计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(1): 41−65, 2026

数据调试综述

李晨阳　马超红　孟小峰
（中国人民大学信息学院　北京　100872）
（chenyangli@ruc.edu.cn）

Survey on Data Debugging

Li Chenyang, Ma Chaohong, and Meng Xiaofeng
（*School of Information*, *Renmin University of China*, *Beijing* 100872）

Abstract　The booming of artificial intelligence (AI) plays a crucial role in various fields (e.g., healthcare, bioinformatics, and financial services). The main paradigm of AI applications involves building machine learning models to explore insights and patterns within data, which are then utilized for reasoning and decision-making. The effectiveness and efficiency of AI systems hinge on two key factors: the first is the model aspect (model-centric), involving the enhancement of network structure like the transitioning from RNN to LSTM, and the optimization of model hyperparameters; the second is the data aspect (data-centric), such as standardizing data formats, increasing data size, reducing data noises. For a long time, debugging AI systems primarily focuses on optimizing models. However, the rise of the digital era, characterized by activities such as social networking and e-commerce, has produced vast and varied data sets, making model-centric debugging fall short of fulfilling the requirements of modern applications. Consequently, both research and industry communities have redirected their focus from models to data to address this gap. To this end, "data debugging" has emerged. Rather than tuning model, data debugging places emphasis on scrutinizing data, which understands the impact of data errors on downstream tasks at each stage of machine learning pipeline, and then debug the corresponding errors to improve model performance. On the basis of a comprehensive survey of data debugging related work, the research framework for data debugging is first proposed. According to the interaction between data debugging methods and machine learning pipeline, the existing methods are categorized into closed data debugging, immersive data debugging, and hybrid data debugging. Then, we thoroughly provide a detailed overview of related work in this field. Next, we empirically evaluate data debugging methods, and summarize the commonly used datasets and evaluation metrics in this research area. Finally, the challenges and future directions of data debugging are pointed out.

Key words　data debugging; machine learning; data quality; data cleaning; artificial intelligence

摘　要　人工智能的蓬勃发展，对医疗健康、生物信息、金融服务等各领域产生深远影响。人工智能应用的主要范式是构建机器学习模型，探索数据中的规则和模式，以用于推理和决策。人工智能系统的有效性和效率取决于2个关键方面：其一是模型方面（以模型为中心），包括增强网络结构，如RNN到LSTM的转变、模型超参数调优等；其二是数据方面（以数据为中心），如标准化数据格式、增大数据量、减少数据噪声等。一直以来，调试人工智能系统主要侧重于优化模型。然而，以社交网络和电子商务为代表的数字化时代的到来产生庞大且多样的数据，使得以模型为中心的调试已无法满足人们对人工智能系统的

收稿日期：2025-03-10；**修回日期**：2025-09-15
基金项目：国家自然科学基金项目（62172423）
This work was supported by the National Natural Science Foundation of China (62172423).
通信作者：孟小峰（xfmeng@ruc.edu.cn）

需求。因此,研究界和工业界将注意力从模型转向数据,以弥补这一差距。为此,“数据调试”(data debugging)应运而生。与优化模型不同,数据调试侧重检查数据,即理解错误数据在机器学习管道的各阶段对下游任务的影响,进而调试相应错误以提高模型性能。基于此,在全面调研数据调试相关工作的基础上,首先,提出数据调试研究框架,根据数据调试方法与机器学习管道的交互,将现有方法分为封闭式数据调试、浸入式数据调试和混合式数据调试3类。接着,详细概述本领域的相关工作。然后,对数据调试方法进行实验评估,同时总结该研究领域常用的数据集和评价指标。最后,指出数据调试面临的挑战及未来发展方向。

关键词 数据调试;机器学习;数据质量;数据清洗;人工智能

中图法分类号 TP274

DOI: 10.7544/issn1000-1239.202550195 **CSTR:** 32373.14.issn1000-1239.202550195

人工智能自20世纪40年代萌芽以来,经历了60年代的寒冬、80年代的复兴,并在21世纪初迎来快速发展[1]。如今,人工智能正广泛应用于医学影像[2]、智能机器[3]、生物基因[4]、金融服务[5]等众多领域,推动社会各行业的变革。这些领域不断产生海量、多样化数据,为机器学习提供了前所未有的土壤。机器学习模型凭借其在大规模数据中提取规则和模式的能力,通过迭代优化参数,使最终预测更贴近真实结果,以用于下游应用[6]。因此,模型和数据成为人工智能取得成功的2个主要原因。长期以来,为追求更优的预测结果,研究人员一直专注于模型的优化,例如变换模型架构、增加模型层数或更改模型参数等[7]。这种以模型为中心的调试方式虽然显著提升了模型性能,但目前已遭遇难以突破的瓶颈,且往往忽略了数据的影响[8-9]。随着数据时代的发展,网络社交、线上购物和互联网交易等行为不断涌现出体量急剧扩张且模式复杂多样的新型数据源,数据日益成为决定模型性能上限的关键因素。这促使人们将目光从以模型为中心的调试转向以数据为中心的调试。

在这一背景下,人工智能推动了软件1.0到软件2.0的发展[10]。相较于传统软件1.0依赖人工编码实现功能,软件2.0(即人工智能系统(artificial intelligence system, AI system))由代码(模型或算法)、数据和硬件3部分组成[11],其行为主要由数据驱动生成,可视为“由数据编译出的软件”。因此,在整个人工智能系统开发过程中有效管理数据已成为保障高质量软件2.0的重要任务[12]。在当前人工智能系统效能提升的需求下,以数据为中心的调试正发挥关键作用,主要体现在以下2个方面:

首先,数据始终具有潜在的质量问题。例如,某健康监测系统以磅作为体重的基本单位,但用户却输入以千克为单位的体重[13];某训练模型期待数据以每周的频率输入,但数据提供者将频率更换成每日[14];某购物商场根据会员集中地做出免费配送范围的决策,但不同种族所占会员比例不同[15]。由此,上述示例说明数据总是存在格式错误、缺失和不平衡等质量问题。然而,研究表明,高质量的数据更容易得到良好的模型。例如,最早成功的卷积神经网络之一AlexNet[16]是基于具有高质量标注的ImageNet数据集[17]设计的;AlphaFold[18]在大量人工标注的蛋白质序列上实现[19],成为人工智能在科学发现领域的重大突破之一;大型语言模型的发展同样需要依赖大型文本数据[8,20-23];监督学习模型的决策能力取决于众包数据的标注质量[24-25]。

其次,数据对模型性能产生极大影响,存在于数据中的错误通常会引发下游应用不可预期的后果[26-27]。例如,某供应链中,携带错误标签的输入数据导致商品供应预测延迟的问题[26];某航班延误预测模型由只包含日间航班的数据集训练,将夜间航班作为测试致使模型预测错误[28]。最近,人工智能学者吴恩达(Andrew Ng)提出“以数据为中心的人工智能(data-centric AI)”理念[8],指出在过去一段时间内调试人工智能系统的主流范式是改进模型和参数,因此,机器学习模型已经具备显著成效,对于多数应用,模型和参数不再是一个亟需解决的问题。该理念强调,进一步提高性能应转向数据,同时指出,从机器学习管道(收集、处理、训练、评估和应用)的各阶段中寻找并修复数据能够取得更好的性能。为了验证这一理念,吴恩达团队组织一项Data-Centric AI Competition[29]比赛,这一比赛要求参与者在固定模型结构下,通过迭代改进数据集,显著提升了模型性能[8]。这一系列研究与实践促使人们重新认识数据在人工智能系统中的作用。

在这一趋势下,数据已成为人工智能系统的核

心基础设施，并且其支撑作用日益凸显[7,30-32]。人工智能系统正从以模型为中心(model-centric)向以数据为中心(data-centric)转变[33]，这一转变标志着数据不仅是人工智能系统的基础，更是推动其性能提升的关键驱动力。与此同时，数据在人工智能系统中的价值同样得到各行各业的一致认可。为了构建性能卓越的人工智能系统，许多大型科技公司正在积极提出组织、理解和调试数据的底层机制和方法[8,34-37]。上述产学研各界采取的行动，进一步说明以数据为中心调试人工智能系统的重要性。从而引出了以数据为中心调试人工智能系统的思想，打破了传统以模型为中心调试人工智能系统的方式。

综上，随着人工智能系统调试变革引出以模型为中心和以数据为中心2种不同调试方式，其版本、所处社会形态及相关案例如表1所示。

Table 1 Revolution in AI System Debugging

表1 AI系统调试的变革

AI系统调试方式	版本	社会形态	案例
以模型为中心	AI系统调试1.0	信息时代	RNN到LSTM的改进[38]、超参数调优[8]
以数据为中心	AI系统调试2.0	数据时代	统一数据格式[39]、增大数据量、减少噪声[13]

在以数据为中心的推动下，人工智能开发中逐渐涌现出一系列数据质量检查工序[40]。过去，这些数据质量检查工作大多源自个别开发者基于特定需求提出的非普适性方法，缺乏系统性和一致性。由于长期缺少对数据的投入与关注，用于人工智能开发的数据质量检查工作整体水平有限，尤其是在医疗诊断等对数据可靠性要求极高的高风险行业中，这一问题更为突出[41]。在此情形下，研究人员开始尝试设计用于识别和修复数据问题的系统化手段，其中较早出现的代表性方法是数据清洗。这类方法一般是在机器学习管道的初期阶段(数据收集和预处理)使用，目的是通过检测和修复格式错误或缺失数据等问题来改善数据。然而，它通常是静态的，即只关注管道前期的数据质量，忽视了机器学习管道中后续阶段出现的复杂多样的数据问题[42-43]。这种静态策略难以应对数据质量问题在人工智能系统运行过程中的动态演化，可能导致模型在下游任务中的性能下降，即使模型具备一定程度的错误容忍能力，也难以完全缓解这些隐形风险。

为克服静态清洗的局限性[44-45]，逐渐兴起一种更具系统性和动态性的研究方法——数据调试(data debugging)[13]。数据调试更强调数据在整个机器学习管道中的作用，迭代地采用系统化策略来解决其各阶段(收集、处理、训练、评估到应用)产生的数据错误问题，从而获得更有效的模型。具体而言，数据调试构建了一个动态的识别、修复和评估过程，用于揭示并消除数据中潜在的错误与偏差，从调试数据的角度改善人工智能系统。因此，数据调试可以被视为联结数据质量管理与机器学习开发之间的一种中介机制[46]，正在成为人工智能系统开发中的一个关键研究热点。

数据调试具有广泛的研究与用途。例如，在可信机器学习方面，通过数据调试消除数据偏见，以保证模型的稳健和公平性[7,47]；在人工智能安全性方面，通过数据调试生成对抗样本来攻击神经网络[48]，能够间接提升模型的鲁棒性；在机器学习可解释性方面，数据调试通过报告训练数据中错误，能够间接理解模型(尤其是复杂的黑盒模型)对训练数据的推理过程，以帮助模型开发人员改善模型性能[49]；在数据管理方面，通过数据调试增强数据的一致性和规范性，有助于后续数据分析和应用[50-51]。

基于此，本文将围绕数据调试展开系统性综述，总结数据调试研究领域的相关工作，提出数据调试研究框架，给出数据调试的定义，对已有方法评估，并指出该领域面临的挑战及未来研究方向。本文的主要贡献有3个方面：

1)从以数据为中心调试人工智能系统的视角，引入了数据调试这一概念，强调数据调试是面向整个机器学习管道各阶段，用于识别并修复数据错误的过程，从而提升模型性能。进一步地，本文给出数据调试的定义，明确其涵盖当前以数据为中心的主要调试策略。

2)对数据调试方法进行系统划分，构成数据调试研究框架。类比软件调试方法的分类结构[52]，数据调试方法可以分成封闭式数据调试、浸入式数据调试以及混合式数据调试三大类。更具体地，本文将详细介绍三大类中的具体方法，并总结和归纳了具体方法的优缺点、适用范围以及应用方式。

3)整理数据调试研究领域常用的评估指标和数据集，并对部分数据调试方法进行测试评估，填补了尚未有研究在统一的指标和数据集下对数据调试方法进行定量比较的空缺。此外，本文根据对现有数据调试研究的分析，结合当前社会在数据质量管理与人工智能系统交叉方面面临的问题，讨论未来研究方向和机遇，为解决该领域存在的具体挑战提供参考思路。

1 数据调试研究框架

本节首先定义数据调试;接着根据数据调试的适用范围,将数据调试分为封闭式数据调试、浸入式数据调试和混合式数据调试三大类,并提出数据调试研究框架。表 2 是本文使用的符号及含义。

Table 2 List of Notations

表 2 符号表

符号	含义
D_{train}	原始训练数据集
D_{test}	原始测试数据集
E	评估函数
$M^{+}(\theta)$	模型架构
θ^{+}	参数配置
D_{error}	错误数据集(可能存在于D_{train}或D_{test})
D'_{train}	修复训练数据集
D'_{test}	修复测试数据集
$M(\theta)$	原始训练数据集上训练获取的模型
M_{debug}	修复(调试)训练数据集上训练获取的模型
E_{debug}	调试后模型在评估函数下对应的指标
E_{u}	评估函数对应的指标上限
$Sub(x,y)$	计算 x 与 y 之间差值的函数
$\mathcal{F}$	数据调试方法
M_{best}	理想训练数据集上训练获取的模型
φ	一致性约束
lb	数值属性之间算术关系的下限
ub	数值属性之间算术关系的上限
P	数值属性之间算术关系

1.1 研究问题定义

在人工智能系统中,数据错误对模型性能影响显著。数据调试的目的是:系统性地定位并修复机器学习管道中的数据错误,从而使在修复数据上得到的模型具有更好的准确性,即尽可能贴近评估函数的指标上限E_{u}。数据调试具有 2 个重要特点:1)流程覆盖性,聚焦机器学习管道中各阶段可能引入的数据错误;2)目标驱动性,以提升模型性能为直接目标。

相对于以模型为中心的调试,即通过堆叠机器学习模块或更换模型参数在D_{train}上获得训练模型,并在D_{test}上迭代寻找更优的架构$M^{+}(\theta)$或参数θ^{+}以获得更佳评估函数E;以数据为中心的调试更关注数据与模型性能之间的因果链条,其基本思想是:通过纠正影响模型的数据错误来提高M_{debug}的性能E_{debug},使性能$E_{\text{debug}}=E_{\text{debug}}(M_{\text{debug}}(\theta),D'_{\text{train}},D_{\text{test}})/E_{\text{debug}}(M_{\text{debug}}(\theta),D_{\text{train}},D'_{\text{test}})$与$E_{\text{u}}$的差值最小。下面进一步给出数据调试的形式化定义。

