软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(7):2831−2848 [doi: 10.13328/j.cnki.jos.007585] [CSTR: 32375.14.jos.007585]

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

PCLog: 近端策略优化与行为克隆自适应日志异常检测*

周俊伟, 胡 淼, 王春龙, 杜亚娟, 谈 诚

(武汉理工大学 计算机与人工智能学院, 湖北 武汉 430070)

通信作者: 周俊伟, E-mail: junweizhou@msn.com

摘 要: 日志数据记录了系统的运行状态、用户行为及错误信息. 基于日志的异常检测可快速识别潜在的安全风险或性能瓶颈, 提升运维效率, 助力故障诊断. 然而, 现有的日志异常检测方法仍面临诸多挑战, 如无法有效适应系统升级引起的日志模式变化, 缺乏高效反馈机制导致难以持续保持检测性能等. 为此, 提出一种日志异常检测框架PCLog, 采用强化学习方法中的近端策略优化 (proximal policy optimization, PPO) 算法进行模型训练. 该方案将检测模型视为智能体, 日志对应的语义向量视为状态, 事件视为动作, 通过最大化正常序列的累计奖励来学习系统的正常行为模式从而实现异常检测. 此外, 当检测性能下降时, PCLog 可通过收集错误预测的样本作为专家示范数据, 并结合模仿学习中的行为克隆方法, 最大化专家数据的对数似然, 从而使模型更有效地逼近专家行为, 实现模型的自适应修正, 有效减少误报率, 提升系统长期运行的可靠性. 在 HDFS、BGL 与 OpenStack 这 3 大公开日志数据集上的实验结果表明, PCLog 相较于现有方法表现更优, 具备较强的动态日志模式适应能力.

关键词: 日志异常检测; 近端策略优化; 行为克隆

中图法分类号: TP311

中文引用格式: 周俊伟, 胡淼, 王春龙, 杜亚娟, 谈诚. PCLog: 近端策略优化与行为克隆自适应日志异常检测. 软件学报, 2026, 37(7): 2831–2848. http://www.jos.org.cn/1000-9825/7585.htm

英文引用格式: Zhou JW, Hu M, Wang CL, Du YJ, Tan C. PCLog: Adaptive Log Anomaly Detection Based on Proximal Policy Optimization and Behavior Cloning. Ruan Jian Xue Bao/Journal of Software, 2026, 37(7): 2831–2848 (in Chinese). http://www.jos.org.cn/1000-9825/7585.htm

PCLog: Adaptive Log Anomaly Detection Based on Proximal Policy Optimization and Behavior Cloning

ZHOU Jun-Wei, HU Miao, WANG Chun-Long, DU Ya-Juan, TAN Cheng

(School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China)

Abstract: Log data record system operating status, user behavior, and error information. Log-based anomaly detection enables the rapid identification of potential security risks or performance bottlenecks, thereby enhancing operational efficiency and facilitating fault diagnosis. However, existing log anomaly detection methods still face several challenges, including the inability to effectively adapt to log pattern changes caused by system updates and the lack of efficient feedback mechanisms to consistently maintain detection performance. To address these issues, this study proposes a log anomaly detection framework, PCLog, which adopts the proximal policy optimization (PPO) algorithm from reinforcement learning for model training. In the proposed framework, the detection model is formulated as an agent, semantic vectors of logs are regarded as states, and events are treated as actions. By maximizing the cumulative rewards of normal sequences, normal system behavior patterns are learned to enable anomaly detection. In addition, when detection performance decreases, PCLog collects mispredicted samples as expert demonstration data and integrates behavior cloning from imitation learning to maximize the

* 基金项目: 湖北省重点研发基金 (2025BEB012)
本文由“智能化基础软件可信构造和供应链安全”专题特约编辑王林章教授、司徒凌云研究员、钟浩研究员、 向剑文教授、张路教授推荐.
收稿时间: 2025-09-07; 修改时间: 2025-10-20; 采用时间: 2025-12-08; jos 在线出版时间: 2025-12-26
CNKI 网络首发时间: 2026-06-02

log-likelihood of expert data. This mechanism enables the model to more effectively approximate expert behavior, achieve adaptive self-correction, reduce false positives, and enhance long-term reliability. Experimental results on three public log datasets, HDFS, BGL, and OpenStack, show that PCLog outperforms existing methods and exhibits high adaptability to dynamic log patterns.

Key words: log anomaly detection; proximal policy optimization (PPO); behavior cloning

1 研究背景与动机

1.1 研究背景

随着系统规模的不断扩大, 现代计算系统正朝着两种方向演进: 一方面通过横向扩展构建包含成千上万台商用设备的分布式系统, 另一方面通过纵向扩展构建具有数千处理器的高性能计算系统[1]. 日志作为捕捉系统状态、事件与行为的重要数据来源, 蕴含着大量与系统健康状况、潜在故障及安全威胁相关的关键信息. 基于日志的异常检测不仅可以保障业务连续性、降低经济损失, 还可显著提升运维效率、降低运维成本. 因此, 日志异常检测技术在学术研究和工程实践中均具有重要的研究价值和应用前景. 近年来, 日志异常检测与机器学习、文本挖掘和时间序列分析等前沿技术深度融合, 促进了智能运维 (AIOps) 技术的发展与落地, 成为推动系统可观测性和智能诊断能力提升的重要技术手段.

日志异常检测通常包括 4 个主要阶段: 日志采集、日志解析、日志分组与特征提取、异常检测. 首先, 使用日志解析工具[2−7]将原始的非结构化文本日志转化为结构化形式, 从中提取出日志事件及其参数. 随后, 依据唯一标识符 (如 BlockID 或 InstanceID) 对日志进行分组, 若无标识符, 则使用滑动窗口方式切分日志序列. 之后, 结合自然语言处理技术[8−10], 将日志事件嵌入为语义向量; 如采用预训练模型 LogBERT[11]、LogFit[12]等可进一步增强语义表达能力, 但也会带来较大的计算开销. 最后, 提取后的特征被送入检测模型, 实现实时异常识别.

早期的日志异常检测方法主要依赖人工规则和模式匹配技术, 如关键字匹配与正则表达式等, 尽管实现简单、执行高效, 但难以覆盖复杂多样的异常模式. 当前方法可大致分为两类: 基于机器学习的方法与基于深度学习的方法. 机器学习方法又可细分为有监督与无监督方法. 尽管有监督方法[13−16]在特定场景下具有较好效果, 但其对大量标注数据的依赖限制了实际应用的泛化能力. 相较之下, 无监督方法[17−25]不依赖标签数据, 可直接从大规模日志中挖掘潜在异常模式, 具有更强的通用性与可扩展性. 然而, 系统日志数据的高维特性使得无监督学习面临“维度灾难”, 距离度量的有效性降低, 从而影响聚类与检测性能.

