reliability, 2020.

[226] Xiaoyuan Xie, Joshua Wing Kei Ho, Christian Murphy, Gail E. Kaiser, Baowen Xu, Tsong Yueh Chen: Application of Metamorphic Testing to Supervised Classifiers[C]. QSIC 2009: 135-144.

[227] Xiaoyuan Xie, Joshua Wing Kei Ho, Christian Murphy, Gail E. Kaiser, Baowen Xu, Tsong Yueh Chen: Testing and validating machine learning classifiers by metamorphic testing[J]. J. Syst. Softw. 84 (4): 544-558(2011).

[228] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao: DeepCT: Tomographic Combinatorial Testing for Deep Learning Systems[C]. SANER 2019: 614-618.

[229] Jeongju Sohn, Sungmin Kang, Shin Yoo: Search Based Repair of Deep Neural Networks[C]. CoRR abs/1912.12463(2019).

[230] Hao Zhang, W. K. Chan: Apricot: A Weight-Adaptation Approach to Fixing Deep Learning Models [C]. ASE 2019: 376-387.

[231] Junhua Ding, Xiaojun Kang, Xin-Hua Hu: Validating a Deep Learning Framework by Metamorphic Testing[C]. MET@ ICSE 2017: 28-34.

[232] Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, Lin Tan: CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries[C]. ICSE 2019: 1027-1038.

[233] Yi Qin, Huiyan Wang, Chang Xu, Xiaoxing Ma, Jian Lu: SynEva: Evaluating ML Programs by Mirror Program Synthesis[C]. QRS 2018: 171-182.

[234] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, Zhenyu Chen: DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks[C]. ISSTA 2020: 177-188.

[235] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, Yadong Wang: DeepGauge: multi-granularity testing criteria for deep learning systems[C]. ASE 2018: 120-131.

[236] 刘秀华, 韩建, 魏新江. 基于中间观测器的多智能体系统分布式故障估计[J]. 自动化学报, 2020, 46(1): 142-152.

[237] 陈海波, 刘杰, 顾荣, 张悠慧, 陈凯, 孟国柱, 周喆, 李武军, 李睿. AI 与系统软件的深度融合研究进展与趋势. 2017-2018 中国计算机科学技术发展报告: 100-146. 机械工业出版社, 2018 年.

[238] 李俊杰, 王茜. 感知相似的图像分类对抗样本生成模型[J]. 计算机科学与探索. 2020.

[239] 马玉琨, 毋立芳, 简萌, 刘方昊, 杨洲. 一种面向人脸活体检测的对抗样本生成算法[J]. 软件学报, 2019, 30(2): 469-480.

[240] 王曙燕, 金航, 孙家泽. GAN 图像对抗样本生成方法[J]. 计算机科学与探索.

[241] 王文琦, 汪润, 王丽娜, 唐奔宵. 面向中文文本倾向性分类的对抗样本生成方法[J]. 软件学报, 2019, 30(8): 2415-2427.

[242] 崔铁军, 李莎莎. 人工智能系统故障分析原理研究[J]. 智能系统学报: 1-7[2020-08-23]. http://kns.cnki.net/kcms/detail/23.1538.TP.20200720.1629.004.html.

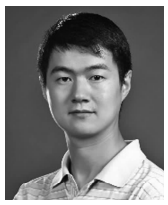
[243] 刘艾杉, 王嘉凯, 刘祥龙. 人工智能安全与评测[J]. 人工智能, 2020, (03), 32-42.

[244] 张伟娜, 李鸣, 刘亭杉, 于泉杰. 基于对抗性样本的深度学习算法可靠性评估研究[J]. 电子测试, 2020. 03: 35-37.

[245] 张璨, 张明英. 人工智能深度学习算法可靠性评估方法研究[J]. 信息技术与标准化, 2018,

(08), 38-42.
[246] 甘滨. 面向深度学习模型的安全测试平台的研究与实现[D]. 山东大学, 2020.
[247] 赵威, 王锴, 徐皑冬, 曾鹏, 杨顺昆. 面向智能制造的工业机器人健康评估方法[J]. 机器人, 2020, 42(04), 460-468.

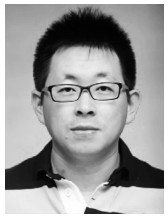
作者简介



陈雨亭 上海交通大学副教授、博士生导师，CCF 系统软件专委会常务委员。主要研究方向包括程序分析与软件测试、系统软件可靠性、编译器和虚拟机等。

谢晓园 武汉大学计算机学院教授，博士生导师，CCF 软件工程专委会委员，CCF YOCSEF 委员。主要研究方向为软件测试、程序分析与调试、智能软件工程等。

周 宇 南京航空航天大学教授、博士生导师，CCF 软件工程专委会委员、系统软件专委会委员，江苏省计算机学会软件专委会副主任。主要研究方向包括智能化软件开发、软件可靠性技术、软件演化分析、云计算和大数据等。



马 雷 日本九州大学副教授。主要研究方向包括智能软件的品质保障及可靠性、软件测试与分析等。