定义 1. 数据调试。假设给定原始模型$M(\theta)$、原始训练数据集D_{train}、原始测试数据集D_{test}、错误数据集D_{error}($D_{\text{error}}\subseteq D_{\text{train}}$或$D_{\text{error}}\subseteq D_{\text{test}}$)、数据调试方法$\mathcal{F}$、修复训练数据集$D'_{\text{train}}$、修复测试数据集$D'_{\text{test}}$、差值函数$Sub(E_{\text{debug}},E_{\text{u}})$,其中$|Sub(E_{\text{debug}},E_{\text{u}})|\geqslant 0$恒成立,$M_{\text{debug}}$越接近$M_{\text{best}}$,则$|Sub(E_{\text{debug}},E_{\text{u}})|$越小。数据调试是指通过数据调试方法$\mathcal{F}$定位并修复在机器学习管道各阶段中由$D_{\text{train}}$或$D_{\text{test}}$产生的$D_{\text{error}}$,以使修复训练数据集$D'_{\text{train}}$或原始训练数据集$D_{\text{train}}$上训练获得调试后的$M_{\text{debug}}$趋近于$M_{\text{best}}$,且在原始测试数据集$D_{\text{test}}$或修复测试数据集$D'_{\text{test}}$上获得的模型性能满足$|Sub(E_{\text{debug}},E_{\text{u}})|\to 0$。

针对如何衡量M_{debug}和M_{best}之间的逼近程度,现有研究普遍采用模型预测结果的准确率作为评估函数。在具体任务中,该指标可细化为预测准确率、分类准确率、回归准确率等。进一步地,对于预测与分类任务,通常采用准确率和召回率来判断当前模型是否在机器学习管道中隐含数据错误,当模型的准确率和召回率与指标上限的差值越小,则说明该模型应用的数据质量越好、在机器学习管道中的数据处理越可靠;对于回归任务,一般采用均方误差来确定数据集或模型运行中是否存在需要调试的数据错误,当模型的均方误差与指标上限尽可能地贴近,则认为不存在影响该模型的数据错误。尽管如此,现有研究尚未形成统一的性能评估标准,它们依据不同任务对模型的不同评估方式来指导数据调试(详细可见第 5 节“数据调试方法评估”)。此外,需要注意的是,在数据调试过程中盲目追求模型性能、无限制地修复数据是不可取的[53],这不仅可能引入过拟合风险,也会带来较高的计算与资源开销。因此,探索有效反映模型受数据影响的指标也是数据调试研究的目标之一,以保证模型性能的同时,减少不必要的模型重训练成本。

1.2 研究框架

如定义 1,数据调试方法专注于定位和修复机器学习管道各阶段中出现的数据错误,使在修复数据上获得的模型性能尽可能接近理论指标上限,从而提高人工智能系统的整体有效性。为了系统地研究和组织现有数据调试方法,有必要从机器学习管道的全流程视角构建数据调试的研究框架。

人工智能系统通常以机器学习模型为核心,其开发依赖于一套标准化的机器学习管道。图 1 中间

部分展示的是机器学习管道，该管道涵盖从数据收集、数据预处理、特征工程、训练模型、评估模型到部署(应用)等关键阶段。值得注意的是，整个机器学习管道均有可能发生数据错误，降低模型性能，因此数据调试从数据角度出发解决不同阶段产生的数据错误来提高模型性能。

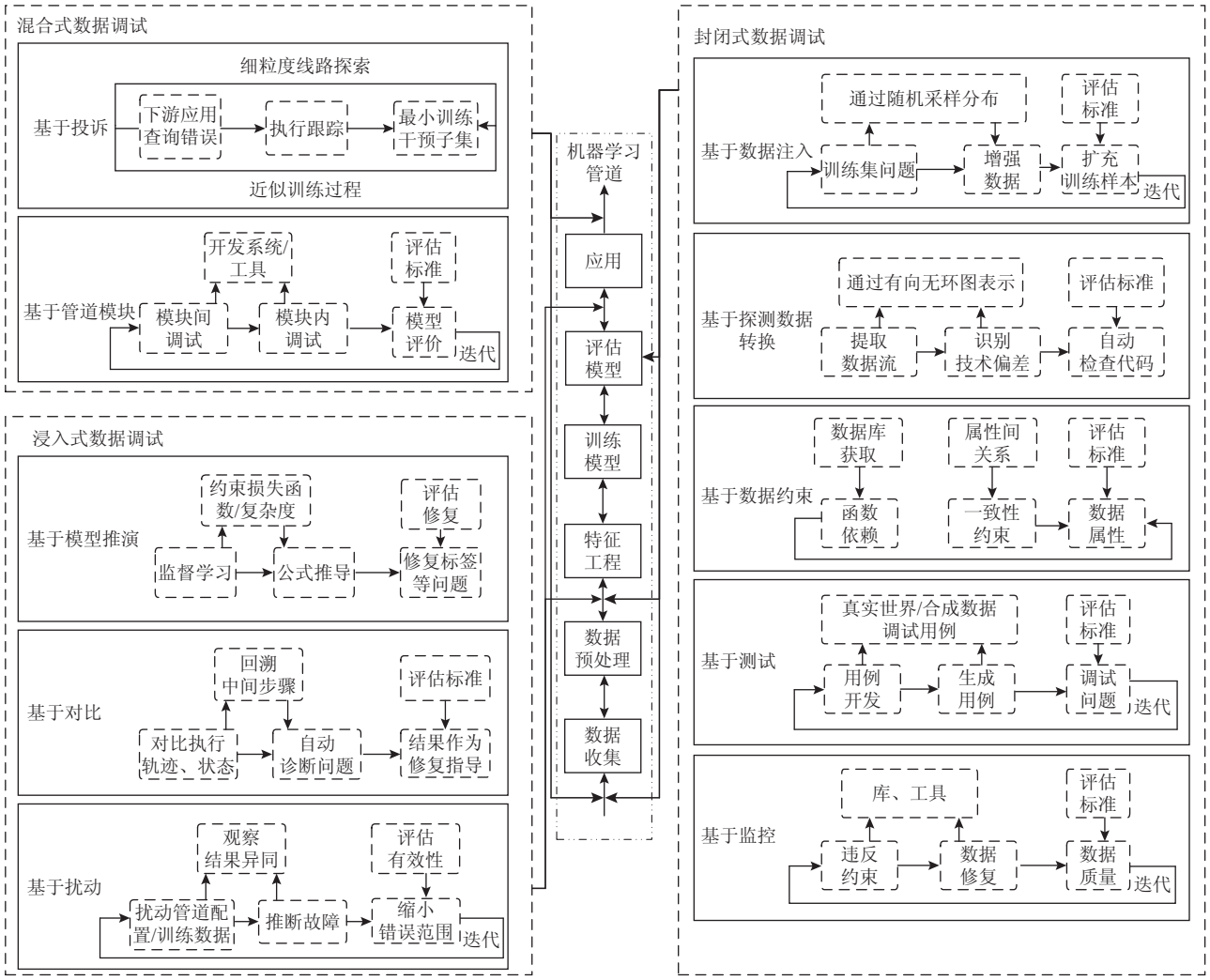


Fig. 1 Data debugging research framework

图 1 数据调试研究框架

基于此，本文提出如图 1 所示数据调试研究框架，按照数据调试过程中模型的可访问性或透明性，将数据调试方法分为三大类：封闭式数据调试、浸入式数据调试以及混合式数据调试。主要围绕调试策略和执行机制介绍每一类方法的调试过程，以及在机器学习管道中所处的阶段。

1)封闭式数据调试。封闭式数据调试(closed data debugging)对应于机器学习管道的两端(end_pipeline)，包含数据收集、预处理和评估阶段，是一类在受限条件下进行的数据调试方法。它一般在模型未知的环境中实施，只对输入数据展开操作，而不涉及整个机器学习管道。具体地，封闭式数据调试可表示为一个函数$\mathcal{F}_{\text{clo}}$，该函数接收原始训练数据集D_{train}或原始测试数据集D_{test}作为输入，利用数据注入、探测数据转换、数据约束、测试和监控的策略调试D_{train}或D_{test}中的D_{error}，并返回修复训练数据集D'_{train}或修复测试数据集D'_{test}。这一过程可形式化为$\mathcal{F}_{\text{clo}}(D_{\text{train}}/D_{\text{test}}) \rightarrow D'_{\text{train}}/D'_{\text{test}}$。封闭式数据调试中不同调试策略对应的详细流程见图 1 右侧。关于封闭式数据调试的介绍详见第 2 节。

2)浸入式数据调试。浸入式数据调试(immersive data debugging)对应于机器学习管道的中间阶段(in_pipeline)，即特征工程、训练模型和评估模型，是一类在未受限条件下进行的数据调试方法。它一般在模型已知的环境中实施，直接深入模型内部观测运行状态和数据流动，涉及整个机器学习管道的任意中

间阶段。具体地,浸入式数据调试可表示为$\mathcal{F}_{\text{imm}}$,该函数接收在原始训练数据集D_{train}上训练获取到的模型$M(\theta)$作为输入,利用扰动、模型推演和对比的策略调试D_{train}中影响模型$M(\theta)$的错误数据集D_{error}或其他错误管道配置,并返回在修复训练数据集或正确管道配置上训练获取的模型M_{debug}。这一过程可形式化为$\mathcal{F}_{\text{imm}}(M(\theta)) \rightarrow M_{\text{debug}}$。浸入式数据调试中不同调试策略对应的详细流程见图 1 左下。关于浸入式数据调试的介绍详见第 3 节。

3)混合式数据调试。混合式数据调试(hybrid data debugging)对应于整个机器学习管道的所有阶段(full_pipeline),是一类在结合受限和未受限条件下进行的数据调试方法。它一般在模型已知的环境中实施,同时考虑输入数据和模型内部,可在整个机器学习管道的任意阶段灵活切换。具体地,混合式数据调试可表示为一个函数$\mathcal{F}_{\text{hyb}}$,该函数接收与原始训练数据集D_{train}相关的信息或在原始训练数据集D_{train}上训练获取到的模型$M(\theta)$作为输入,利用投诉和管道模块化的策略找到D_{train}中的错误数据集D_{error}或返回在修复训练数据集上训练获取的模型M_{debug}。这一过程可形式化为$\mathcal{F}_{\text{hyb}}(D_{\text{train}}/M(\theta)) \rightarrow D_{\text{error}}/M_{\text{debug}}$。混合式数据调试中不同调试策略对应的详细流程见图 1 左上。关于混合式数据调试的介绍详见第 4 节。

综上,封闭式数据调试主要聚焦于模型的输入和输出,通常通过直接检查、识别和修复原始输入数据及其转换过程中的问题来辅助后续模型开发,这类方法执行速度较快,但难以保障M_{debug}的性能。相较之下,浸入式数据调试深入模型内部,通过分析、探索和定位模型内部运行机制和管道配置中的数据问题来改进模型性能,这类方法保证了M_{debug}的性能,但调试速度较慢。混合式数据调试则同时考虑了上述 2 种方法的思想,兼顾对输入输出数据的处理与对模型结构的深入分析。这类方法通过复合策略全面保障了M_{debug}性能。

相对于数据调试的陌生,软件调试(software debugging)更为人们熟知。软件调试和数据调试分别在软件开发和模型开发过程中承担重要“责任”,它们之间既有紧密联系又有明显区别。这种联系体现在它们的核心目标都是调试错误。而区别在于软件调试致力于源码与软件运行之间隐匿的故障原因;然而,数据调试则关注数据与模型应用之间潜在的脱节问题[13],这种脱节问题大多由不同类型的数据错误引起。因此,数据调试的重点放在对有可能影响模型开发或执行性能的数据错误的根因追溯和修复上,相关的数据错误类型见表 3。

Table 3 Data Error Types and Meanings

表 3 数据错误类型及含义

错误类型	含义
异常值[54]	数值处于经验区间外的数据
模式错误[55]	与整体数据的构成形式不一致的数据
格式错误[55]	与系统存储数据格式不相符的数据
重复数据[56]	至少存在 2 条完全相同记录的数据
缺失数据[56]	非正常空值的数据
数据漂移[28]	测试数据分布与训练数据分布不匹配
标签错误[57]	训练数据与其标签不匹配
特征错误[58]	提取特征过于特定或不具代表性
对抗攻击样本[59]	添加微小扰动或修改特征的数据
数据稀缺[60]	训练数据规模过小
数据转换错误[61]	数据预处理中过滤器或连接等操作改变训练数据代表的不同分布
嵌入错误[62]	数据编码错误
数据不平衡[15]	训练数据/测试数据中不同类别数量差异显著
噪声[60]	具有随机误差或方差的干扰数据

2 封闭式数据调试

封闭式数据调试方法一般指模型未知的情况下,通过纠正输入数据来实现数据调试。这类方法通常对纠正输入的策略进行划分,利用测试用例、数据规范、数据改进、更正技术偏差以及数据约束来调试数据,进而调整模型,使模型性能接近指标上限,得到的模型与最优模型更接近。基于此,本文将此类方法细分为 5 种,分别是基于测试、监控、数据注入、探测数据转换和数据约束的调试策略。

2.1 基于测试的调试方法

基于测试的封闭式数据调试方法通过查询语句或创建模板生成测试用例,以破坏测试用例的方式让操作者快速发现引起下游模型故障的数据原因。这类方法的关键设计在于查询即将应用的输入数据[63],自动验证或返回给操作者查阅,保留正确数据并直接修改错误数据。具体地,不同形式的测试可归纳为连续测试和单元测试。

2.1.1 连续测试

连续测试通过使用空闲 CPU 在后台对操作者正在更改的数据库开展测试查询,从而迅速发现测试失败原因。虽然在分析时将测试失败的原因主要归咎于数据错误,但实际上数据错误的形式是多种多样的。即使数据库内置完整性约束等规范数据模式

的功能，却依然会遗漏个别数据录入错误、数值错误或数据融合错误等[64]，这些未被发现的错误会在后续数据检索和实际应用中逐渐显现，从而扩散其对下游应用的负面影响。

为了应对上述挑战，Muslu 等人[54]利用连续测试来预防数据库在更新数据时引入的错误，降低从错误引入到错误发现的时间开销。如图 2 所示，一旦操作者更新数据库的数据与测试冲突，操作者能够即刻发现错误数据问题。进一步地，假设每个测试都通过人为编写或模板生成一条 SQL 查询，并以黑盒形式不断在数据库更新过程中对数据执行测试，实时地将测试结果的变化报告给操作者，那么越早报告越有助于操作者纠正错误。而 Kontokostas 等人[65]同样利用查询测试的方式来整合词汇表、知识库与测试用例，一旦数据被更新则应用对应的测试用例发现错误编辑并保证数据质量。

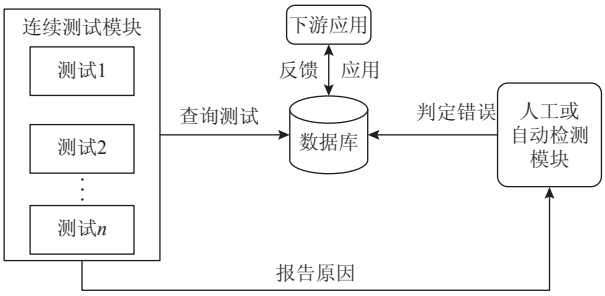


Fig. 2 Continuous testing framework

图 2 连续测试框架

此外，在推进连续测试的研究过程中，研究者发现测试生成和调试应用/模型性能[66-67]同样值得深入探索。然而，相比之下，针对数据问题的连续测试策略的研究更为重要，是上述方向的基础。