深度学习方法凭借其强大的表征能力, 在处理高维系统日志数据方面展现出明显优势. 序列模型如门控循环单元 (gated recurrent unit, GRU)、长短期记忆网络 (long short-term memory, LSTM) 与 Transformer 模型可有效捕捉日志序列中的长程依赖与复杂行为模式[26−35]. 此外, 图神经网络 (graph neural network, GNN) 被用于建模日志事件间的拓扑结构[36−40], 生成对抗网络 (generative adversarial network, GAN) 则被用于构建丰富的日志特征表示[41]. 这些深度学习方法的应用显著提升了异常检测的准确性. 近年来, 大语言模型 (large language model, LLM)[42−45]也逐渐应用于日志分析领域, 凭借其强大的上下文理解能力, 可更好地挖掘潜在异常. 结合少样本学习和提示调优技术, LLM 方法在减少标注数据依赖方面具有显著优势, 提升了日志分析系统在多场景下的灵活性与适应性.

1.2 研究动机

尽管现有深度学习与 LLM 方法在日志异常检测中取得了显著进展, 但其应用效果在动态、长期运行的系统环境中仍存在局限. 实际工程实践表明, 随着软件版本迭代、系统组件重构以及配置参数调整, 日志模式往往会持续演化, 导致模型面临“概念漂移 (concept drift)”问题. 此时, 基于静态训练的检测模型在新日志上的识别性能显著下降, 甚至出现误报率骤增、召回率骤降的现象. 例如, 在 OpenStack 数据集的实验中, 我们观察到: 当系统更新引入新的日志模板后, 原先基于 Transformer 结构的检测模型召回率下降超过 5%, 说明模型难以适应日志分布的动态变化.

另一方面, 尽管 LLM 具有强大的语义建模与上下文理解能力, 但其在日志异常检测中的应用仍面临 3 个方面的挑战.

1) 高资源成本: 模型参数规模庞大, 训练与推理均需要大量计算资源, 难以在高并发的生产环境中实时部署.

2) 领域适应性差: 预训练模型语料主要来源于通用文本, 无法充分捕获系统日志的结构化语义与事件逻辑关系, 导致领域迁移性能不稳定.

3) 缺乏持续自适应能力: 当系统环境变化或日志分布演化时, LLM 需要依赖人工微调或提示工程调整, 维护代价高且响应滞后.

针对上述问题, 本文提出一种融合强化学习与模仿学习的自适应日志异常检测框架——PCLog. 该方法通过近端策略优化 (proximal policy optimization, PPO) 实现检测策略的动态更新, 使模型能够在日志分布变化时自主调整行为决策; 当检测性能下降时, 系统自动收集错误预测样本作为专家示范, 引入行为克隆 (behavior cloning, BC) 机制进行策略微调, 从而在不依赖人工标注或重新训练的情况下快速恢复性能. 这种基于强化学习与模仿学习协同的机制, 能够有效缓解静态模型的性能漂移问题, 提升检测的持续稳定性与鲁棒性.

综上, 本文的主要贡献如下.

1) 提出一种基于近端策略优化 (PPO) 算法的强化学习日志异常检测方法 PCLog, 能够在无需大量历史日志的情况下学习系统的正常行为策略, 用于异常检测.

2) 融合模仿学习机制, 在检测性能下降时收集错误预测样本作为专家示范, 引入行为克隆算法对模型策略进行调整, 有效降低误报率, 提升模型鲁棒性.

3) 在 HDFS、BGL 与 OpenStack 这 3 个公开日志数据集上进行充分实验, PCLog 在各数据集上均取得 99% 以上的召回率, 展现出对动态日志模式的良好适应性.

本文第 2 节介绍相关工作. 第 3 节阐述 PCLog 的设计与实现细节. 第 4 节描述并讨论评估实验. 第 5 节对本文工作进行总结. PCLog 的源代码可在以下网址获取: https://github.com/whutwcl3093/PCLog.

2 相关工作

2.1 自适应日志异常检测方法

为了有效地应对日志模式的变化并实现自适应学习, 近年来学者们提出了多种具有代表性的方法. Wang 等人[46]提出了一种名为 LogOnline 的半监督异常检测方法, 该方法通过在线学习机制不断从新生成的日志序列中学习正常模式, 无需人工标注与干预, 较好地缓解了日志模式不稳定问题. Han 等人[47]提出了 ROEAD, 一种鲁棒的在线演化异常检测框架, 通过引入鲁棒特征提取机制以削弱噪声干扰, 并采用在线演化机制实现模型参数的动态更新. 理论分析表明该方法性能可逐步逼近最优. Yuan 等人[48]设计了 ADA, 一种基于 LSTM 的自适应深度日志异常检测模型, 可实时捕捉新的日志模式, 并结合自适应模型选择策略以提高资源利用率, 同时采用动态阈值算法, 根据近期检测结果优化阈值设置, 进一步提升检测性能. 然而, 这些基于深度学习的模型往往需要频繁重训练或微调, 而在原始超参数配置下, 这些模型在新日志数据上的检测性能仍存在不确定性.

2.2 基于强化学习与模仿学习的日志异常检测方法

近年来, 强化学习 (reinforcement learning, RL) 在日志异常检测领域逐渐显示出其潜力. Duan 等人[49]最早将强化学习引入该领域, 提出了基于 Q-learning 的 QLLog 方法. 该方法能够通过 Q-learning 算法自动学习日志行为模式, 引入反馈机制与异常严重度评级, 并可检测多种异常类型, 同时实现对异常事件的严重程度排序. 实验证明, 该方法在多个公共数据集上具有良好的检测精确率与适应性. Zhou 等人[50]在此基础上提出了 DRLLog, 一个基于深度强化学习的日志异常检测框架. DRLLog 以深度 Q 网络 (DQN) 为智能体, 在日志数据构建的环境中进行交互, 通过动作选择与反馈学习, 实现对日志序列中模式变化的持续适应, 克服了传统静态检测模型的局限性.

总体来看, 强化学习在日志异常检测中的应用具有显著优势. 一方面, 日志数据天然具有时序性, 强化学习可通过策略学习机制建模系统行为的执行路径, 从而提高对异常行为的识别能力[51]; 另一方面, 强化学习的动态学习能力使智能体能够通过与环境的持续交互, 适应日志模式的不断演化, 显著增强检测模型的泛化性与鲁棒性.

此外, 模仿学习 (imitation learning, IL) 作为另一种决策策略学习范式, 通过专家示范学习状态-动作映射, 不依

赖于外部定义的奖励信号. 常见模仿学习技术包括行为克隆与逆强化学习 (inverse reinforcement learning, IRL) 等. 在日志异常检测任务中, Wang 等人[52]提出了一种基于行为克隆的离线模仿学习方法, 通过提取行为最优性与序列关联性两类行为特征, 并构建基于 Transformer 的策略网络建模行为轨迹. 该方法结合 Q 函数与状态值函数的学习, 从正常序列中提取行为模式, 有效提升了对异常行为的识别能力. Oh 等人[53]提出了一种基于逆强化学习的序列异常检测框架, 旨在从观察到的动作序列与元数据中反推出智能体的潜在决策策略. 他们使用神经网络建模奖励函数, 并引入贝叶斯推断以增强异常评分的稳定性与可靠性.

值得注意的是, 融合模仿学习与强化学习的混合框架已在复杂环境下展现出较高的学习效率与泛化能力. 一种常见策略是先通过模仿学习预训练接近最优的策略, 再结合强化学习进行策略细化; 另一种策略则是在训练过程中联合利用专家示范与实时交互反馈, 实现端到端的策略优化. 此类混合方法特别适用于高风险或数据稀缺场景, 在奖励稀疏或难以定义时亦具备较大优势. 然而, 这两类方法仍存在局限性: 例如依赖人工标注反馈、难以应对未见过的日志模板、对异常数据的鲁棒性不足等.