2.1.2 单元测试

单元测试要求操作者结合预先定义的数据质量约束与验证代码，对特定区域内的数据问题进行测试。由于机器学习的广泛使用，让研究人员越来越关注机器学习极度依赖的输入数据。然而，实际生活中的输入数据来源十分广泛，将这些以不同格式存在的数据源直接用于训练是不现实的。然而，这种情况不仅出现在结构化数据中，类似视频等半结构化数据作为推荐系统的输入数据时，也需要根据使用状况录入日志文件，而视频出现的数据问题也会反映到日志文件内。因此，为了维护输入数据，需要对相应的日志文件进行验证。

Schelter 等人[68]提供了一个声明性的应用程序接口 D-API，试图让操作者利用大量组合约束来定义数据的规范。具体来说，操作者可以从现有约束功能中自由选取或自主编写多个函数，组成个性化的自定义约束集合，并根据约束条件将数据质量转化为度量计算，一旦最终度量结果不佳，则生成错误或警告。Barowy 等人[69]介绍了一种结合统计辅助程序分析的方法，用于发现电子表格中对计算结果影响较大的单元格，并协助操作者判断其是否错误，从而缩小定位范围。该方法受灵敏性分析[70-71]启发，不仅从输入影响模型输出的角度来确定重要输入，还深入挖掘模型错误的根源。

2.2 基于监控的调试方法

基于监控的封闭式数据调试方法通过将检测、诊断和评估等监控模块分别引入机器学习模型的输入和输出，辅助分析模型不良性能，并据此发现潜在的数据问题。这类方法的中心思想体现在先利用一些已有库或日志发现数据错误，再基于其他现有算法修正错误，随后将修复数据用于模型重训练。

Rukat 等人[55]提出自动化机器学习执行过程中的操作，尤其关注对输入数据的监控：首先用 Deequ[68]库检测输入数据集中可能的数据错误，再通过 DataWig[72-73]库将错误数据替换为适当的值，最后利用一个可学习的管道性能预测器（pipeline performance prediction，PPP）来验证纠正的错误是否是导致模型性能下降的原因。Abdelaal 等人[56]考虑到表格数据集（tabular data）往往不只存在 1 类错误数据，而可能会出现不同类别的数据问题，如空值、冗余以及分布不一致等。由此，为存在复杂数据问题的数据集匹配合适的检测和修复方法是令人苦恼的。为了应对这一挑战，Abdelaal 等人[56]引入一个较为全面的基准框架 REIN，如图 3 所示。它首先利用错误注入模块为数据集生成多类错误，再通过检测模块和修复模块解决所有可获取的数据问题，并设计控制器模块来辅助数据

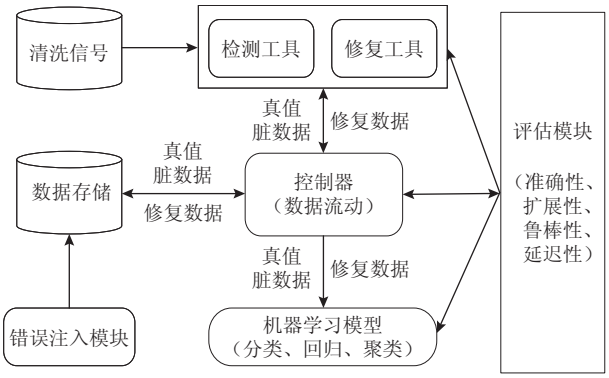


Fig. 3 REIN framework

图 3 REIN 框架

准备、检验、流动需求,从而系统地评价各种检测和修复方法对机器学习模型的影响。

除了监控输入之外,Shankar 等人[74]将监控范围扩展到机器学习模型执行全程,提出一个额外的机器学习可观察性系统架构。它从模型分类有误出发,定位可能出错的机器学习相关组件(如预处理或训练),并重点考虑硬错误、软错误或漂移 3 类常见问题。通过结合操作者反馈修复具体组件错误,以解释和改善模型性能,从而维护部署到下游任务的模型。

此外,在采用基于监控的调试方法时,研究人员通常会面临多源数据整合的复杂性问题。当多个数据源同时观测同一对象时[75],由于各数据源采集条件存在差异,往往导致同一对象属性出现多个相互冲突的值。针对这种情况,需要分析多源数据间的关联与矛盾,为每个对象推断出最可能为真的值,即真值发现[76-77]。

2.3 基于数据注入的调试方法

基于数据注入的封闭式数据调试方法通常通过向现有训练数据中加入额外训练数据来修正训练数据,以提升模型在未见过的数据上的泛化能力。这类方法的重点在于发现原始测试数据集D_{test}与原始训练数据集D_{train}之间不一致的子集,并通过获取、生成或修正新的训练数据集D'_{train}重新训练模型$M(\theta)$,获得M_{debug},从而提高M_{debug}在D_{test}上的性能。

数据注入的核心思想是将应用生成对抗网络(generative adversarial networks, GAN)[78-79]或已有数据增强技术[80]获得的额外数据直接加入训练集D_{train},从而消除由训练数据不足造成的模型偏差。机器学习模型需要在训练数据上训练,然后再扩展到训练数据之外的真实数据上应用,然而,真实世界数据不可能与训练数据完全相似。因此,在该情况下,很可能出现训练数据不丰富而导致测试数据效果不佳的现象。目前数据注入在计算机视觉(computer vision, CV)[2]领域得到广泛关注和研究,旨在通过调用生成模型在随机采样分布的基础上增强图像来获得新数据,以缓解训练数据单一问题。

Singla 等人[81]通过重新收集来扩充原始训练数据,应对训练数据无法充分覆盖部署后真实数据的场景。具体地,假设存在一个错误测试分布,它表示图像分类模型在真实场景上分类错误,这源于训练数据中未涵盖与测试数据相似的样本,使模型无法有效学习这些特定模式。由此,一个直观的方法 DCD 是直接获取与错误测试分布相似的额外样本,但这一举措的成本较高,所以通过获得错误测试分布的子集,并利用第三方弱标签数据匹配与错误测试分布子集相似的图片,加入和创建一个修复训练数据集。Motamedi 等人[82]认为数据的规模和可用性是训练机器学习模型的核心,通常对于小规模数据集来说很难在其上获得较优模型。因此,提出 DOP 框架,通过检测、删除和均衡等一系列优化策略来帮助消除无效数据,并设计一种条件生成对抗网络来合成额外训练数据,以缓解数据稀缺和质量引起的问题。Chen 等人[83]提出一种高度自动化的数据调试框架 HiBug,它利用大型预训练模型(例如 ChatGPT[84])来识别表现不佳的具有共同视觉属性的数据样本,并以此为调试依据来生成修复模型的训练数据。Malon 等人[85]研究关于 Transformer 模型在自然语言理解上的快速调试方法 FFD,其通过从原始训练数据中选出误导模型行为的错误数据,并使用它们进行少量步骤的模型更新,避免完全重训练,在保证模型整体准确率的前提下降低错误数据对模型行为的误导。Dang 等人[86]提出 TRT,它首先用原始文本训练数据微调 Transformer 等模型,然后从微调使用的文本数据提取代表性统计特征(如句子长度、标签分布),再将上述统计特征作为输入、Transformer 等模型的性能作为标签,用于训练一个外部预测模型,通过外部模型确定影响模型鲁棒性的统计特征,并据此调整训练数据以提升模型的鲁棒性。

此外,研究人员探索重新收集来增设训练数据时,需借助图片相似度匹配[87-89]来实现收集过程,并通过训练数据的整合方式[90-93]来完成数据集的创建。因此,如何度量图片间相似性以及设计高效的数据集,也是促进数据调试的重要一环。同时,数据增强技术[61]更是缓解训练数据稀缺问题的关键。

2.4 基于探测数据转换的调试方法

基于探测数据转换的封闭式数据调试方法希望以一种自动检查的方式深入探测机器学习预处理过程中的特定数据输出,向操作者提供程序中存在的数据转换问题,并突出数据转换问题的根本原因。这类方法不同于只关注原始输入数据的方法,它更倾向源于机器学习模型在数据预处理期间产生的错误,在与原始数据来源尽可能近的位置减少这种错误,并开始训练模型$M(\theta)$。

Grafberger 等人[94-97]设计并开发了 mlinspect 库,协助操作者自动化检测机器学习管道中预处理阶段的数据转换错误。具体来说,mlinspect 库从应用 Scikit-learn[98]和 Pandas[99]等库的管道中抽取逻辑运行计划,通过有向无环图来建模与关系操作符相结

合的预处理工作流程，并实现自动对照程序代码来跟踪其中某一操作对数据的影响。由此，mlinspect库帮助操作者更容易地查看已部署机器学习模型中的数据转换错误。

为方便检查机器学习管道初期出现的技术偏差，许多研究[100]致力于将相关方法集成到统一的模型开发工具内。随着以数据为中心的机器学习的广泛应用，数据科学也逐渐成为模型开发工具的核心组成部分之一，并且这些工具已经实现通过可视化和连续交互的形式[101]来调整偏差。

2.5 基于数据约束的调试方法

基于数据约束的封闭式数据调试方法指理解数据内部隐藏模式，并将其作为数据规范的外部约束，以帮助识别和调试错误数据。这类方法通过手动指定或高效枚举外部约束来检测测试数据D_{test}与D_{train}的不一致。其重点是遵循不同历史数据之间的关系模式来设计约束，利用约束匹配输入数据，返回无法匹配约束的错误数据，从而帮助操作者确认和修正错误数据。根据不同场景所需约束不同，可将现有数据约束分为定性约束和定量约束2类。

2.5.1 定性约束

定性约束依据从历史表格数据中获取到的属性间关系，来衡量当前元组在1组属性间关系上的违反程度，从而增强表格数据中的元组。机器学习模型所需的结构化数据一般存储于传统数据库管理系统，而传统数据库管理系统采用表格形式存放数据，为保证数据的正确性和有效性，设计和制定约束来预防数据错误导致的系统故障或下游应用失效。

为应对上述情况，最近研究相继提出多种约束策略，例如松弛函数依赖[102]、条件函数依赖[103]、顺序依赖[104]以及否定约束[105-106]等。其中，Ding等人[102]提出了松弛函数依赖(relaxed functional dependency, RFD)，通过松弛函数依赖来灵活描述数据的复杂关系，并设计算法将松弛函数依赖映射为统一关系。具体指某些属性可以确定其他属性的值；反过来，一旦其他属性无法由这些属性推出，则会被认定为错误数据，以供操作人员修改。为了弥补松弛函数依赖忽视特定数据语义的缺陷，条件函数依赖(conditional functional dependency, CFD)被提出。Jiang等人[103]通过强制使用相关语义的变量值来捕捉数据的规范性。实际生活中充满有序数据，但该方法无法应对这类数据，因此，Golab等人[104]设计一种描述有序数据语义的顺序依赖(sequential dependency, SD)来协助识别存在质量问题的数据。为了解决之前约束无法满足实际业务规则的数据问题，Parisi等人[105]采用否定约束(denial constraints, DC)这一技术。该技术要求任一元组不能同时满足否定约束集合中某一约束的任意谓词，以更复杂丰富的语义来有效发现错误元组。

然而，由于现实世界的复杂性和动态性，手动设置的简易约束已不再适合不断变化的数据语义。在研究这类数据约束时，人们逐渐将注意力转移到如何快速和自动挖掘约束，或是增量数据集上的约束发现[105]等工作。

2.5.2 定量约束

由于定性约束往往适用于文本数据且表达属性间语义有限，数值属性中的错误无法通过建模这种定性约束来识别，而这种错误恰恰可能是导致机器学习模型失效的原因。因此，定量约束通过训练数据提供的数值属性间算术关系，来量化机器学习模型在测试数据上预测结果的信任程度，从而推断出使模型失效的元组数据。

为了解决上述问题，Fariha等人[28]希望弥补函数依赖无法提供关系的参数表达式、否定约束过于复杂以及其他统计方法需要额外先验知识的缺陷，提出一种通过提取数值属性间算术关系来捕捉测试数据与训练数据间不一致性的数据约束。具体地，首先在训练数据上发现一组能够代表大多数数值属性关系的一致性约束(conformance constraints, CC)$\varphi:=(lb\leqslant P(D_{\text{train}})\leqslant ub)\vee(\varphi_1\wedge\cdots\wedge\varphi_k)$，这种一致性约束潜在地蕴含于模型内部，而违反一致性约束很可能导致建立在此一致性约束上的机器学习模型失败。由此，可以将测试数据中元组数据违反一致性约束的偏差作为信任模型预测该元组数据的代理。Ding等人[107]提出了一种新型的数据约束方法RC，融合了对数值属性间及其时序关系的约束支持。该方法首先通过分析数值属性间的约束关系识别时序数据中的错误，随后利用时序约束对识别出的错误数据进行修复。该双重约束机制有效提升了时序数据质量处理的准确性与可靠性。

实际上，定量约束的研究旨在提升人工智能系统的可信度。基于这一目标，可靠机器学习[108-109]和人工智能系统安全性[110-111]等方面的研究也在日益吸引数据调试领域研究人员的兴趣。

2.6 小 结

本节介绍并总结了封闭式数据调试，相关方法可继续分成基于测试的封闭式数据调试、基于监控的封闭式数据调试、基于数据注入的封闭式数据调

试、基于探测数据转换的封闭式数据调试和基于数据约束的封闭式数据调试5类，表4总结了对应子类的代表方法、错误类型、应用方式、数据可扩展性、适用范围以及优缺点。其中，错误类型是指数据问题所属类别(参见表3)，应用方式是指调试方法是否以自动化形式实施，可扩展性是指调试方法是否支持增量数据，而适用范围则是指调试方法是否适用于不同量级的数据集和模型。

Table 4 Closed Data Debugging

表4 封闭式数据调试

类别	子类	代表方法	错误类型	应用方式	可扩展性	适用范围	优点	缺点
测试	连续测试	DDCT[54]，Databugger[65]	异常值 模式错误	人工	有增量	大型 数据集	数据更新时检查，易调试	特定于系统
	单元测试	D-API[68]，CheckCell[69]	模式错误 格式错误	无人工	有增量 无增量	大型 数据集	可用于半结构化数据	需要验证函数，速度慢
监控		monitor[55]，REIN[56]	格式/模式错误 缺失/重复数据	半人工	无增量	大中小 型数据集	考虑数据错误的异质性	无法发现具体错误数据
数据注入		DCD[81]，DOP[82]，HiBug[83]， FFD[85]，TRT[86]	数据漂移 缺失数据	无人工	无增量	小型 数据集	考虑无效数据的存在	样本多样性提升有限
探测数据转换		mlinspect[94-97]	数据转换错误	无人工	有增量	大中小 型数据集	自动捕捉数据转换错误	仅用于预处理，成本大
数据约束	定性约束	RFD[102]，CFD[103]， SD[104]，DC[105]	模式错误	有/无人工	有增量	中小型 数据集	有效捕获真实数据语义	仅用于表格数据
	定量约束	CC[28]，RC[107]	数据漂移	无人工	有增量	中小型 数据集	量化数值属性间算术关系	仅用于数值数据

基于测试的封闭式数据调试在诱因考量方面表现全面，但通常要求操作者具备较高的专业素养，并提供充足的调试资源。进一步地，基于测试的封闭式数据调试根据测试方式的不同，又可分为连续测试和单元测试2个子类，它们均适用于大型数据集。其中连续测试可在数据更新过程中进行持续监测，但需要人工验证导致警报的错误；单元测试则无需人工干预，但存在不可扩展性的限制。

与基于测试的方法不同，基于监控的封闭式数据调试对操作者的专业素养要求较低，但仍需半人为式地引入领域知识。该方法仅在无增量数据环境中各类数据集上执行从检测到验证的紧密流程，且调试期间无法确定致使模型性能下降的具体数据。

基于数据注入的封闭式数据调试虽然计算成本较低且效率较高，但仅适用于图像和文本数据。另外，这类方法无法在增量数据上开展，且不适用于大型数据集。

基于探测数据转换的封闭式数据调试虽然实现了自动化检查，但开发和维护难度较大。该方法主要针对预处理阶段由过滤器或连接等操作引发的数据错误，适用于具有增量特征的各类数据集。

基于数据约束的封闭式数据调试能够对特定任务设计个性化约束，并在元组级别实现调试。然而，所设计的约束适用范围极其局限，针对不同数据集或任务需要不断更新获取新的约束。该方法依据处理数据类型不同进行归类，其中定性约束和定量约束2个子类可在中小量级且不断增加的数据上执行，但需要人为预先提供外部约束。

整体来看，封闭式数据调试能够在模型未知且性能要求不高的场景中发挥重要作用。然而，由于不涉及模型内部结构，该方法无法判定所检测错误的有效性，因此在某些高精度要求的场景中可能存在局限性。

3 浸入式数据调试

浸入式数据调试方法通常探究模型的执行管道(即特征工程、训练和评估)，而不再重点关注输入。依据探究模型的策略，此类方法可以分为：1)扰动。依据推测的故障原因扰动配置或训练数据，并通过模型性能的变化来判定真实错误。2)模型推演。运用公式推导影响模型性能的最小数据单元。3)对比。在模型内部，采用不同的对比方法分析影响其输出的原因。

3.1 基于扰动的调试方法

基于扰动的浸入式数据调试方法通过扰动管道中的配置或训练数据，识别导致模型性能变化的数据错误。与后面基于模型推演和对比的方法不同，该方法先扰动数据，再通过训练或近似训练评估影响，从而定位错误。具体来说，根据扰动对象的不同，基于扰动的调试方法继续分为迭代故障推断和影响分析。

3.1.1 迭代故障推断

尽管之前也有研究集中于管道调试方面[112]，但它们大多数的关注点是分析和识别历史执行管道中的错误，忽视了新管道的正确执行对推断这类错误的促进作用[113]。而基于迭代故障推断的浸入式数据调试侧重迭代更新各模块的配置，测试新管道，推断最小错误。

具体地，基于迭代故障推断的浸入式数据调试方法，首先执行原始的训练管道，推测管道中可能出错的配置（数据、计算方式或操作顺序等）；接着，更改配置并运行更改后的新管道，通过评估更改后管道的性能确定是否迭代执行故障推断过程。

过去的几十年里，软件调试[114]得到广泛研究，其中很多研究的基本机制是将运行失败的代码与拥有相似结构却运行成功的代码进行比较，从而推断代码中的错误。受到软件调试的启发，并且考虑到之前工作的缺陷，Lourenço 等人[115-116]提出 BugDoc，迭代推测管道配置（数据、计算方式或操作顺序等），采用扰动的方式来发现并解释错误配置，进而实现数据调试的目的。具体地，当存在指向输入数据作为错误的根本原因时，首先运行 BugDoc 解释输入数据的元特征和统计数据，这些数据可以揭示有关管道性能的有意义的相关信息，进一步采取组测试方式来识别具体的错误数据子集[117]。之后，手动扰动错误数据子集形成新配置并替换管道中对应的旧配置，执行和评估每次替换配置后的新管道的性能，从而判断迭代的有效性并决定是否继续。

此外，这类方法依赖于配置空间，过大的配置空间会降低其应用效果。为控制计算成本和执行时间，有效搜索配置空间的研究同样重要[118-120]。

3.1.2 影响分析

影响分析是统计学中一种传统有效的策略，它利用近似算法估计训练数据受到扰动后对模型结果的影响，从而在给定模型结果的情况下可确定具有“责任”的训练数据。影响分析通常基于反事实思想：在训练数据集中去除某个训练数据，亦或是轻微改动某个训练数据的大小来观察模型结果的变化程度[121]。

Koh 等人[121]将用于统计学的影响函数（influence functions, IF）扩展到机器学习，用于分析模型预测的成因。影响函数描述了当某个数据点从数据集中被去除或修改时，模型参数相应的变化。在机器学习中使用这一概念来调试与模型结果相关的有害训练数据。然而，传统方法通过依次扰动数据并重新训练模型来定位错误，但成本高昂，因此，影响函数通过 2 阶优化技术近似参数变化（如梯度），评估训练数据扰动对模型的影响，从而识别关键数据。具体地，在某个训练数据的权重上增加一个无穷小的扰动值，然后利用影响函数来估计其影响。通过分析模型参数在数据扰动下的变化，识别对模型性能产生显著负面影响的数据点，从而确定错误数据。针对现有基于随机梯度下降函数（stochastic gradient descent, SGD）及其变体训练的机器学习模型，Hara 等人[122]重新定义一种影响函数，称之为随机梯度下降影响函数（stochastic gradient descent influence functions, SGD IF）。SGD IF 通过近似算法估计数据变化对 SGD 的影响，直接推出决定模型行为的数据，而不需要重新训练模型。具体来说，返回随机梯度下降训练过程中各个阶段的参数，并结合每一阶段持有的中间模型来判定对模型行为负有“责任”的数据。Xie 等人[123]提出修正模型的错误预测，需要了解导致错误预测的根本原因，即有害的训练数据。为此，设计 RNNrepair 方法，首先应用聚类捕捉训练数据表现出的状态行为，在抽象的状态行为上通过影响函数精准识别有害训练数据，并评估注入与训练数据相关的额外数据对模型结果的影响程度，从而反推调试的有效性。Lee 等人[60]关注更多的训练数据问题，如训练数据中的噪声、稀缺的训练数据以及不平衡的训练数据等，提出一套完整流程 A&V 来解决上述问题。进一步来说，首先利用影响函数删除使模型变差的训练数据，接着通过已有数据增强技术针对性地扩充剩余数据，并借助对比学习来训练模型和进行 2 次数据删除，最后不断交替数据删除和数据增强来获得新的训练数据。Aamir 等人[58]扩展了 Koh 等人[121]的研究，探索模型的单一层（explore model’s single layer, EMSL）行为。具体来说，其目标是通过分析损失函数的扰动来理解源自训练数据表示的机器学习模型（VGG16）的行为（例如分类）。为此，进一步计算并确定每个训练数据中特征在模型各层的影响程度。Wang 等人[124]提出的 InfEmbed 方法，其核心机制是切片发现方法（slide discovery method, SDM），通过影响函数识别受类似训练数据影响的测试数据。Deng 等人[125]提出 MisDetect，在模型训练过程中检测标签错误的数据。具体而言，MisDetect 首先利用早期损失来迭代地识别和删除标签错误的数据，在此过程中引入影响函数对上述识别结果进行验证。在早期损失失效后，MisDetect 对于剩余仍无法确定的数据，提取一个高置信的核心子集[126]进行标签学习，并用标签模型的泛化能力对剩余数据做出判定。

影响函数分析[127-129]通过识别和纠正重要错误输入数据来改善模型性能,是数据调试领域中被广泛研究的核心技术之一。这类方法基于影响函数,能够应用于非凸或不可微的深度神经网络,同时保持一定准确性。然而,影响函数主要衡量局部数据变化对模型的影响,当模型需要保持相对稳定时,若全局数据发生显著变化,则影响函数的效用可能大幅减弱,甚至失效。

3.2 基于模型推演的调试方法

模型推演的浸入式数据调试方法通过公式化模型的预测过程,探究模型在评估阶段出现错误的根本原因。这类方法通常需要指定一类已观察到的错误类型,并通过公式推导来识别出对模型损失影响最大的错误训练数据项。不同于直接关注训练数据与测试数据不一致的封闭式数据调试,模型推演侧重于从模型在训练数据和测试数据上预测差异的角度来确定使模型产生错误的训练数据项。

Cadamuro 等人[57]提出了一种调试方法 OFG,重点关注训练数据中已知错误类型导致测试数据性能差的原因,并且仅致力于公式推导机器学习管道的训练阶段。具体地,利用公式推演模型的调试过程可描述为:通过考虑训练数据中的已知标签错误,并将最小化模型损失函数与修复复杂度作为约束条件,寻找一组训练数据子集,以减轻测试数据上的预测错误。值得注意的是该训练数据子集除了应能改善模型性能,还需保持尽可能小的规模。然而,对于更复杂的模型和训练过程(意味着更复杂的公式推导过程),例如随机森林和反向传播的神经网络等,上述调试方法将不再具备应用条件。在这种情况下需要直接通过更改训练数据和重新训练的方式,达到优化损失函数并改善模型性能的目的。由于训练数据空间极为庞大,迭代修复训练数据的方式同样不再适用,因此 Cadamuro 等人[57]提出了一种启发式模型相似度方法,来衡量修复训练数据与测试数据之间的差异。Zhang 等人[130]提出一种新的算法 DUTI,可以检测到错误的训练数据项,并对其更正错误标签。为此,DUTI 首先利用机器学习算法的相关知识,寻找训练数据集中的最小变化集,以便进一步从修正训练数据集上学习的模型能够正确预测可信任项的标签。之后,将被修正标签的训练数据项标记为错误,其标签由专家检查准确性。

3.3 基于对比的调试方法

基于对比的浸入式数据调试是指在模型内部通过某种分析策略来跟踪影响 $M(\theta)$ 性能的 D_{train} 子集。与基于模型推演的浸入式数据调试不同,这类方法可以在不提供特定错误类型的情况下,通过分析模型内部特征推断出具有“责任”的错误数据,从而使操作人员更有针对性地调试错误数据,降低直接使用数据生成和增强方法带来的盲目性。基于对比的浸入式数据调试的核心在于利用基于对比的方式探索模型内部影响其输出的根本原因,以便修复策略实施调试。进一步说,不同的分析策略可细分为状态分析和轨迹分析。

3.3.1 状态分析

状态分析最早源于软件调试领域,在以往的软件调试中,将操作者比较程序执行失败和成功对应状态的过程称为状态分析,其差异可有效帮助操作者找到和修复程序错误[131]。而在数据调试中,错误数据所造成的不良行为可能是由模型内一大批神经元导致的。另外,由于神经元难以解释,操作者无法像软件调试中定位和修复程序一般,直接识别并调整模型内部的权重参数。而简单地增加额外的训练数据或者仅将引发不良行为的数据加入训练集,可能不会改善(甚至降低)模型性能。因此,受软件调试领域的激励,数据调试领域引入的状态分析利用热图(heat maps)刻画某隐藏层上神经元对正确分类和错误分类的重要性,通过执行差异分析并生成差异热图来识别错误且具有影响力的神经元,为后续选取数据以修复错误神经元提供指导[132-133]。

Ma 等人[132]提出了一种基于状态分析的调试方法 MODE,它首先诊断模型的状态差异,分析和识别致使模型性能不佳的隐藏层及其所在层的具体神经元,并执行类似软件调试中回归测试的数据修复操作,将影响神经元故障的输入样本加入训练数据,从而使调试工作更具针对性。Feng 等人[59]提出一种定位对抗攻击样本的框架 DeepFeature,它能够确定那些不能被模型转换的弱特征,以便 DeepFeature 选择包含弱特征的样本(数据)进行对抗性再训练。

另外,这类方法的有效性得益于状态差异分析,而软件调试领域已具有众多相关工作,例如 GDB、增量调试[134]和基于频谱的错误定位[135-136]等,这促使状态差异分析同样成为数据调试领域的研究热点之一。

3.3.2 轨迹分析

轨迹分析同样借鉴于软件调试领域。在软件开发中,操作人员将通过对比程序正确和错误执行的轨迹来定位错误的过程称为轨迹分析,这一分析方法有助于操作人员调试错误并恢复程序的正常运行[137-138]。类似地,数据调试提出一种跟踪模型执行

来定位错误的轨迹分析，但与软件调试不同，由于数据与模型内部的抽象状态之间存在复杂的依赖关系，因此数据调试中的轨迹分析旨在利用抽象状态发现和修正错误数据来改善模型性能。尽管轨迹分析与状态分析都试图根据模型内部行为来探究导致性能下降的数据，但状态分析一般只适用于卷积神经网络等前馈神经网络。然而，针对循环神经网络的数据调试方法并不多，由此促进了轨迹分析的出现和发展。

Tao 等人[62]提出一种自动诊断有问题的数据嵌入的框架 Trader，通过回溯循环神经网络模型正确和错误行为对应的执行轨迹来实现。该框架的诊断结果可用于指导数据嵌入(embedding)的修复。Trader 的核心在于强化数据嵌入，以降低这种偏差导致模型性能不佳的可能性。具体来说，由于循环神经网络严重依赖于文本编码成数值的数据嵌入，因此，为对其开展调试，将循环神经网络看作具有特殊语义的“程序”。由此，利用类似软件调试中轨迹分析的技术来识别错误的嵌入向量维度，并设计一种扰动嵌入向量维度的方法来强化模型在错误嵌入向量维度上的鲁棒性。Du 等人[139]为表征循环神经网络的内部行为，将其抽象成一个状态转换系统。具体地，先将循环神经网络建模为离散时间马尔可夫链(discrete-time Markov chain, DTMC)来捕捉其内部状态，通过离散时间马尔可夫链对训练后的循环神经网络进行内部轨迹剖析，从而基于设计的轨迹相似度指标和覆盖准则探究影响循环神经网络行为的数据因素。值得注意的是，这些轨迹记录了模型训练阶段的状态和数据转换，并且为了提高实践的可行性，利用主成分分析(principal components analysis, PCA)[140]来减少状态的维度。

3.4 小 结

本节介绍并总结了浸入式数据调试，对应方法可细分为基于扰动的浸入式数据调试、基于模型推演的浸入式数据调试和基于对比的浸入式数据调试 3 类，表 5 是对上述方法在详细指标上的概述。

Table 5 Immersive Data Debugging

表 5 浸入式数据调试

类别	子类	代表方法	错误类型	应用方式	可扩展性	适用范围	优点	缺点
扰动	迭代故障推断	BugDoc[115-116]	模式错误	无人工	有增量	小型参数空间的计算管道	可并行确定计算管道中错误	参数多时运行成本高
	影响分析	IF[121]，SGD IF[122]，RNNrepair[123]，A&V[60]，EMSL[58]，InfEmbed[124]，MisDetect[125]	数据漂移/稀缺标签/特征错误	无/半人工	无增量	中小型数据集	无需重新训练模型，效率高	无法衡量全局变化效果
模型推演		OFG[57]，DUIT[130]	标签错误	有人工	有增量	大中小型数据集和中小型模型	可用于不同数据类型或模型	仅适用于监督学习
对比	状态分析	MODE[132]，DeepFeature[59]	特征错误/对抗攻击样本	无人工	无增量	小型数据集和小型模型	不易引入新错误	需要提前明确错误类型
	轨迹分析	Trader[62]，DeepStellar[139]	嵌入错误/对抗攻击样本	无人工	无增量	中小型数据集和小型模型	易于理解模型行为的由来	仅针对 RNN 模型