3 方 法

PCLog 框架主要由 3 个部分组成: 数据预处理、模型训练和异常检测, 如图 1 所示. 首先, 将原始半结构化日志文本中的参数替换为对应的标签符号, 以提取标准化的日志事件. 随后, 采用句子嵌入技术将日志文本转换为高维稠密向量, 从而保留其语义信息.

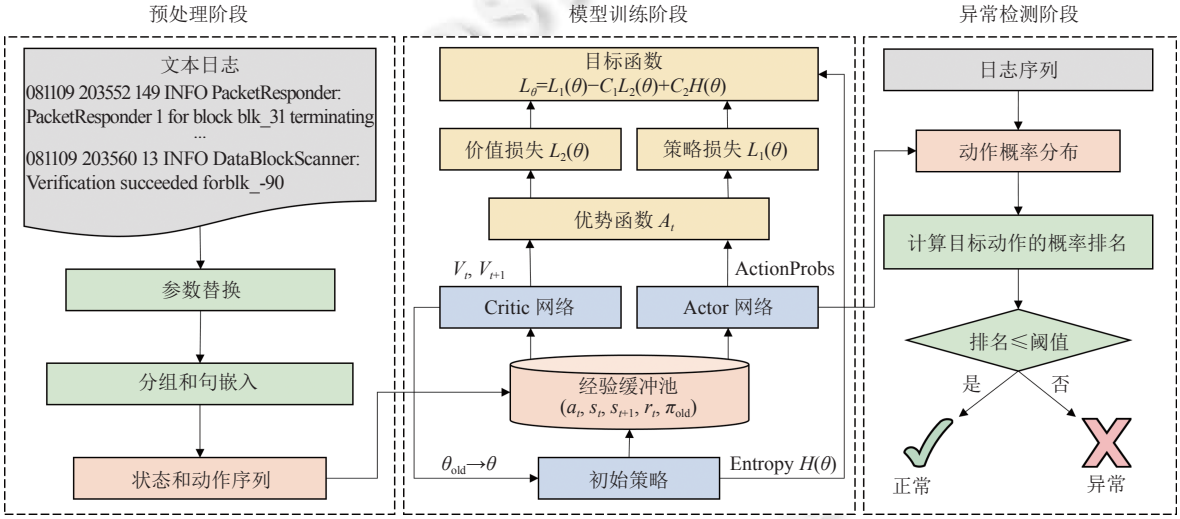


图 1 PCLog 框架

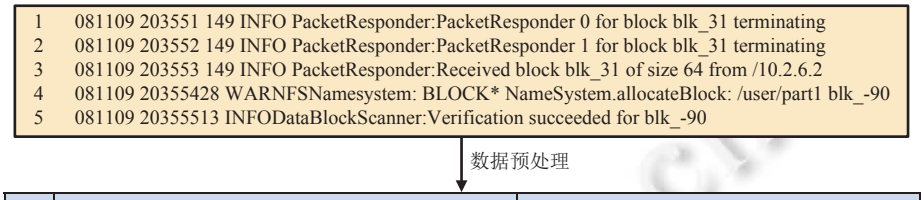
在模型训练阶段, PCLog 引入了两个结构相同但参数独立的神经网络: 策略网络 (Actor) 和价值网络 (Critic), 均由简单的线性层构成. Actor 负责输出可能下一个事件的概率分布, 用于在当前状态 s_t 下引导智能体选择动作 a_t; Critic 则对当前状态的价值 V_t 进行评估, 用以衡量其长期回报. 训练过程中, 从正常日志中获取语义向量和事件序列, 构成状态与动作对, 生成单步训练样本, 并存储在经验回放缓冲区中. 当缓冲区达到预设的批大小后, 启动模型训练. 其中, Actor 通过剪辑目标函数进行更新, Critic 则基于均方误差损失进行优化.

模型训练完成后, 可对日志序列中每个事件的概率分布进行评估, 并与实际事件进行比对, 以实现异常检测. 当检测效果下降时, 管理员可对错误预测提供反馈, 模型将这些反馈样本整合为专家轨迹, 并采用行为克隆方法对策略进行微调, 从而提升模型的适应能力和检测精度.

3.1 数据预处理

系统日志通常包含丰富的系统运行状态信息, 需首先进行规范化与结构化处理以满足后续模型输入要求. 预

处理流程如图 2 所示, 主要包括参数替换、日志事件提取与句向量生成 3 个步骤. 一条原始日志记录由日志头 (Header) 与日志体 (Body) 两部分组成, 其中日志头包含时间戳、进程编号、日志级别与系统模块等信息; 日志体则记录了系统在特定时刻的运行状态, 通常由开发人员通过打印语句输出, 包括固定格式的日志事件 (模板) 与变化部分的参数内容 (如文件路径、IP 地址、MAC 地址、主机名等). 为统一格式, 本文首先利用正则表达式识别日志体中的参数值, 并将其替换为相应的标签标记. 部分典型替换示例如表 1 所示.



	日志头				日志体	
行号	时间戳	进程号	警告级别	组件	模板 (日志事件)	参数列表
1	081109 203551	149	INFO	PacketResponder	PacketResponder number for block terminating	[0,blk_31]
2	081109 203552	149	WARN	PacketResponder	PacketResponder number for block terminating	[1,blk_31]
3	081109 203553	149	INFO	PacketResponder	Received block of size number from IP	[64,blk_31,/10.2.6.2]
4	081109 203554	28	INFO	FSName system	BLOCK* NameSystem allocateBlock path	[/user/part1,blk_-90]
5	081109 203555	13	INFO	DataBlockScanner	Verification succeeded for block	[blk_-90]

图 2 日志预处理

表 1 参数替换示例

正则表达式	参数实例	替换标签
\d{1, 3}\.\d{1, 3}\.\d{1, 3}\.\d{1, 3}	10.2.6.2	ip
0x[a-fA-F0-9]{8}	0x00544ea8	hex
R\d+-[-]*	R23-M1-N4-I1J8-U01	device
^(\w+)*\w+$	/user/part1	path
(\d+)	64	number

替换完成后, 日志体格式高度统一, 通过去重即可提取出标准化的日志事件. 由于日志事件的种类有限, 本文为每类事件分配唯一编号, 并将其作为动作集输入至强化学习模型中. 进一步地, 为保留日志的语义信息, 本文采用句向量嵌入方法将文本形式的日志事件映射为固定维度的高维稠密向量. 具体而言, 首先利用公开语料预训练词向量, 再通过平滑逆频率 (smooth inverse frequency, SIF) 方法生成日志句向量. 设一条日志消息 m 中包含$|m|$个词, 其句向量表示为:

$$V_m = \frac{1}{|m|}\sum \frac{a}{a+p(w)}w_i, V_m = V_m - uu^{\mathrm{t}}V_m \tag{1}$$

其中, a 为平滑常数, $p(w)$ 为词 w 的出现概率; 第 1 个等式得到的向量被拼接成矩阵 X, 而 u 是 X 的第 1 个奇异向量; 第 2 个等式从每个句子向量中移除了其在 u 上的投影. 此方法能够有效保留词语的关键信息, 同时具有良好的计算效率与鲁棒性.