基于扰动的浸入式数据调试通过扰动管道配置或训练数据来定位影响模型的错误。根据扰动对象的不同，分为迭代故障推断和影响分析。迭代故障推断自动搜索、更新和组合管道配置来改善模型性能，可以扩展到任意机器学习管道上执行，但在调试期间需要对管道中的每个模块枚举大量参数，导致时间成本随参数数量线性增长，因此不适用于参数空间较大的模型。影响分析利用影响函数估计数据扰动对参数的影响来发现错误数据，适用于中小型数据集，但仅对衡量局部变化有效。

基于模型推演的浸入式数据调试需要人工推导模型内部训练过程来确定错误数据，虽然可以被用于增量数据环境中的各量级数据和各类机器学习模型，但却不适用于内部复杂的大模型(例如深度学习模型)。目前，这类方法仅在监督学习场景下对传统机器学习模型进行了简单尝试和探索。

基于对比的浸入式数据调试虽然填补了之前方法无法理解模型错误行为的空白，但仍存在适用范围有限的问题。根据分析模型内部的不同方式，该方法可进一步分为状态分析和轨迹分析 2 个子类，其中状态分析通过调整影响神经元的数据来提高模型性能；轨迹分析则专注于跟踪、对比和修正执行轨迹之间的差异来调试模型。这 2 个子类主要在无增量数据环境中的中小型数据集上实现自动化执行。另外，应用状态分析和轨迹分析时，需要考虑参数空间大小，因此它们仅限于小型模型。

整体来看，浸入式数据调试适用于模型已知且性能要求高的场景，因为这类方法可以根据回溯训

练过程来精准定位错误,但需要明确模型内部结构。

4 混合式数据调试

混合式数据调试是一种介于封闭式数据调试与浸入式数据调试之间的方法,不仅关注输入数据的正确性,而且考虑模型内部结构,通过二者结合的形式来实现数据调试。与封闭式数据调试直接聚焦输入数据和浸入式数据调试侧重模型执行过程不同,这类方法通过调整用户告知和管道模块探索到的数据问题来改善模型性能。基于此,本文将此类方法细化为2种,分别是基于投诉和管道模块的调试方法。

4.1 基于投诉的调试方法

用户已知并提供下游应用查询结果中存在的错误,称之为投诉(complaint)[141]。基于投诉的混合式数据调试通过用户与调试系统交互来反馈导致模型行为错误的训练数据,便于进一步干预相应训练数据。封闭式数据调试中的单元测试和浸入式数据调试中的影响分析等已有研究[68,121],尚未考虑下游应用产生的投诉,只是尝试在模型训练前识别错误训练数据或者根据模型性能变化定位错误训练数据,但它们无法解释直接提供给用户并用于产业决策的模型预测结果。而基于投诉的混合式数据调试将下游应用产生的投诉作为调试训练数据的指导,填补了现有研究的空白。与浸入式数据调试中对比模型正确和错误执行轨迹上差异的方法不同,这类方法直接根据投诉的轨迹来确定改善模型不佳所需的最小训练数据。

Wu等人[51]考虑到数据库管理系统(database management system, DBMS)存储模型预测所依据和产生的数据,因此有必要将数据调试引入SQL查询。由此,Wu等人[51]提出基于投诉的训练数据调试系统Rain,致力于解决错误训练数据造成的意外查询结果。Rain依照用户指定查询结果的投诉来返回一组最可能的错误训练数据,通过删除它们来解决投诉问题。为了实现上述目标,Rain摒弃重新训练这种指数级计算成本的解决策略,而是基于机器学习中影响函数的启发式方法来近似训练过程。然而,Rain虽然有效,但运行时间却依赖于训练数据和模型大小,导致Rain的平均调试时间过长。此外,Rain尚未注意到调试成本与投诉无关、现有模型普遍过度参数化,以及忽视模型参数对删除训练数据或模型结果对参数变化的灵敏度等缺陷。Flokas等人[142]在Rain基础上进行拓展和优化,提出的Rain++考虑上述见解,并避免访问不被披露的模型训练基础设施(数据、参数以及模型定义)等问题。具体地,Rain++以离线形式预先计算损失函数逆Hessian运算中的少量特征值和特征向量,并通过上述特征值进行快速数据调试,以提高效率和可扩展性。

4.2 基于管道模块的调试方法

基于管道模块的混合式数据调试通过利用第三方开发工具调查程序中间步骤处理的数据,来找到模型错误的原因。与封闭式数据调试中只关注机器学习模型预处理阶段的数据转换探测技术不同,这类方法支持在机器学习管道内跨模块来发现导致模型结果错误的中间数据问题。

Rezig等人[143-144]提出一种端到端的调试系统Dagger,并设计了一种类似SQL的语句DQL(dagger query language)来调试机器学习管道中与数据错误相关的问题,例如,代码执行造成的数据转换错误、训练数据不足影响模型合理推断,或噪声数据导致模型性能不佳。尤其,Dagger同时支持对机器学习管道中特定模块之间的封闭式数据调试和模块内的浸入式数据调试。此外,为了弥补当前尚未有融合数据调试与机器学习开发研究的空缺,Rezig等人[145]开发了一个交互式端到端的系统Data Civilizer 2.0,该系统旨在帮助操作人员完善从数据准备到模型预测的整个管道。具体地,操作人员依据可视化度量指标判断模型内部是否存在数据问题,通过Data Civilizer 2.0跟踪模型执行过程,并检查管道内部处理产生的中间数据来发现出错位置,从而使得数据级别的调试功能加速了机器学习管道的迭代构建过程。

4.3 小 结

本节介绍和概述了混合式数据调试,这类方法进一步分成基于投诉的混合式数据调试和基于管道模块的混合式数据调试2类,表6总结了相关方法在

Table 6 Hybrid Data Debugging

表6 混合式数据调试

类别	子类	代表方法	错误类型	应用方式	可扩展性	适用范围	优点	缺点
投诉		Rain[51], Rain++[142]	标签错误、特征错误	半人工	有增量	大中小型数据集/中小型模型	允许用户交互	响应时间长、计算成本高
管道模块		Dagger[143-144], Data Civilizer 2.0[145]	模式错误	半人工	无增量	中小型数据集/中小型模型	通用性强	运行开销大

各指标上的具体情况。

基于投诉的混合式数据调试根据人工提供的错误来调试模型。这类方法可以在增量数据上扩展，且适用于中小模型和各类数据集，但具有响应时间长和计算成本高的缺陷。

基于管道模块的混合式数据调试利用第三方工具和人工审查中间数据来查找错误。这类方法能够支持大多数机器学习管道的需求，但局限于无增量数据环境下的中小模型和数据集，并且需要花费较高的运行开销。

整体来看，混合式数据调试能够改进封闭式数据调试在错误发现上的可靠性问题，同时满足浸入式数据调试对明确模型内部结构的需求。由此，混合式数据调试适用于模型已知且性能要求高，并且提供人工支持的场景。然而，这种方法通常需要持续的人员参与，这可能增加额外的成本。

5 数据调试方法评估

本文将数据调试方法分为封闭式、浸入式和混合式三大类，其中各大类别的代表性方法如表 7 所示。通过对这些方法的研究，可以发现不同数据调试方法往往对应不同的评价指标和数据集，这些评价指标和数据集涵盖了多领域、多角度和多场景。

Table 7 Main Research Directions of Data Debugging

表 7 数据调试主要研究方向

研究方向	相关代表性方法
封闭式数据调试	DDCT[54]，D-API[68]，monitor[55]，DCD[81]，mlinspect[94-97]，RFD[102]，CC[28]，FFD[85]
浸入式数据调试	BugDoc[115-116]，OFG[57]，MODE[132]，SGD IF[122]，RNNrepair[123]，MisDetect[125]，Deep Stellar[139]
混合式数据调试	Rain[51]，Rain++[142]，Dagger[143]

本节首先概括和总结在数据调试领域常用的性能评价指标；其次，介绍常用数据集及其特点；最后，在相同数据集上探究不同数据调试方法对模型的改进效果，并开展相关实验验证。

5.1 评价指标

现有评价指标可分为测试充分性、模型有效性、输入输出差异性和时空成本 4 类，如表 8 所示。表 8 详细介绍本文所提数据调试方法中涉及的评价指标。

Table 8 Evaluation Metrics of Data Debugging Methods

表 8 数据调试方法评价指标

类别	子类	评价依据	应用该评价指标的数据调试方法
测试充分性	可用性	用例在不同数据集上测试成功、失败或超时的数量、占比或置信度	Databugger，D-API，SD，DC，DeepSteller
	覆盖率	用例验证数据集中属性的域、范围、基数、依赖关系等出现的比率	
模型有效性	精确率	真正被正确执行为正例与全部被执行为正例的个数之比	CheckCell，monitor，REIN，DCD，DOP，RNNrepair，A&V，DC，CC，BugDoc，OFG，MODE，IF，SGD IF，Trader，DeepSteller，Rain，Rain++，Dagger，Data Civilizer 2.0
	召回率	真正被正确执行为正例与全部本应被执行为正例的个数之比	
	*F*1 分数	精确率与召回率的调和平均值	
输入输出差异性	数值型误差	数据集部分修正后的值与数据集全部修正后的值的差	CheckCell
	属性型误差	累计用“1”所表示的修正属性值和用“0”所表示的未修正属性值	
时空成本	执行时间	数据调试方法 1 次执行消耗的时间	D-API，CheckCell，REIN，DOP，mlinspect，CFD，SD，DC，Trader
	占用空间	数据调试方法运行时所占内存的空间大小	

测试充分性是指验证手动或自动生成用于检查后续应用所运行数据的测试用例的充分性。其中，充分性是度量数据集完备性的概念，它包括可用性(utility)和覆盖性(coverage)。模型有效性是指下游模型在检查和纠正后的数据上取得的性能改善。其中，有效性由精确率(precision)、召回率(recall)和 *F*1 分数(*F*1-score)协同衡量，它们也是下游模型正确执行的互补指标。输入输出差异性是指修复部分数据与修复全部数据得到的剩余误差(error)。差异性是量化错误数据与真实数据不一致程度的指标，它包含数值型误差和属性型误差。时空成本是指整个调试算法或策略执行占用的时间和空间开销。

其中，由于数据是提高下游模型性能的核心驱动力，因此绝大多数方法都考虑了模型的有效性。在评价调试时间方面，根据策略中具体实施步骤的差异，采取不同评价方式。例如，增量调试执行时间，即每次在增加相同数量的数据上执行调试的时间；批量调试执行时间，即每次在相同数量但不同数据上执行调试的时间；平均调试执行时间，即多次在同一数据上执行调试的平均时间；单个调试执行时间，即每次在不同单元数据(如数据集中的一条记录或一个样本)或操作(如预处理中的连接)上执行调试

的时间。另外,仍存在一些延伸的评价指标[146-147],以不同角度量化数据调试方法的效果。例如平均绝对误差(mean absolute error, MAE),计算所有调试数据与真值之间误差绝对值的平均值;均方误差(mean square error, MSE),计算所有调试数据与真值之间误差平方和的平均值;均方根误差(root mean square error, RMSE),计算所有调试数据与真值之间误差平方和的平均值的平方根;平均相对误差(mean relative error, MRE),计算所有调试数据和真值之间绝对误差与真值的相对误差的平均值。上述指标均从调试策略对数据本身的直接影响出发,为研究人员提供判断调试数据可靠性的依据。

5.2 常用数据集

人工智能广泛用于医学影像[2]、智能机器[3]、生物基因[4]、金融服务[5]等不同领域,由此获得的机器学习模型是建立在图像、文本、结构化数据、时序数据等不同模式上的。而这些模式与其对应的机器学习模型可能需要不同的调试策略适配,因此,进一步了解数据对高效运用数据调试具有重要作用。

表 9 介绍数据调试研究中常用的数据集,并详细描述数据的类型、量级、所用数据调试方法以及简短概述。

Table 9 Data Sets and Their Characteristics

表 9 数据集及其特点

数据集	类型	量级	使用该数据集的数据调试方法	数据集概述
reddit-comments[148]	结构化数据	17 亿	D-API[68]	2015 年 reddit 网站评论数据集
DBpedia[149]	结构化数据	13 亿	monitor[55], TRT[86]	由维基百科文章提取数据集
Adult[150]	结构化数据	45 223	DUIT[130], SGD IF[122], REIN[56], Rain[51], Rain++[142]	美国人口普查收入数据集
ImageNet[151]	图像	1 400 万	DCD[81], HiBug[83], DeepFeature[59]	涵盖动物/植物等图像数据集
Roman-MNIST[152]	图像	2 162	DOP[82], A&V[60]	手写罗马数字图像数据集
MNIST[153]	图像	7 万	RNNrepair[123], MODE[132], SGD IF[122], DeepStellar[139], Rain[51], Rain++[142], DeepFeature[59]	黑底白字手写数字图像数据集
RSNA[154]	图像	13 236	InfEmbed[124]	美国放射学会骨龄数据集
Toxic[155]	文本	792	RNNrepair[123]	评论数据集
Hospital[156]	结构化数据	11.5 万	DC[105]	来自美国政府的医院数据集
Airlines[157]	数值数据	5.8 万	CC[28], MisDetect[125]	某地 2008 年航班信息数据集
Human Activity Recognition	结构化数据	75 万	CC[28]	人体活动相关指标数据集
Flight delay[158]	结构化数据	60 万	OFG[57]	包含航班推迟信息的数据集
CIFAR-10[159]	图像	6 万	MODE[132], SGD IF[122], Rain++[142], DeepFeature[59]	包含 10 种对象的图像数据集
IMDB	文本	5 万	Trader[62]	电影评论数据集
Yelp reviews	文本	600 万	Trader[62], TRT[86]	餐饮、住宿等评论数据集
SST-2	文本	6.7 万	Rain++[142], RNNrepair[123], FFD[85]	情绪分析数据集
EEG	时序	3.5 亿	Dagger[143-144], Data Civilizer 2.0[145]	脑电图片段数据集

5.3 实验测试

由于针对不同机器学习模型的数据调试方法通常可能需要依据机器学习模型和数据具备的特性来设计,一种数据调试方法一般只适用于特定类别的机器学习模型、数据和下游应用,通常不具有通用性。为准确评估不同数据调试方法的效果,理应在相同实验条件下进行比较。因此,基于上述情况,本节在相同机器学习模型和数据集下对不同数据调试方法展开实验探究。

5.3.1 实验设置

考虑到囊括方法的多样性和论文提供代码的状况,本文在浸入式数据调试和混合式数据调试的方法中分别选取代表性数据调试方法进行测试。所有实验均在 Nvidia TITAN V GPU 上开展,GPU 显存为 24 GB,所使用的 CPU 为 Inter® Xeon® Gold 5122 CPU,共 16 核。所有实验均在 Ubuntu 16.04 LTS 64 位操作系统上运行,CUDA 版本为 10.2。