最后, 依据日志中的唯一标识符 (如 BlockID、InstanceID) 对日志进行分组, 形成按时间排序的日志序列, 作为状态-动作序列输入模型. 在 HDFS 数据集中使用 BlockID 分组, 在 OpenStack 中使用 InstanceID 分组, 对于 BGL 数据集则采用滑动窗口方式切分. 每条日志序列均记录了系统事件的执行顺序与模式, 为后续建模提供关键上下文信息.

3.2 目标函数与模型训练

在 PCLog 中, 日志异常检测被建模为一个序列决策问题, 其状态转移过程满足马尔可夫性质, 即当前状态的转移仅依赖于前一状态, 与历史状态无关. 因此, 该问题可形式化为马尔可夫决策过程 (Markov decision process, MDP), 表示为五元组 $\langle S, A, T, R, \gamma\rangle$, 其中, S 为状态空间, 表示通过日志嵌入得到的向量; A 为动作空间; 对应离散的日志事件编号, $T(s, a, s')$ 为状态转移概率; $R(s, a)$ 为即时奖励; $\gamma\in[0, 1)$ 为折扣因子.

在日志异常检测任务中, 经过预处理的日志数据可以转换为适合强化学习的输入格式. 具体而言, 日志根据标识符进行分组, 形成多个日志序列, 这些序列可视为动作轨迹. 通过对原始半结构化文本日志进行特征提取获得的向量表示构成了状态集合, 而不同的日志事件则对应动作集合. 为有效处理序列边界, 每个序列以固定的零向量 v_{start} 作为初始状态开始, 并以固定事件 e_{end} 作为终止动作结束. 检测模型作为智能体, 通过与定制环境的交互来学习策略 π. 从形式上看, 策略 π 被定义为从状态空间 S 到动作空间 A 的映射函数, 即 π: $S\rightarrow A$. 环境交互与奖励定义的示意图见图 3.

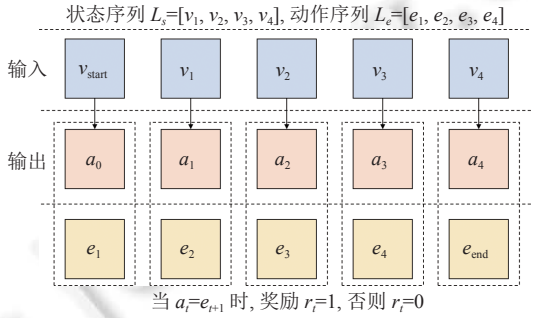


图 3 环境交互和奖励定义

假设日志序列对应状态序列 L_s 和动作序列 L_e. 作为智能体的检测模型接收当前状态向量 v_t, 并输出该状态下所有可能动作的概率分布. 随后从该分布中随机采样得到动作 a_t. 若所选动作与目标动作匹配, 则智能体获得奖励 1; 否则奖励为 0. 该模型的目标是通过与环境的持续交互生成轨迹, 并基于这些轨迹学习获得最优策略, 使得所有生成轨迹的期望累积奖励最大化.

本文提出的 PCLog 模型集成了强化学习中的近端策略优化算法 (PPO)[54]用于日志异常检测. PPO 算法的架构包含两个核心组件: 策略网络 (Actor) 和值函数网络 (Critic). 其中, 策略网络通过输出所有可能动作的概率分布来指导智能体决策, 而值函数网络则评估当前状态的价值, 为策略优化提供基准. 与 IRL 需要先学习奖励函数不同, PCLog 直接设定任务驱动的奖励函数, 结合反馈式行为克隆进行数据层修正, 两者在建模目标和优化方式上存在显著区别. 具体而言, 策略网络的更新通过优化带裁剪的目标函数实现, 值函数网络则通过最小化均方误差来逼近状态价值函数. 设 θ 表示策略网络的神经网络参数, ϕ 表示值函数网络的参数. 在训练过程中, 每个数据点被视为单步样本, 批次大小设置为 N. 近端策略优化 (PPO) 中执行器 (策略网络) 的优化目标定义为:

$$\max_{\pi_\theta} J^{\text{CLIP}}(\pi_\theta)=\mathbb{E}_t[\min(r_t(\theta)A_\phi^{\text{GAE}}(s_t,a_t), r_t^{\text{CLIP}}(\theta)A_\phi^{\text{GAE}}(s_t,a_t))] \tag{2}$$

其中, $r_t(\theta)$ 表示新旧策略之间的概率比:

$$r_t(\theta)=\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3}$$

为避免策略更新幅度过大导致训练不稳定, 我们在算法中引入了概率比的裁剪版本:

$$r_t^{\text{CLIP}}(\theta)=\text{CLIP}(r_t(\theta), 1-\varepsilon, 1+\varepsilon) \tag{4}$$

优势函数估计值 $A_\phi^{\text{GAE}}(s_t,a_t)$ 通过广义优势估计 (generalized advantage estimation, GAE) 计算得到, 其定义为:

$$A_\phi^{\text{GAE}}(s_t,a_t)=\sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l} \tag{5}$$

其中, δ_t 表示时序差分 (temporal difference, TD) 误差:

$$\delta_t = r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t) \tag{6}$$

该方法避免了显式估计动作价值函数 $Q_\pi(s_t, a_t)$, 从而有效降低了计算复杂度. 价值函数网络 (Critic) 的优化目标是最小化预测值与目标值之间的均方误差:

$$\min_{V_\phi} L(V_\phi) = \mathbb{E}_t\left[\left(V_\phi(s_\mathrm{t}) - V_t^{\mathrm{target}}\right)^2\right] \tag{7}$$

其中, 目标值的定义如下:

$$V_\mathrm{t}^{\mathrm{target}} = A_t^{\mathrm{GAE}} + V_{\phi_{\mathrm{old}}}(s_t) \tag{8}$$

为了在联合训练策略网络和价值网络的同时增强探索能力, PPO 算法引入了熵正则项, 其完整优化目标函数为:

$$\mathcal{L}^{\mathrm{PPO}}(\theta, \phi) = \mathbb{E}_t\left[J^{\mathrm{CLIP}}(\pi_\theta) - c_1 L(V_\phi) + c_2 \mathcal{H}(\pi_\theta(\cdot|s_t))\right] \tag{9}$$

其中, $\mathcal{H}(\pi_\theta(\cdot|s_t))$ 表示策略的熵, c_1 和 c_2 为调节系数, 通常设置为 $c_1 = 1.0$, $c_2 = 0.01$.

PCLog 模型采用经典的双网络架构设计, 包含策略网络与价值网络两个模块. 这两个网络采用完全相同的线性层结构, 但各自维护独立的参数体系. 具体而言: 策略网络通过 *Softmax* 函数输出动作空间的概率分布, 其功能是指导智能体基于当前日志状态特征 s_t 选择最优事件动作 a_t. 价值网络则输出标量估值 V_t, 用于量化评估当前状态的长期收益潜力.

训练过程中, 策略函数 π_θ 和价值函数 V_ϕ 分别由参数 θ 和 ϕ 进行初始化建模. 系统将正常日志中的语义向量和事件序列分别作为状态和动作输入, 其中每个序列以固定零向量 v_{start} 作为初始状态, 并以固定事件 e_{end} 作为终止动作. 日志状态被依次输入策略函数后, 系统根据输出的动作概率分布进行随机采样, 若所选动作与序列中下一事件匹配则给予 1 的奖励, 否则奖励为 0. 完成动作选择后系统将状态转移至下一日志状态, 直至序列结束. 每个长度为 L 的日志序列对应 $L+1$ 步的轨迹数据, 每一步生成形如 (s_t, s_{t+1}, r_t, a_t) 的训练样本, 其中 s_t 和 s_{t+1} 分别表示当前状态和下一状态的语义向量, r_t 为即时奖励值, a_t 表示策略选择的动作. 每个序列产生的 $L+1$ 个单步训练样本最终被存储在经验回放缓冲区中.

当经验回放缓冲区的数据量达到预设批次大小时, 模型训练随即启动. 首先从缓冲区随机采样一个迷你批次数据, 用于计算当前策略与初始策略的概率比; 接着采用广义优势估计 (GAE) 方法逐步骤计算优势函数和累计折扣回报; 随后分别计算裁剪目标函数 $L^{\mathrm{CLIP}}(\theta)$ 和价值函数损失 $L^{\mathrm{VF}}(\phi)$, 并基于这两个损失函数通过梯度下降法更新策略网络与价值网络的参数. 经过多次迭代后, 将更新后的策略参数赋值给旧策略. 上述过程循环执行直至满足终止条件, 具体算法流程参见算法 1.

算法 1. PCLog 算法.

输入: 初始策略参数 θ_0, 价值函数参数 ϕ_0, 裁剪阈值 ε, 总迭代次数 K, 日志向量序列 L_s, 事件序列 L_e, 学习率 α, 训练轮数 E;
输出: 优化后的策略参数 θ.

1. 初始化策略网络 π_{θ_0} 和价值函数 V_{ϕ_0}
2. **for** k = 1 to K **do**
3. 使用当前策略 π_{θ_k} 和序列 L_s、L_e 收集轨迹 τ 计算优势估计 $\hat{A}_t$ 和折扣回报 R_t
4. **for** e = 1 to E **do**
5. 从 $\{\tau\}$ 中采样小批量 $\{(s_t, a_t, r_t, s_{t+1})\}$
6. 计算重要性采样比率:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_0}(a_t|s_t)}$$

7. 计算裁剪替代目标函数:

$$L^{\mathrm{CLIP}}(\theta)=E_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \mathrm{CLIP}(r_t(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_t\right)\right]$$

8. 更新策略参数 θ 以最大化 $L^{\mathrm{CLIP}}(\theta)$

9. 通过最小化以下函数更新价值函数参数 ϕ:

$$L^{\mathrm{VF}}(\phi)=\mathbb{E}_t\left[\left(V_\phi(s_t)-R_t\right)^2\right]$$

10. **endfor**

11. 更新旧策略 $\theta_{k-1}\leftarrow\theta_k$

12. **endfor**

13. **return** 最终策略参数 θ

3.3 异常检测与反馈调节机制

训练完成后, 模型可部署于日志异常检测任务中, 根据异常的表现形式与语义特征, 日志异常可划分为 3 类: 日志点异常 (log point anomaly)、日志执行序列异常 (log execution sequence anomaly)、日志模板变更引发的新型异常.

日志点异常是指单个日志事件在时间间隔或内容特征上偏离正常分布的情况. 为了刻画正常行为, 系统在训练阶段统计各类型日志事件之间的平均时间间隔及其方差范围, 并构建基于上下文的语义嵌入分布. 在推理阶段, 系统实时监测输入事件的时间间隔与语义一致性, 当观测到的间隔偏离超过阈值 δ_t 或嵌入距离超过预设限度时, 即判定该事件为点异常. 此类异常通常反映了模块延迟、执行阻塞或参数异常等问题. 模型输出的点异常不仅可独立作为检测结果, 也作为辅助信号参与序列异常的判断. 如 HDFS 数据集中“Block 分配”类日志正常间隔均值为 2.1 s, 某条日志间隔达 15 s; 内容维度异常表现为参数格式错误, 如日志参数 “blk_abc”不符合“blk+数字”的正常格式. 检测时通过统计时间间隔均值与参数正则规则, 偏离阈值 (3 倍标准差) 则判定为异常.

日志执行序列异常则关注事件顺序或逻辑关系的异常模式. 本文将日志序列的演化建模为一个强化学习过程: 当前日志序列的语义表示作为状态 s_t, 下一个标准化日志事件作为动作 a_t, 并通过近端策略优化 (PPO) 算法学习正常状态转移规律. 在推理阶段, 模型根据策略分布 $\pi(a_t|s_t)$ 生成多个候选事件; 若真实事件未出现在前 g 个高概率候选中, 则判定为执行序列异常. 该类异常多对应系统内部逻辑紊乱、调用链异常或依赖关系错误等情况 (如正常序列“5→5→22→26→23→21”变为“22→5→9→3→23→21”的顺序异常).

日志模板变更引发的新型异常则是由于系统升级导致日志模板新增或修改, 现有模型未学习过该模板正常模式的异常. 如 BGL 超级计算机硬件升级后, 新增“GPUTempMonitor: Temperature of GPU [gpu-id] is [temp] °C”模板, 原有模型易将正常温度监测日志误判为异常. 检测时通过性能监控识别模板变更, 收集新增模板正常日志微调模型以适配.

此外, 为应对日志模板演化带来的概念漂移问题, 当系统检测到未见事件时, 模型输出特殊标记<UNK>, 并将其纳入反馈池. 随后, 通过行为克隆模块利用反馈样本进行微调, 使模型能够快速适应新的日志模式, 从而保持检测性能的稳定性与鲁棒性.

通过引入上述机制, PCLog 实现了对时间偏差、顺序异常以及概念漂移的综合检测与自适应修正, 为复杂动态系统环境下的智能日志分析提供了更加稳健的解决方案.

给定一个日志序列 $L=[e_1,e_2,e_3]$ 及其对应的句子嵌入向量 $V=[v_1,v_2,v_3]$, 系统首先将第 1 个嵌入向量 v_1 作为模型输入. 模型输出对可能的下一个日志事件的概率分布. 若实际出现的下一个事件 e_2 位于概率最高的前 g 个事件之中, 则判定为正常事件. 该过程对序列中的每个后续事件重复执行. 仅当所有事件都满足前 g 条件时, 整个日志序列才被视为正常; 否则, 该序列将被标记为异常.

在自动化运维环境中, 异常检测模型往往难以持续保持高检测性能. 特别是在增量训练过程中, 随着新数据不断涌入, 模型可能逐渐偏离初始优化轨迹, 进而影响日志异常检测的整体表现. 实践中, 这种偏离主要表现为对已知异常检测的精确率下降, 导致系统难以同时满足实时性和精确性双重要求.