本文在 MNIST 和 CIFAR10 数据集上对 4 类机器学习模型分别应用 4 种不同的数据调试方法,具体相关信息如下所述。MNIST 数据集包含 70 000 张具有 28 × 28 像素点的手写 0~9 数字灰度图像,其中,60 000 张用于训练,10 000 张用于测试,并采用将训练集中部分为“1”的标签翻转为“7”的形式引入错误;CIFAR10 包含 60 000 张具有 32 × 32 像素点的 10 类

彩色图像，其中，50 000 张用于训练，10 000 张用于测试，并将训练集中部分为“automobile”的标签改为“horse”来引入错误。本文选取浸入式数据调试中 MODE, SGD IF 和混合式数据调试中 Rain++, Rain 共 4 种数据调试方法，并探究它们在逻辑回归、全连接神经网络、卷积神经网络和残差神经网络 4 类不同机器学习模型上的效果。

5.3.2 实验方法

本文通过 3 种形式来对比 4 种数据调试方法在机器学习模型上的调试潜力。

1）原始测试。原始测试是指将在具有错误的训练数据集上获得的模型直接用于测试，以得到模型在 MNIST/CIFAR10 数据集上分类的原始测试准确率，并将其作为评估基准之一。其中，准确率是指预测正确的样本占总样本的比例，将其作为表征模型分类的评价指标。

2）随机干预。随机干预是指将在随机删除或修复错误后的训练数据集上获得的模型用于测试，以得到模型在 MNIST/CIFAR10 数据集上分类的随机干预准确率，并将其作为评估基准之一。

3）调试测试。调试测试是指利用数据调试方法识别和修复训练数据集中有“责任”的错误，再将在调试后的训练数据集上获得的模型用于测试，以得到模型在 MNIST/CIFAR10 数据集上分类的调试测试准确率，并将其与上述 2 个评估基准进行对比。

5.3.3 实验结果

本节首先详细分析表 10 和表 11，然后分别从泛化性、有效性和时效性 3 个方面总结实验。

Table 10 Accuracy of Four Data Debugging Methods on Four Types of Machine Learning Models

表 10 4 种数据调试方法在 4 类机器学习模型上的准确率

数据集	数据调试方法	逻辑回归			全连接神经网络			卷积神经网络			残差神经网络		
		原始测试	随机干预	调试测试	原始测试	随机干预	调试测试	原始测试	随机干预	调试测试	原始测试	随机干预	调试测试
MNIST	MODE	0.829	0.891	0.931	0.847	0.850	0.944	0.830	0.825	0.929	0.857	0.834	0.935
	SGD IF	0.838	0.863	0.929	0.805	0.831	0.934	0.833	0.992	0.993	0.868	0.813	0.966
	Rain++	0.716	0.714	0.977	0.587	0.612	0.893	0.716	0.737	0.920	0.824	0.826	0.912
	Rain	0.790	0.796	0.984	0.603	0.609	0.853	0.691	0.700	0.918	0.849	0.792	0.919
CIFAR10	MODE	0.517	0.522	0.603	0.572	0.578	0.700	0.780	0.762	0.823	0.793	0.785	0.828
	SGD IF	0.505	0.509	0.598	0.634	0.611	0.753	0.836	0.819	0.871	0.839	0.841	0.873
	Rain++	0.489	0.494	0.610	0.586	0.573	0.694	0.810	0.800	0.905	0.820	0.798	0.889
	Rain	0.534	0.530	0.627	0.605	0.620	0.718	0.794	0.795	0.899	0.816	0.801	0.897

Table 11 Time Consumption of Four Data Debugging Methods on Four Types of Machine Learning Models

表 11 4种数据调试方法在4类机器学习模型的时间消耗

min

数据集	数据调试方法	逻辑回归	全连接神经网络	卷积神经网络	残差神经网络
MNIST	MODE	9	60	100	117
	SGD IF	10	190	240	246
	Rain++	4.02	27	60	62
	Rain	5.5	31	65	73
CIFAR10	MODE	11	76	133	149
	SGD IF	10.5	228	269	291
	Rain++	6	57	80	83
	Rain	7	61	88	96

表 10 展示了 4 种数据调试方法 MODE, SGD IF, Rain++, Rain 在逻辑回归、全连接神经网络、卷积神经网络和残差神经网络 4 种机器学习模型上的准确率表现。通过对比原始测试、随机干预和调试测试下的准确率，可以得到：

1）调试测试与原始测试对比。对于每一种模型，在 MNIST 和 CIFAR10 上经过数据调试方法处理后的模型的准确率均明显高于原始测试下模型的准确率。上述结果说明数据调试方法能够针对性地处理错误数据，从而有效提升模型的性能。

2）调试测试与随机干预对比。尽管在 MNIST 和 CIFAR10 上随机干预错误数据也可能在一定程度上提高模型的准确率，但其提升幅度通常低于经过数据调试方法处理的模型。这一现象表明随机干预虽然可能会带来一些改善，但效果并不稳定，也远不如数据调试方法有效。

3）调试测试中不同数据调试方法对比。不同数据调试方法在不同模型上呈现出不同的效果。例如，

在逻辑回归上, Rain 取得了最高的准确率, 其在 MNIST 和 CIFAR10 上的调试后准确率达到 0.984 和 0.627; 而在全连接神经网络、卷积神经网络和残差神经网络中, MODE, SGD IF, Rain++, Rain 则在 MNIST 和 CIFAR10 上分别表现出了最佳性能。这一分析说明在选择数据调试方法时, 理应根据模型类别进行权衡。

表 11 展示了 4 种数据调试方法 MODE, SGD IF, Rain++, Rain 在逻辑回归、全连接神经网络、卷积神经网络和残差神经网络 4 种机器学习模型上的时间消耗。从表 11 可以得到:

1)不同模型的时间消耗。同一种数据调试方法在不同模型上的时间消耗存在差异。例如, Rain 在 MNIST 上调试逻辑回归和卷积神经网络消耗的时间分别为 5.5 min 和 65 min。这是由于不同模型的复杂度和计算需求不同所致。

2)调试测试中不同数据调试方法的时间消耗。不同数据调试方法在同一模型上的时间消耗也会有所不同。例如, 在逻辑回归中, Rain++通常具有最短的时间消耗, 而 SGD IF 则总需要更长的时间来完成整个调试过程。这提示在选取数据调试方法时, 除了考虑其提升准确率的效果以外, 还需要斟酌消耗的时间。

在对表 10 和表 11 进行具体分析后, 从泛化性、有效性和时效性 3 个方面对实验进行全面总结, 并得出以下结论。

1)泛化性

泛化性是指同一数据调试方法可在不同模型上实施, 获得的准确率高于原始测试下模型的准确率。

通过表 10 可知, 4 种数据调试方法均具备良好的泛化性, 能够在不同类型的机器学习模型上提高准确率。这代表上述数据调试方法具有跨模型的通用性, 可以被广泛应用于不同机器学习场景中。

2)有效性

有效性是指经过不同数据调试方法处理后模型获得的准确率高于随机干预下模型获得的准确率。

根据表 10 显示, 相比随机干预, 经过数据调试方法处理后的模型在准确率上均取得显著提升, 并且所有模型在调试测试下的效果均优于在随机干预下的效果, 这证明了数据调试方法的有效性。

3)时效性

时效性是指由数据调试方法调试不同模型所耗费的时间, 其中, 耗费时间是指从训练、识别、修复到评估过程的用时。

结合表 11 所示, 数据调试方法提升模型性能, 同时也会带来一定的时间消耗。同一数据调试方法在不同模型上耗费的时间不同, 这种差异会因模型不同而变化。而不同数据调试方法在同一模型上的耗费时间同样存在区别, 这种差异也会因数据调试方法不同而变化。由此可知, 在数据集一致的情况下, 影响数据调试方法时效性的因素主要取决于模型和方法本身 2 个方面。

上述分析和结论为数据调试领域中后续评估不同数据调试方法的调试潜力提供一定的参考依据。

6 挑战与展望

尽管当前数据调试研究在错误发现与修复机制、特定任务适配性等方面取得了初步成果, 但整体仍处于早期发展阶段, 面临多重挑战, 概括如下:

1)调试策略通用性问题

多数数据调试方法依赖于特定的模型类别、错误类型或数据种类, 适用于某一单一任务或场景(如图像分类中的标签错误、文本生成中的语义不一致)。缺乏跨模型、跨任务的统一解决方案, 导致在新任务或多任务环境中适用性和有效性大幅降低, 迁移能力有限。由于调试策略在通用性和灵活性方面存在显著不足, 不仅提升了数据调试的门槛, 还阻碍了该领域的推广与演进。

2)数据类型覆盖问题

当前数据调试研究主要集中在表格数据和图像数据, 对于在实际生产中广泛存在的文本数据、时序数据和图结构数据等关注较少。这些复杂数据通常伴随语义、上下文或结构层面的约束, 其调试策略更具挑战性, 现有策略难以直接适配。

3)调试过程自动化问题

现有数据调试方法往往需要人工先识别数据中存在的错误类型, 面对多种不同错误需要不同调试策略, 但选择这些方法通常靠人工判断, 而不是自动适配。调试流程呈现出高度碎片化, 缺乏集成且协同的工具链。这种方式对非专业用户不友好, 且在多类错误共存时效率与有效性难以保证。

4)评估体系标准化问题

不同任务采用的错误注入方式、实验数据集、评价指标差异较大, 导致各数据调试方法的有效性难以直接比较, 制约了方法间的公平竞争。这影响了该领域在理论、方法、工具等方面建立系统发展框架的进程, 阻碍了形成协调一致、相互促进的研究

体系。

5)分布式场景下的调试问题

随着联邦学习等分布式机器学习范式的兴起，传统集中式数据调试方法不再适用。这类分布式系统中，数据不可访问、训练过程分散，现有调试策略难以分析全局元信息，也无法对局部数据质量进行有效控制。

针对上述挑战，未来数据调试研究亟需从5个关键方向深入突破。

1)发展通用性强的调试策略框架

已有数据调试方法大多针对特定模型结构或任务目标，设计具有高度依赖性的策略，导致难以在新任务或多任务中快速复用。面对实际生产中多模型(如树模型、神经网络)和多任务(如分类、生成)共存的场景，研究人员亟需构建具有泛化能力的通用调试框架。该框架应实现模块化设计，使得错误发现与修复阶段可解耦、可组合，并具备良好的可迁移性，以支持从一个模型或任务平滑迁移到另一个模型或任务。在实现上可以借助因果分析、图模型、任务抽象建模等方式提升调试策略的通用性，从而达成跨模型、跨任务的灵活适配。

2)拓展复杂数据类型的调试能力

复杂数据(如文本、时序与图结构数据)在实际应用中同样广泛且重要，具有高语义密度、强上下文依赖和多维耦合等复杂特征，对调试策略提出更高要求。针对文本数据，调试策略需要处理语义抽取、上下文一致性维护、不平衡语料分布以及敏感词汇引起的模型偏差；针对时序数据，其存在维度间共现错误、数据漂移与错误片段混杂的问题，建议融合动态时序依赖性和多变量关系来应对；针对图结构数据，需在调试过程中考虑拓扑结构不变性和图中节点属性的合理性。因此，应加强对文本、时序和图结构等数据类型的调试方法研究，充分挖掘其语义、时序趋势等结构特性，以进行有效的调试。

3)实现调试流程的一体化与自动化

随着数据体量增大与错误类型复杂化，调试系统未来应具备“自动识别—自动选择—自动调试”的闭环能力。一方面，应建立可自动识别多种错误类型(如标签错误、特征错误、异常值)的检测机制，提升调试系统对混合型错误的感知能力；另一方面，需设计基于任务目标(如分类、生成)与上下文环境(如错误类型、数据种类)的策略选择机制，实现多种调试方法的智能联动。在实现上可融合元学习、监督学习等方式从历史调试经验中抽取模式，以增强调试的自动性、适应性与泛化性。同时，进一步可以考虑用户友好性，让非专业人员也能简单、迅速地完成高质量数据调试工作。

4)构建统一、权威的评估基准

数据调试方法的科学性与有效性依赖于系统化的评估框架。然而，目前缺乏具备广泛共识的测试基准，导致以往的研究结果难以复用，方法对比缺乏公信力。未来应重点建设具备以下3方面要素的评估体系：①代表性数据集。涵盖多模态与多任务的数据资源，且包括多类典型错误。②标准化注入手段。支持不同类型错误的可控注入和生成，以模拟真实环境中的错误数据形态。③多维度评价指标。覆盖检测准确性、修复有效性、模型性能、算法稳定性与计算代价等维度的指标。同时，应创立开放式评测平台与排行榜，促进研究成果的可持续对比与迭代优化。

5)开展联邦学习中的数据调试研究

联邦学习作为分布式协同训练的重要范式[160]，其调试策略尚待深入研究[161]。与集中式系统不同，联邦学习面临数据孤岛、本地模型异构与通信受限等问题。未来研究应聚焦于2类关键机制：①本地调试建模。在不泄露原始数据的前提下，基于本地元特征(如梯度统计、更新方向等)构建错误发现和修复策略。②联邦协同调试机制。在联邦通信中嵌入调试信号，通过聚合调试表示实现跨客户端的协同调试。进一步地，可结合联邦自监督在无标签或弱标签场景下挖掘错误并进行低资源调试。此外，需关注调试策略的通信效率及隐私安全，确保其在实际应用中的可部署性和可扩展性。这一深入研究将不仅提高联邦学习模型的稳定性与鲁棒性，而且有望拓宽数据调试在分布式协同学习中的操作模式。

7 总　　结

随着数据驱动的人工智能系统在各类实际场景中广泛落地，人工智能系统的性能关乎下游应用的成败。越来越多的研究表明，仅依赖优化模型架构和参数已难以持续带来性能的提升，而数据质量对模型性能的影响愈发显著。在此背景下，数据调试作为保障数据可靠性与模型有效性的重要环节，正成为智能系统开发中不可或缺的重要组成部分。通过对整个开发过程中的潜在数据错误进行精准定位和高效修复，数据调试为模型性能的进一步提升提供了新的可能性，从而使模型更好地适应任务需求。

本文系统探讨数据调试,首先构建其研究框架,将相关工作归纳为封闭式、浸入式和混合式数据调试三大类。其次,介绍每一大类的不同调试方法,分析其能力边界。随后,归纳现有数据调试方法常用的评价指标和数据集,设计并实现统一的实验评估。最后,分析数据调试面临的挑战,并讨论未来数据调试中可探索和深入的方向,旨在为相关研究提供有益参考,给该领域内研究人员一些启发。

作者贡献声明:李晨阳总结并撰写论文;马超红给出论文的撰写意见并修改论文;孟小峰指导论文的整体框架并给出修改意见。

参 考 文 献

[1] Nardin A. Artificial intelligence as a general purpose technology: An exploratory analysis of PCT patents[D]. Venezia, IT: Università Ca' Foscari Venezia, 2021: 1−113

[2] Voulodimos A, Doulamis N, Doulamis A, et al. Deep learning for computer vision: A brief review[J]. Computational Intelligence and Neuroscience, 2018, 2018: 1−13

[3] Chowdhary K R. Fundamentals of Artificial Intelligence[M]. Berlin: Springer, 2020