为了解决上述的性能漂移问题, 我们通过反馈驱动的模仿学习机制, 在模型性能下降时利用误判样本进行策略微调, 有效缓解了长期运行中性能漂移的问题. 当模型在滑动窗口上的召回率下降超过设定阈值时, 触发一次微调, 使策略适应最新的日志模式. 这些样本的真实标签是已知的, 用于模拟运维人员提供的纠正反馈. 具体实现如下, 当模型性能出现显著下降时, 我们假设在经过一定预测周期后可获得一批标注样本. 这些正确标注的样本可视为专家示范数据, 其数学表示为:

$$D_{\text{expert}} = \{(s_i, a_i)\}_{i=1}^{N} \tag{10}$$

其中, s_i 表示环境状态, a_i 表示在状态 s_i 下专家推荐的动作, N 表示带标签样本的总数. 基于这一思想, PCLog 采用行为克隆这一模仿学习方法, 从专家策略中进行学习. 通过该方法, 模型可以被逐步微调和修正, 从而有效提升其在动态环境中持续进行异常检测的能力. 该方法的整体框架如图 4 所示.

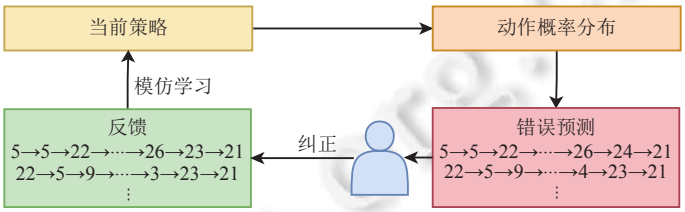


图 4 日志异常检测方法示意图

为了同时利用 PPO 优化与专家示范, 我们构建了一个联合目标函数. 设 λ 为平衡因子, 联合目标函数定义如下:

$$J(\theta) = L^{\text{CLIP}}(\theta) + \lambda \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{expert}}}[\log \pi_\theta(a \mid s)] \tag{11}$$

该联合目标函数中的第 2 项鼓励模型最大化专家动作的对数似然, 从而在给定状态 s 的情况下, 提高选择专家提供的动作 a 的概率. 在实际训练过程中, 我们采用指数衰减策略动态调整平衡因子 λ 的取值. 具体而言, 当有新的专家反馈可用时, λ 初始设定为约 0.5, 以便模型能够快速吸收专家知识, 从而迅速提升其检测精度. 而在缺乏新专家反馈的情况下, λ 会逐渐衰减至零, 使模型的训练重心平滑过渡到标准的 PPO 强化学习过程.

4 实验设置与分析

本节将通过回答以下 4 个研究问题来评估 PCLog 的性能.

- RQ1: 与基线方法相比, PCLog 在异常检测中的表现如何?
- RQ2: 在顺序训练数据下, PCLog 如何适应日志模式的变化?
- RQ3: 反馈数据对 PCLog 异常检测性能有何影响?
- RQ4: PCLog 在不同实验设置下的有效性如何?

4.1 实验设置

4.1.1 数据集

我们在 3 个广泛使用的公开数据集上进行了实验, 分别为 HDFS[55]、BGL[56]和 OpenStack[57]. HDFS 日志数据集是通过在 200 多个 Amazon EC2 节点上运行基于 Hadoop 的 MapReduce 作业生成的. 通过 BlockID 对日志进行分组, 该数据集可划分为 575 061 个日志序列, 其中包含 16 838 个异常样本. BGL 数据集采集自美国某实验室的一台 BlueGene/L 超级计算机, 包含 4 747 963 条日志记录, 其中 348 460 条 (约占 7.3%) 被标注为异常. OpenStack 数据集则是在 CloudLab 平台上部署 OpenStack Mitaka 版本时生成的. 这些数据集的详细信息汇总如表 2 所示. 我们从每个数据集中选取 80% 用于训练, 剩余 20% 用于测试. 同时, 训练仅使用正常日志, 异常数据保留在测试和微调阶段使用.

表 2 数据集信息

数据集名称	大小	采集时间跨度	日志条数
HDFS	1.55 GB	38.7 h	11 175 629
BGL	708 MB	7 m	4 747 963
OpenStack	58.5 MB	29.5 h	207 820

4.1.2 评估指标

预测结果分为 4 类: 真正例 (*TP*)、真反例 (*TN*)、假正例 (*FP*) 和假反例 (*FN*). 基于上述 4 类预测结果, 本文采用精确率 (*Precision*)、召回率 (*Recall*) 和 *F*1 值 (*F*1-*score*) 这 3 个指标来评估 PCLog 的性能. 其计算公式如下:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{14}$$

4.1.3 基线方法

在基线方法的选择上, 本文力求覆盖不同类别的代表性技术, 以保证对比的全面性与公平性. 具体而言, 选择 PCA[58]、IM (invariant mining)[19]、LogCluster[18]作为传统方法的代表, 其中 PCA 通过降维方式挖掘日志特征的主要成分, IM 基于不变量约束规则检测异常, LogCluster 则利用日志模板聚类方法进行异常模式识别. 这 3 类方法实现简单、可解释性较好, 是传统日志分析的重要代表.

在深度学习方法方面, 本文采用 DeepLog[26]与 LogAnomaly[59]作为基线. DeepLog 基于 LSTM 模型进行序列建模, 是较早应用深度学习于日志异常检测的工作, 代表了时间序列预测范式; LogAnomaly 则结合了注意力机制与上下文特征建模, 在异常检测准确性和鲁棒性方面均达到较高水平, 两者均属于近年来该领域的代表性甚至是 SOTA 方法.

而在强化学习方法方面, 本文则选取 QLLog[49]作为对照. QLLog 是少数将强化学习范式引入日志异常检测的工作之一, 其思想与本文方法最为相关, 为对比不同 RL 框架的性能提供了直接参考.

基于上述基线方法, 我们在 HDFS、BGL 和 OpenStack 数据集上, 使用相同的实验参数、训练集和测试集进行了对比实验. 所有实验均选取并记录最佳结果.

4.1.4 实验环境

我们基于 Python 3.8 和 PyTorch 1.10.1 实现了 PCLog. 语义向量与状态向量的维度均为 100. 策略网络和价值网络结构相同, 均由 3 层全连接层组成, 层维度依次为 256、128 和 64. 动作数量设置为数据集中唯一日志事件的数量, 分组窗口大小设为 20. 超参数的具体设置见表 3. 所有实验均在配置为 64 核 Intel(R) Xeon(R) E5-2640 CPU、256 GB 内存及 16 GB 显存 GPU 的 Linux 服务器上进行.

表 3 训练超参数设置

参数名	数值	说明
actor_lr	0.000 3	策略网络学习率
critic_lr	0.001	价值网络学习率
ε	0.2	剪切比例
γ	0.99	折扣因子
Epochs	100	优化迭代次数

4.2 RQ1: 与基线方法相比, PCLog 在异常检测中的表现如何?

- 动机: 现有方法 (如 DeepLog、QLLog) 在动态日志模式下精度低、泛化弱, 需验证 PCLog 的整体优势.
- 评估方法: 采用 HDFS、BGL、OpenStack 这 3 个数据集, 训练集为 80% 正常日志, 测试集为 20% 正常日志+

全部异常日志; 基线方法覆盖传统方法 (PCA、IM、LogCluster)、深度学习方法 (DeepLog、LogAnomaly)、强化学习方法 (QLLog); 以精确率、召回率、*F*1 值为指标.

本文方法 PCLog 在 HDFS、BGL 和 OpenStack 数据集上的实验结果如表 4 所示 (加粗数字为最优值). 结果表明, PCLog 在所有数据集上的整体异常检测性能均优于所有基线方法, 精确率、召回率和 *F*1 值均超过 0.95.

表 4 HDFS、BGL、OpenStack 数据集上的实验结果

方法	HDFS数据集			BGL数据集			OpenStack数据集		
	精确率	召回率	*F*1 值	精确率	召回率	*F*1 值	精确率	召回率	*F*1 值
PCA[58]	**0.978**	0.668	0.794	0.501	0.612	0.551	0.778	0.992	0.872
IM[19]	0.882	0.948	0.914	0.836	0.988	0.906	0.832	**1.000**	0.908
LogCluster[18]	0.873	0.742	0.802	0.436	0.858	0.578	0.913	0.991	0.950
DeepLog[26]	0.938	0.934	0.936	0.912	0.972	0.941	0.944	0.993	0.968
LogAnomaly[59]	0.961	0.944	0.952	**0.971**	0.942	0.956	**0.971**	0.975	0.973
QLLog[49]	0.946	0.987	0.966	0.952	0.964	0.958	0.952	0.997	0.974
PCLog	0.964	**0.992**	**0.978**	0.953	**0.990**	**0.971**	0.961	1.000	**0.980**

PCLog 在 3 个数据集上优于基线方法, 核心原因如下.

对比传统方法 (PCA、IM、LogCluster): 传统方法依赖手工特征或聚类规则, 丢失日志序列与语义信息. 如因降维丢失“Block 分配-数据传输”序列关联导致 PCA 在 HDFS 数据集召回率为 66.8%; 而 PCLog 通过句嵌入保留语义、PPO 捕捉序列依赖, 召回率达 99.2%. 对比深度学习方法 (DeepLog、LogAnomaly): DeepLog 用模板索引输入, 信息稀疏导致新模板适应弱; LogAnomaly 虽有词嵌入但无动态反馈. 在 BGL 数据集 (20% 模板变更) 中, DeepLog *F*1 值为 94.1%, PCLog 因模仿学习微调, *F*1 值达 97.1%. 对比强化学习方法 (QLLog): QLLog 基于 Q-learning 用离散 Q 表, 难以处理高维日志向量; PCLog 用 PPO 连续策略优化, 新增行为克隆修正, 在 OpenStack 数据集精确率为 96.1%, 高于 QLLog 的 95.2%, 且其基于离散 Q 表的策略学习方法难以充分捕捉日志序列中的复杂模式. 与之对比, PCLog 将日志序列建模为策略优化的轨迹, 直接对日志事件的执行策略进行优化, 使其能够在不同类型的日志特征和异常模式上表现出更优的检测性能和更强的泛化能力.

RQ1: 与基线方法相比, PCLog 在异常检测中的表现如何?

回答: 在 HDFS、BGL 和 OpenStack 这 3 个真实数据集上的实验结果表明, PCLog 在召回率、*F*1 值等主要指标上均显著优于现有的 6 种基线方法. 这说明本文提出的基于 PPO 与行为克隆的框架能够有效捕捉日志事件的时序模式和上下文信息, 进而提升了异常检测的整体性能.

4.3 RQ2: 在顺序训练数据下, PCLog 如何适应日志模式的变化?

- 动机: 系统升级导致日志模式持续变化, 现有方法需频繁重训, 需验证 PCLog 的动态适应能力.
- 评估方法: 将数据集按时间戳排序, 用 10%–80% 训练数据增量训练模型, 测试剩余数据性能; 重点关注召回率下降幅度 (≤3% 为适应良好); 与 PCLog 方法对比, 验证强化学习基础模型的适应潜力.

图 5 展示了随着日志数据量增加, 日志数据集中出现的模板数占全部模板总数的比例的变化. 如图 5 所示, HDFS 和 OpenStack 数据集的采集时间较短, 均不足 2 天, 且采集期间系统未进行重大更新, 因此其模板数量在后期趋于稳定, 增长速度较为平缓. 相比之下, BGL 数据集的采集时间跨度长达 7 m, 期间系统不断迭代更新, 导致模板数量持续上升, 增长曲线波动较大.

图 6 展示了在 HDFS、BGL 和 OpenStack 数据集上进行的实验结果. 实验中, 日志首先按时间戳顺序严格排序, 随后使用不同比例的训练数据训练模型, 以模拟 PCLog 对不断变化日志模式的适应能力.

从实验结果可以看出, 由于采集周期较短, HDFS 和 OpenStack 数据集的日志内容和序列模式变化较小. 此外, OpenStack 数据集规模较小且系统更新频率较低, 因此即使使用较少的训练数据, PCLog 仍能取得优异的表现, 各项指标均超过 0.9.

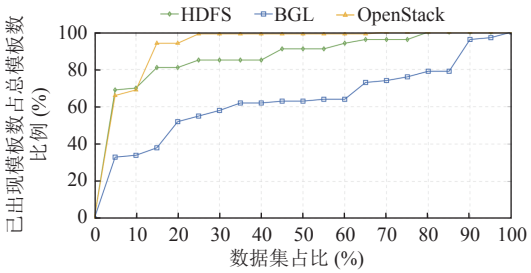
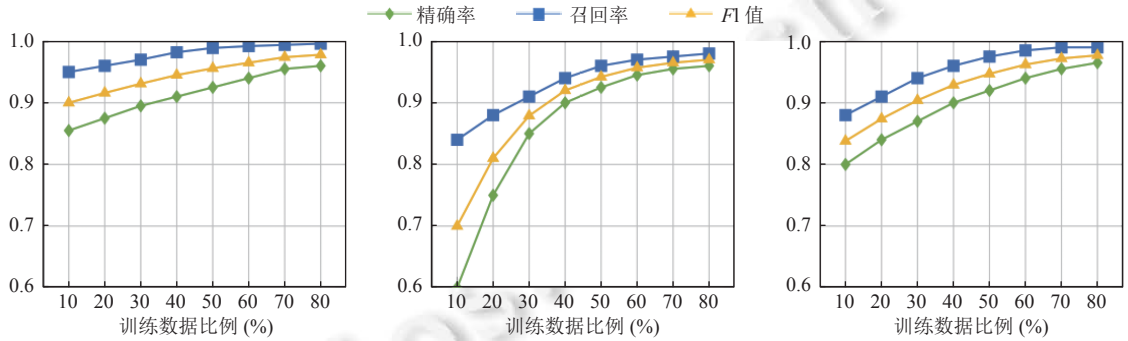


图 5 日志模板覆盖率随数据量增长的变化



(a) HDFS 数据集 (b) BGL 数据集 (c) OpenStack 数据集

图 6 3 个数据集在不同训练数据比例下的实验结果

相比之下, BGL 数据集的采集时间跨度较长, 且系统持续更新, 导致日志内容和序列模式频繁变化, 使得模型在初期训练阶段的异常检测性能较低. 但随着训练数据比例的增加, PCLog 能够逐步学习并适应新的日志模式, 无需频繁对历史数据进行重训练, 从而实现性能的稳定提升. 以上结果表明, PCLog 具备较强的日志模式适应能力, 能够在保持稳定性能的同时有效完成异常检测任务.