[4] Webb S. Deep learning for biology[J]. Nature, 2018, 554(7693): 555−557

[5] Ozbayoglu A M, Gudelek M U, Sezer O B. Deep learning for financial applications: A survey[J]. Applied Soft Computing, 2020, 93: 106384

[6] Taye M M. Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions[J]. Computers, 2023, 12(5): 91

[7] Jarrahi M H, Memariani A, Guha S. The principles of data-centric AI (DCAI)[J]. arXiv preprint, arXiv: 2211.14611, 2022

[8] Zha Daochen, Bhat Z P, Lai K H, et al. Data-centric artificial intelligence: A survey[J]. ACM Computing Surveys, 2025, 57(5): 1−42

[9] Mazumder M, Banbury C, Yao Xiaozhe, et al. Dataperf: Benchmarks for data-centric AI development[C]//Proc of the 36th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2023: 5320−5347

[10] Carbin M. Overparameterization: A connection between software 1.0 and software 2.0[C]//Proc of the 3rd Summit on Advances in Programming Languages. New York: ACM, 2019: 1: 1−1: 13

[11] Steve B. Artificial intelligence/ machine learning–explained[R/OL]. [2025-07-17]. https://gordianknot.stanford.edu

[12] Renggli C, Rimanic L, Gürel N M, et al. A data quality-driven view of MLOps[J]. IEEE Data Engineer Bulletin, 2021, 44(1): 11−23

[13] Galhotra S, Fariha A, Lourenço R, et al. Dataprism: Exposing disconnect between data and systems[C]//Proc of the 2022 Int Conf on Management of Data. New York: ACM, 2022, 217−231

[14] Rawal K, Kamar E, Lakkaraju H. Can I still trust you?: Understanding the impact of distribution shifts on algorithmic recourses[J]. arXiv preprint, arXiv: 2012.11788, 2020

[15] The Christian Science Monitor. Is Amazon same-day delivery service racist?[EB/OL].[2025-07-17].https://www.csmonitor.com/Business/2016/0423/Is-Amazon-same-day-delivery-service-racist

[16] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84−90

[17] Deng Jia, Dong Wei, Socher R, et al. ImageNet: A large-scale hierarchical image database[C]//Proc of the 2009 Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2009: 248−255

[18] Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold[J]. Nature, 2021, 596(7873): 583−589

[19] Mirdita M, Von Den Driesch L, Galiez C, et al. Uniclust databases of clustered and deeply annotated protein sequences and alignments[J]. Nucleic Acids Research, 2017, 45(1): 170−176

[20] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[C]//Proc of the 33rd Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2020, 33: 1877−1901

[21] Devlin J, Chang Mingwei, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[C]//Proc of the 2019 Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2019: 4171−4186

[22] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[EB/OL]. [2025-07-17]. https://api.semanticscholar.org/CorpusID:49313245

[23] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[EB/OL]. [2025-07-17]. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[24] Cai Li, Wang Shuting, Liu Junhui, et al. Survey of data annotation[J]. Journal of Software, 2020, 31(2): 302−320 (in Chinese)
(蔡莉, 王淑婷, 刘俊晖, 等. 数据标注研究综述[J]. 软件学报, 2020, 31(2): 302−320)

[25] Guo Hui, Wang Boyu, Yi G. Label correction of crowdsourced noisy annotations with an instance-dependent noise transition model[C]//Proc of the 36th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2023: 347−386

[26] Schoepf S, Foster J, Brintrup A. Parameter-tuning-free data entry error unlearning with adaptive selective synaptic dampening[J]. arXiv preprint, arXiv: 2402.10098, 2024

[27] Sambasivan N, Kapania S, Highfill H, et al. "Everyone wants to do the model work, not the data work": Data cascades in high-stakes AI[C]//Proc of the 2021 Human Factors in Computing Systems. New York: ACM, 2021: 1−15

[28] Fariha A, Tiwari A, Radhakrishna A, et al. Conformance constraint discovery: Measuring trust in data-driven systems[C]//Proc of the 2021 Int Conf on Management of Data. New York: ACM, 2021:

499−512

[29] Ng A, Laird D, He L. Data-centric AI competition[EB/OL]. [2025-07-17]. https://https-deeplearning-ai.github.io/data-centric-comp/

[30] Zha Daochen, Lai K H, Yang Fan, et al. Data-centric AI: Techniques and future perspectives[C]//Proc of the 29th ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2023: 5839−5840

[31] Zha Daochen, Bhat Z P, Lai K H, et al. Data-centric AI: Perspectives and challenges[C]//Proc of the 2023 SIAM Int Conf on Data Mining. Philadelphia, PA: SIAM, 2023: 945−948

[32] Aroyo L, Lease M, Paritosh P, et al. Data excellence for AI: Why should you care?[J]. Interactions, 2022, 29(2): 66−69

[33] Ré C. Is data management the beating heart of AI systems?[C]//Proc of the 2022 Int Conf on Management of Data. New York: ACM, 2022: 3−3

[34] Armbrust M, Ghodsi A, Xin R, et al. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics[C]//Proc of the 11th Conf on Innovative Data Systems Research. New York: ACM, 2021: 28

[35] Barclay T, Gray J, Slutz D. Microsoft TerraServer: A spatial data warehouse[C]//Proc of the 2000 Int Conf on Management of Data. New York: ACM, 2000: 307−318

[36] Thusoo A, Shao Zheng, Anthony S, et al. Data warehousing and analytics infrastructure at Facebook[C]//Proc of the 2010 Int Conf on Management of Data. New York: ACM, 2010: 1013−1020

[37] Mathew S, Varia J. Overview of Amazon web services[J]. Amazon Whitepapers, 2014, 105(1): 22

[38] Sak H, Senior A W, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[C]// Proc of the 15th Annual Conf of the Int Speech Communication Association. Vandoeuvre-lès-Nancy, FR: ISCA, 2014: 338−342

[39] Song Shaoxu, Zhang Aoqian, Wang Jianmin, et al. SCREEN: Stream data cleaning under speed constraints[C]//Proc of the 2015 Int Conf on Management of Data. New York: ACM, 2015: 827–841

[40] Ehrlinger L, Wöß W. A survey of data quality measurement and monitoring tools[J]. Frontiers in Big Data, 2022, 5: 850611

[41] Buckley T, Ghosh B, Pakrashi V. A feature extraction & selection benchmark for structural health monitoring[J]. Structural Health Monitoring, 2023, 22(3): 2082−2127

[42] Rekatsinas T, Chu Xu, Ilyas I F, et al. Holoclean: Holistic data repairs with probabilistic inference[J]. Proceedings of the Very Large Data Base Endowment, 2017, 10(11): 1190−1201

[43] Ge Congcong, Gao Yunjun, Miao Xiaoye, et al. A hybrid data cleaning framework using Markov logic networks[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(5): 2048−2062

[44] Chu Xu, Ilyas I F, Krishnan S, et al. Data cleaning: Overview and emerging challenges[C]//Proc of the 2016 Int Conf on Management of Data. New York: ACM, 2016: 2201−2206

[45] Luo Yuyu, Chai Chengliang, Qin Xuedi, et al. VisClean: Interactive cleaning for progress visualization[J]. Proceedings of the Very Large Data Base Endowment, 2020, 13(12): 2821−2824

[46] Guo Zizheng, Chen Pengyu, Fu Yanzhang, et al. Data debugging is NP-hard for classifiers trained with SGD[J]. arXiv preprint, arXiv: 2408.01365, 2024

[47] Ahmad S, Lavin A, Purdy S, et al. Unsupervised real-time anomaly detection for streaming data[J]. Neurocomputing, 2017, 262: 134−147

[48] Kurakin A, Goodfellow I J, Bengio S. Adversarial examples in the physical world[C]//Proc of the 5th Workshop at the Int Conf on Learning Representations. London, UK: Chapman & Hall, 2018: 99–112

[49] Pradhan R, Zhu Jiongli, Glavic B, et al. Interpretable data-based explanations for fairness debugging[C]//Proc of the 2022 Int Conf on Management of Data. New York: ACM, 2022: 247−261

[50] Liu Yejia, Wu Weiyuan, Flokas L, et al. Enabling SQL-based training data debugging for federated learning[J]. Proceedings of the Very Large Data Base Endowment, 2021, 15(3): 388−400

[51] Wu Weiyuan, Flokas L, Wu E, et al. Complaint-driven training data debugging for query 2.0[C]//Proc of the 2020 Int Conf on Management of Data. New York: ACM, 2020: 1317−1334

[52] Xu Yijiang, Gao Qing, Chen Liguo, et al. Research progress on directed grey-box fuzzing[J]. Chinese Journal of Computers, 2025, 48(5): 1244−1272
(徐邑江, 高庆, 陈立果, 等. 定向灰盒模糊测试技术研究进展[J]. 计算机学报, 2025, 48(5): 1244−1272)

[53] Li Peng, Rao Xi, Blase J, et al. CleanML: A study for evaluating the impact of data cleaning on ML classification tasks[C]//Proc of the 37th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2021: 13−24

[54] Muşlu K, Brun Y, Meliou A. Data debugging with continuous testing[C]//Proc of the Joint Meeting on Foundations of Software Engineering. New York: ACM, 2013: 631−634

[55] Rukat T, Lange D, Schelter S, et al. Towards automated ML model monitoring: Measure, improve and quantify data quality[C/OL]// [2025-07-17]. https://www.amazon.science/publications/towards-automated-data-quality-management-for-machine-learning

[56] Abdelaal M, Hammacher C, Schoening H. REIN: A comprehensive benchmark framework for data cleaning methods in ML pipelines[C]//Proc of the 26th Int Conf on Extending Database Technology. Berlin: Springer, 2023: 499−511

[57] Cadamuro G, Gilad-Bachrach R, Zhu Xiaojin. Debugging machine learning models[C]//Proc of the 2016 Workshop on Automatic Machine Learning. New York: ACM, 2016: 103

[58] Aamir A, Tamosiunaite M, Wrgtter F. Interpreting the decisions of CNNs via influence functions[J]. Frontiers in Computational Neuroscience, 2023, 17: 1172883

[59] Feng Lichao, Wang Xingya, Zhang Shiyu, et al. DeepFeature: Guiding adversarial testing for deep neural network systems using robust features[J]. The Journal of Systems & Software, 2025, 219: 112201

[60] Lee Y, Kwon O J, Lee H, et al. Augment & valuate: A data enhancement pipeline for data-centric ai[J]. arXiv preprint, arXiv: 2112.03837, 2021

[61] Hataya R, Zdenek J, Yoshizoe K, et al. Faster autoaugment: Learning augmentation strategies using backpropagation[C]//Proc of the 16th

European Conf on Computer Vision. Berlin: Springer, 2020: 1−16

[62] Tao Guanhong, Ma Shiqing, Liu Yinqi, et al. Trader: Trace divergence analysis and embedding regulation for debugging recurrent neural networks[C]//Proc of the 42nd ACM/IEEE Int Conf on Software Engineering. New York: ACM, 2020: 986−998

[63] Kirk M. Thoughtful Machine Learning: A Test-driven Approach[M]. Sebastopol, CA: O'Reilly Media, 2014

[64] Hellerstein J M. Quantitative Data Cleaning for Large Databases[M]. Berkeley, CA: University of California, Berkeley, 2008

[65] Kontokostas D, Westphal P, Auer S, et al. Databugger: A test-driven framework for debugging the Web of data[C]//Proc of the 23rd Int Conf on World Wide Web. New York: ACM, 2014: 115−118

[66] Khalek S A, Khurshid S. Systematic testing of database engines using a relational constraint solver[C]//Proc of the 4th IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2011: 50−59

[67] Pan Kai, Wu Xintao, Xie Tao. Database state generation via dynamic symbolic execution for coverage criteria[C]//Proc of the 4th Int Workshop on Testing Database Systems. New York: ACM, 2011: 1−6

[68] Schelter S, Lange D, Schmidt P, et al. Automating large-scale data quality verification[J]. Proceedings of the Very Large Data Base Endowment, 2018, 11(12): 1781−1794

[69] Barowy D W, Gochev D, Berger E D. Checkcell: Data debugging for spreadsheets[C]//Proc of the 2014 ACM Int Conf on Object Oriented Programming Systems Languages & Applications. New York: ACM, 2014: 507−523

[70] Do H, Rothermel G. Using sensitivity analysis to create simplified economic models for regression testing[C]//Proc of the 2008 Int Symp on Software Testing and Analysis. New York: ACM, 2008: 51−62

[71] Mao Chengying, Yu Xinxin. Test data generation for software testing based on quantum-inspired genetic algorithm[J]. International Journal of Computational Intelligence and Applications, 2013, 12(1): 1350004

[72] Biessmann F, Rukat T, Schmidt P, et al. DataWig: Missing value imputation for tables[J]. Journal of Machine Learning Research, 2019, 20(175): 1−6

[73] Biessmann F, Salinas D, Schelter S, et al. Deep learning for missing value imputationin tables with non-numerical data[C]//Proc of the 27th ACM Int Conf on Information and Knowledge Management. New York: ACM, 2018: 2017−2025

[74] Shankar S, Parameswaran A. Towards observability for production machine learning pipelines[J]. arXiv preprint, arXiv: 2108.13557, 2021

[75] Gao Yunjun, Ge Congcong, Guo Yuxiang, et al. Survey on data integration technologies for relational data and knowledge graph[J]. Journal of Software, 2023, 34(5): 2365−2391 (in Chinese)
(高云君, 葛丛丛, 郭宇翔, 等. 面向关系型数据与知识图谱的数据集成技术综述[J]. 软件学报, 2023, 34(5): 2365−2391)

[76] Fang X S, Sheng Q Z, Wang Xianzhi, et al. SmartVote: A full-fledged graph-based model for multi-valued truth discovery[J]. World Wide Web, 2019, 22(4): 1855−1885

[77] Ma Ruxia, Meng Xiaofeng, Wang Lu, et al. MTruths: An approach of multiple truths finding from Web information[J]. Journal of Computer Research and Development, 2016, 52(12): 2858−2866 (in Chinese)
(马如霞, 孟小峰, 王璐, 等. MTruths: Web 信息多真值发现方法[J]. 计算机研究与发展, 2016, 52(12): 2858−2866)

[78] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks[J]. Communications of the ACM, 2020, 63(11): 139−144

[79] Cai Zhipeng, Xiong Zuobin, Xu Honghui, et al. Generative adversarial networks: A survey toward private and secure applications[J]. ACM Computing Surveys, 2022, 54(6): 1−38

[80] Van Dyk D A, Meng Xiaoli. The art of data augmentation[J]. Journal of Computational and Graphical Statistics, 2001, 10(1): 1−50

[81] Singla S, Chegini A M, Moayeri M, et al. Data-centric debugging: Mitigating model failures via targeted data collection[J]. arXiv preprint, arXiv: 2211.09859, 2022

[82] Motamedi M, Sakharnykh N, Kaldewey T. A data-centric approach for training deep neural networks with less data[J]. arXiv preprint, arXiv: 2110.03613, 2021

[83] Chen Muxi, Li Yu, Xu Qiang. HiBug: On human-interpretable model debug[C]//Proc of the 36th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2024: 4753−4766

[84] OpenAI. ChatGPT: Optimizing language models for dialogue[EB/OL]. [2025-07-17]. https://openai.com/index/chatgpt/

[85] Malon C, Li Kai, Kruus E. Fast Few-shot debugging for NLU test suites[C]//Proc of the 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures. Stroudsburg, PA: ACL, 2022: 79−86

[86] Dang C, Le D D, Le T. A curious case of searching for the correlation between training data and adversarial robustness of Transformer textual models[C]//Proc of the Findings of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2024: 13475−13491

[87] Wang Zhou, Bovik A C, Sheikh H R, et al. Image quality assessment: From error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600−612

[88] Wang Zhou, Simoncelli E P, Bovik A C. Multiscale structural similarity for image quality assessment[C]//Proc of the 37th Asilomar Conf on Signals, Systems & Computers. Piscataway, NJ: IEEE, 2003: 1398−1402

[89] Zhang Lin, Zhang Lei, Mou Xuanqin, et al. FSIM: A feature similarity index for image quality assessment[J]. IEEE Transactions on Image Processing, 2011, 20(8): 2378−2386

[90] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint, arXiv: 2010.11929, 2020

[91] Pham H, Dai Zihang, Ghiasi G, et al. Combined scaling for zero-shot transfer learning[J]. Neurocomputing, 2023, 555: 126658

[92] He Kaiming, Chen Xinlei, Xie Saining, et al. Masked autoencoders are scalable vision learners[C]//Proc of the 2022 Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2022: 16000−16009

[93] Yang Jiachen, Zhang Zhou, Gong Yicheng, et al. Do deep neural networks always perform better when eating more data?[J]. arXiv preprint, arXiv: 2205.15187, 2022

[94] Grafberger S, Groth P, Stoyanovich J, et al. Data distribution debugging in machine learning pipelines[J]. The VLDB Journal, 2022, 31(5): 1103−1126

[95] Grafberger S, Guha S, Stoyanovich J, et al. MLINSPECT: A data distribution debugger for machine learning pipelines[C]//Proc of the 2021 Int Conf on Management of Data. New York: ACM, 2021: 2736−2739

[96] Grafberger S, Stoyanovich J, Schelter S. Lightweight inspection of data preprocessing in native machine learning pipelines[C]//Proc of the 11th Conf on Innovative Data Systems Research. New York: ACM, 2021: 27

[97] Schelter S, Stoyanovich J. Taming technical bias in machine learning pipelines[J]. IEEE Data Engineering Bulletin, 2020, 43(4): 39−50

[98] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. Journal of Machine Learning Research, 2011, 12: 2825−2830

[99] McKinney W. Pandas: A foundational Python library for data analysis and statistics[J]. Python for High Performance and Scientific Computing, 2011, 14(9): 1−9

[100] Brachmann M, Bautista C, Castelo S, et al. Data debugging and exploration with vizier[C]//Proc of the 2019 Int Conf on Management of Data. New York: ACM, 2019: 1877−1880

[101] Luo Yuyu, Qin Xuedi, Xie Yupeng, et al. Intelligent data visualization analysis techniques: A survey[J]. Journal of Software, 2024, 35(1): 356−404 (in Chinese)
(骆昱宇, 秦雪迪, 谢宇鹏, 等. 智能数据可视分析技术综述[J]. 软件学报, 2024, 35(1): 356−404)

[102] Ding Xiaoou, Liu Yida, Wang Hongzhi, et al. Efficient relaxed functional dependency discovery with minimal set cover [C]//Proc of the 40th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2024: 3519−3531

[103] Jiang Sijia, Tan Zijing, Wang Jiawei, et al. Guided conditional functional dependency discovery[J]. Information Systems, 2023, 114: 102158

[104] Golab L, Karloff H, Korn F, et al. Sequential dependencies[J]. Proceedings of the Very Large Data Base Endowment, 2009, 2(1): 574−585

[105] Parisi F, Grant J. On measuring inconsistency in definite and indefinite databases with denial constraints[J]. Artificial Intelligence, 2023, 318: 103884

[106] Pena E H M, Porto F, Naumann F. Fast algorithms for denial constraint discovery[J]. Proceedings of the Very Large Data Base Endowment, 2022, 16(4): 684−696

[107] Ding Xiaoou, Li Genglong, Wang Hongzhi, et al. Time series data cleaning under expressive constraints on both rows and columns[C]//Proc of the 40th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2024: 5689−5698

[108] Saria S, Subbaswamy A. Tutorial: Safe and reliable machine learning[J]. arXiv preprint, arXiv: 1904.07204, 2019

[109] Varshney K R. Trustworthy machine learning and artificial intelligence[J]. XRDS: Crossroads, The ACM Magazine for Students, 2019, 25(3): 26−29

[110] Tiwari A, Dutertre B, Jovanović D, et al. Safety envelope for security[C]//Proc of the 3rd Int Conf on High Confidence Networked Systems. New York: ACM, 2014: 85−94

[111] Varshney K R, Alemzadeh H. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products[J]. Big Data, 2017, 5(3): 246−255

[112] Wang Xiaolan, Dong X L, Meliou A. Data X-Ray: A diagnostic tool for data errors[C]//Proc of the 2015 Int Conf on Management of Data. New York: ACM, 2015: 1231−1245

[113] Bala A, Chana I. Intelligent failure prediction models for scientific workflows[J]. Expert System Applications, 2015, 42(3): 980−989

[114] Johnson B, Brun Y, Meliou A. Causal testing: Understanding defects' root causes[C]//Proc of the 42nd ACM/IEEE Int Conf on Software Engineering. New York: ACM/ Piscataway, NJ: IEEE, 2020: 87−99

[115] Lourenço R, Freire J, Shasha D. BugDoc: Algorithms to debug computational processes[C]//Proc of the 2020 Int Conf on Management of Data. New York: ACM, 2020: 463−478

[116] Lourenco R, Freire J, Simon E, et al. BugDoc: Iterative debugging and explanation of pipeline[J]. The VLDB Journal, 2023, 32(1): 75−101

[117] Loureno R, Freire J, Shasha D. Debugging machine learning pipelines[C]//Proc of the 3rd Int Workshop on Data Management for End-to-end Machine Learning. New York: ACM, 2020: 1−10

[118] Dolatnia N, Fern A, Fern X. Bayesian optimization with resource constraints and production[C]//Proc of the 26th Int Conf on Automated Planning and Scheduling. Palo Alto, CA: AAAI, 2016: 115−123

[119] Snoek J, Rippel O, Swersky K, et al. Scalable Bayesian optimization using deep neural networks[C]//Proc of the 32nd Int Conf on Machine Learning. New York: ACM, 2015: 2171−2180

[120] Vrugt J A, Robinson B A, Hyman J M. Self-adaptive multimethod search for global optimization in real-parameter spaces[J]. IEEE Transactions on Evolutionary Computation, 2008, 13(2): 243−259

[121] Koh P W, Liang P. Understanding black-box predictions via influence functions[C]//Proc of the 34th Int Conf on Machine Learning. New York: ACM, 2017: 1885−1894

[122] Hara S, Nitanda A, Maehara T. Data cleansing for models trained with SGD[C]//Proc of the Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2019: 4215−4224

[123] Xie Xiaofei, Guo Wenbo, Ma Lei, et al. RNNrepair: Automatic RNN repair via model-based analysis[C]//Proc of the 38th Int Conf on Machine Learning. New York: ACM, 2021: 11383−11392

[124] Wang F, Adebayo J, Tan S, et al. Error discovery by clustering influence embeddings [C]//Proc of the 36th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2023, 36: 41765−41777

[125] Deng Yuhao, Chai Chengliang, Cao Lei, et al. MisDetect: Iterative mislabel detection using early loss[J]. Proceedings of the Very Large Data Base Endowment, 2024, 17(6): 1159−1172

[126] Chai Chengliang, Wang Jiayi, Tang Nan, et al. Efficient coreset selection with cluster-based methods[C]//Proc of the 29th ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2023: 167−178

[127] Khanna R, Kim B, Ghosh J, et al. Interpreting black box predictions using fisher kernels[C]//Proc of the 22nd Int Conf on Artificial Intelligence and Statistics. Cambridge, MA: MIT, 2019: 3382−3390

[128] Koh P W W, Ang K S, Teo H, et al. On the accuracy of influence functions for measuring group effects[C]//Proc of the 32nd Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2019: 5255−5265

[129] Bae J, Ng N, Lo A, et al. If influence functions are the answer, then what is the question?[C]//Proc of the 35th Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT, 2022: 17953−17967

[130] Zhang Xuezhou, Zhu Xiaojin, Wright S J. Training set debugging using trusted items[C]//Proc of the 32nd AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2018: 4482−4489

[131] Do L N Q, Kruger S, Hill P, et al. Debugging static analysis[J]. IEEE Transactions on Software Engineering. 2018, 46(7): 697−709

[132] Ma Shiqing, Liu Yingqi, Lee W C, et al. MODE: Automated neural network model debugging via state differential analysis and input selection[C]//Proc of the 26th ACM Joint Meeting on European Software Engineering Conf and Symp on the Foundations of Software Engineering. New York: ACM, 2018: 175−186

[133] Lapuschkin S, Wäldchen S, Binder A, et al. Unmasking clever hans predictors and assessing what machines really learn[J]. Nature Communications, 2019, 10(1): 1096

[134] Zeller A. Yesterday, my program worked. Today, it does not. why?[J]. ACM SIGSOFT Software Engineering Notes, 1999, 24(6): 253−267

[135] Jones J A, Harrold M J, Stasko J. Visualization of test information to assist fault localization[C]//Proc of the 24th Int Conf on Software Engineering. New York: ACM, 2002: 467−477

[136] Guo Hongjing, Tao Chuanqo, Huang Zhiqiu. Multi-objective white-box test input selection for deep neural network model enhancement[C]//Proc of the 34th IEEE Int Symp on Software Reliability Engineering. Piscataway, NJ: IEEE, 2023: 521−532

[137] Boci I, Bultan T. Finding access control bugs in Web applications with CanCheck[C]//Proc of the 31st IEEE/ACM Int Conf on Automated Software Engineering. Piscataway, NJ: IEEE, 2016: 155−166

[138] Gao Zheng, Bird C, Barr E T. To type or not to type: Quantifying detectable bugs in JavaScript[C]//Proc of the 39th IEEE/ACM Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2017: 758−769

[139] Du Xiaoning, Xie Xiaofei, Li Yi, et al. Deepstellar: Model-based quantitative analysis of stateful deep learning systems[C]//Proc of the 27th ACM Joint Meeting on European Software Engineering Conf and Symp on the Foundations of Software Engineering. New York: ACM, 2019: 477−487

[140] Jolliffe I. International Encyclopedia of Statistical Science[M]. Berlin: Springer, 2011: 1094–1096

[141] Flokas L, Wu Weiyuan, Wang Jiannan, et al. How I stopped worrying about training data bugs and started complaining[C]//Proc of the 6th Workshop on Data Management for End-To-End Machine Learning. New York: ACM, 2022: 1−5

[142] Flokas L, Wu Weiyuan, Liu Yejia, et al. Complaint-driven training data debugging at interactive speeds[C]//Proc of the 2022 Int Conf on Management of Data. New York: ACM, 2022: 369−383

[143] Rezig E K, Cao L, Simonini G, et al. Dagger: A data (not code) debugger[C]//Proc of the 10th Conf on Innovative Data Systems Research. New York: ACM, 2020: 20

[144] Rezig E K, Brahmaroutu A, Tatbul N, et al. Debugging large-scale data science pipelines using dagger[J]. Proceedings of the Very Large Data Base Endowment, 2020, 13(12): 2993−2996

[145] Rezig E K, Cao Lei, Stonebraker M, et al. Data Civilizer 2.0: A holistic framework for data preparation and analytics[J]. Proceedings of the Very Large Data Base Endowment, 2019, 12(12): 1954−1957

[146] Chai T, Draxler R R. Root mean square error (RMSE) or mean absolute error (MAE)[J]. Geoscientific Model Development Discussions, 2014, 7(1): 1525−1534

[147] Varshini A G P, Kumari K A, Janani D, et al. Comparative analysis of machine learning and deep learning algorithms for software effort estimation[J]. Journal of Physics: Conference Series, 2021, 1767(1): 012019

[148] Reddit. Reddit-comments dataset[EB/OL]. [2025-07-17]. https://www.kaggle.com/datasets/kaggle/reddit-comments-may-2015

[149] Wikipedia. Dbpedia dataset[EB/OL]. [2025-07-17]. http://wiki.dbpedia.org/downloads-2016-10

[150] US Census Bureau. Adult dataset[EB/OL]. [2025-07-17]. https://archive.ics.uci.edu/dataset/2/adult

[151] Li Feifei. ImageNet dataset[EB/OL]. [2025-07-17]. https://image-net.org

[152] Kaggle. Roman-MNIST dataset[EB/OL]. [2025-07-17]. https://worksheets.codalab.org/worksheets/0x7a8721f11e61436e93ac8f76da83f0e6

[153] National Institute of Standards and Technology. MNIST dataset [EB/OL]. [2025-07-17]. https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[154] Radiological Society of North America. RSNA dataset[EB/OL]. [2025-07-17]. https://huggingface.co/fabriceyhc/bert-base-uncased-ag_news

[155] Kaggle. Toxic dataset[EB/OL]. [2025-07-17]. https://www.kaggle.com/datasets/julian3833/jigsaw-toxic-comment-classification-challenge

[156] Centers for Medicare & Medicaid Services. Hospital dataset[EB/OL]. [2025-07-17]. https://data.cms.gov/provider-data/search?page-size=50&theme=Hospitals

[157] Kaggle. Airlines dataset[EB/OL]. [2025-07-17]. https://www.kaggle.com/datasets/iamsouravbanerjee/airline-dataset

[158] Kaggle. Flight delay dataset[EB/OL]. [2025-07-17]. http://oaklea-fblog.blogspot.com/2012/05/accessing-us-air-carrier-flight-delay.

html

[159] Canadian Institute for Advanced Research. Cifar10 dataset[EB/OL]. [2025-07-17]. https://www.cs.toronto.edu/~kriz/cifar.html

[160] Duan Shaoming, Liu Chuanyi, Han Peiyi, et al. Fed-DNN-Debugger: Automatically debugging deep neural network models in federated learning[J]. Security and Communication Networks, 2023, 2023(1): 5968168

[161] Li Anran, Zhang Lan, Wang Junhao, et al. Efficient federated-learning model debugging[C]//Proc of the 37th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2021: 372−383



Li Chenyang, born in 1997. PhD candidate. Her main research interests include data debugging, data management, and machine learning.

李晨阳, 1997 年生。博士研究生。主要研究方向为数据调试、数据管理、机器学习。



Ma Chaohong, born in 1994. PhD. Her main research interests include machine learning for database systems, learned indexes, and data management.

马超红, 1994 年生。博士。主要研究方向为机器学习赋能的数据库系统、学习化索引、数据管理。



Meng Xiaofeng, born in 1964. PhD, professor. His main research interests include database system, data management, privacy protection, and interdisciplinary research like social computing.

孟小峰, 1964 年生。博士，教授。主要研究方向为数据库系统、数据管理、隐私保护、社会计算等交叉性研究。