RQ2: 在顺序训练数据下, PCLog 如何适应日志模式的变化?

回答: 在模拟系统升级和日志模式演化的概念漂移场景中, PCLog 通过反馈驱动的自适应微调机制, 能够在长时间运行过程中保持检测性能的稳定, 而无需频繁重训练. 这表明该方法在应对实际系统运行中的动态变化方面具有良好的适应性.

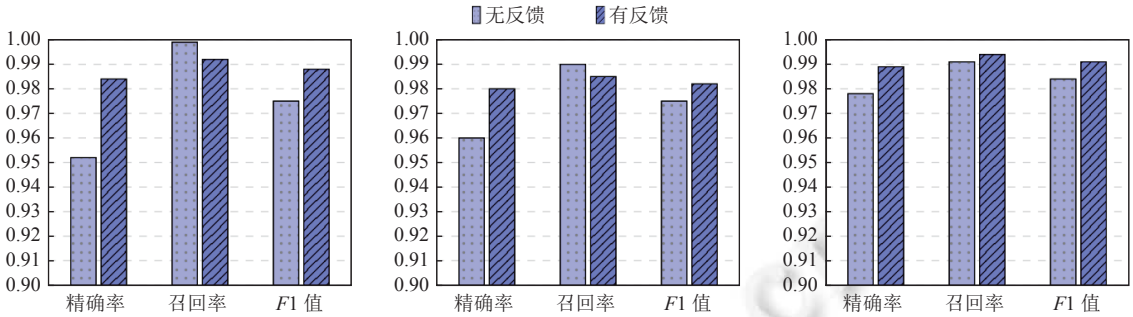
4.4 RQ3: 反馈数据对 PCLog 异常检测的影响如何?

- 动机: 模型长期运行容易性能漂移, 需明确反馈数据作用与依赖程度, 解决“反馈样本获取与用量”问题.
- 评估方法: 为进一步阐明反馈机制, 本节实验中的“专家示范数据”由模型在测试集上的误报样本 (false positive) 模拟构成, 即模型误判为异常、但实际正常的日志序列 (人工标注确认正常), 对比不同比例下精确率提升幅度与微调耗时, 验证实际可行性.

图 7 展示了 PCLog 在不使用反馈数据和引入反馈数据两种情况下的性能变化. 可以看出, 引入反馈数据后, 模型的性能显著提升, 主要体现在精确率的提高上. 这是由于模型在训练阶段仅使用了正常日志序列, 而反馈数据主要由假阳性样本构成, 即模型误判为异常、实则正常的日志. 这些反馈样本有效帮助模型纠正误判, 从而提升了整体检测精度.

通过引入这些反馈样本, 模型能够更好地学习正常日志的特征, 从而有效减少误报数量, 直接提升精确率. 实验结果表明, 对于 HDFS、BGL 和 OpenStack 数据集, 即使仅利用模型自身产生的少量误报样本作为反馈, 也能有效提升模型的精确率. 上述实验在离线环境下验证了反馈机制的技术可行性. 然而, 在真实的在线系统中, “当模型性能出现显著下降时可获得一批标注样本”这一条件的满足, 依赖于运维人员对误报进行人工确认和标注. 本研究

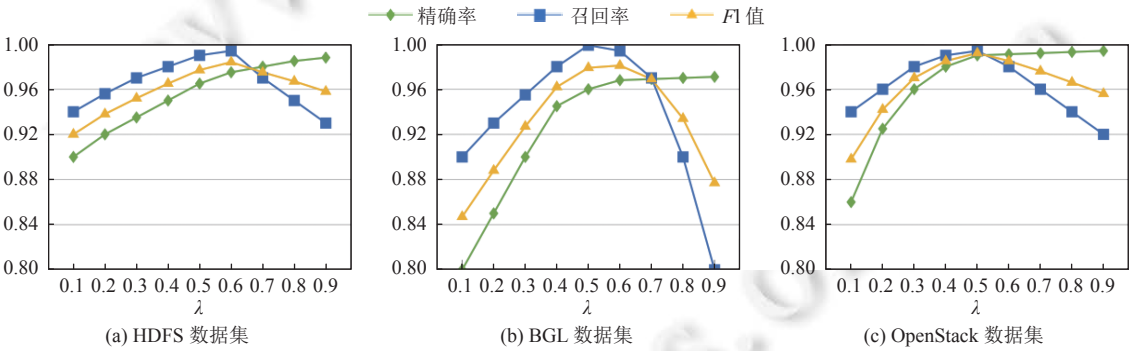
目前通过离线数据集上的模拟验证了此流程, 而反馈机制在实际生产环境中的标注频率、数据量及其对人工的依赖程度, 是未来工作中需要进一步研究和量化的重要方面.



(a) HDFS 数据集 (b) BGL 数据集 (c) OpenStack 数据集

图 7 3 个数据集反馈前后性能比较

另一方面, 由于模型本身在识别异常样本方面已具备较强能力, 因此在优化前后召回率的变化较小. 总体来看, 实验结果验证了 PCLog 在引入专家轨迹并结合模仿学习进行模型优化方面的有效性, 尤其是在减少误报和提升检测精度方面表现尤为突出. 图 8 展示了 PCLog 在 HDFS、BGL 和 OpenStack 数据集上, 不同损失权重参数 λ 取值下的实验结果. 实验结果表明, 随着权重参数 λ 的增大, PCLog 模型逐渐更多地依赖专家数据分布, 所学习的策略趋向于接近专家的决策过程. 这使得精确率呈现出持续上升的趋势, 表明利用专家知识能够显著提升检测的准确性. 然而, 当 λ 超过某一最优阈值后, 模型对专家数据的过度依赖反而削弱了其识别异常样本的能力, 导致召回率出现非单调变化——先升高后下降. 过大的 λ 可能使模型过于局限于专家知识, 降低了对新异常模式的适应性, 从而影响整体性能表现.



(a) HDFS 数据集 (b) BGL 数据集 (c) OpenStack 数据集

图 8 不同 λ 取值下, 3 种数据集上的实验结果比较

为了解决上述问题, 训练过程中采用指数衰减策略动态调整参数 λ 的取值. 起始阶段将 λ 设定为约 0.5, 使模型能够充分利用专家知识, 在训练初期快速提升精确率. 随着训练的进行, λ 的值逐渐减小, 减少对专家数据的依赖, 增强模型的泛化能力和识别新异常模式的能力.

为了评估 PCLog 在实际部署中的可行性, 本节还对比了 PCLog 与基线方法在 HDFS、BGL 和 OpenStack 数据集上的训练时间开销, 具体结果如表 5 所示 (加粗数字为最优值). 实验结果表明, PCLog 通过融合强化学习与模仿学习机制, 显著提升了训练效率. 首先, PPO 算法通过经验回放缓冲区实现了数据的复用, 提升了训练效率; 其次, PCLog 的策略网络和价值网络结构相对简单, 仅由 3 层全连接层构成, 计算开销较小. 此外, 在引入模仿学习进行微调时, 由于只需在少量专家数据上进行训练, 其额外时间开销极低. 这表明 PCLog 具备应用于实际在线环境的潜力.

表 5　3 个数据集单轮训练时间对比 (s)

方法	HDFS	BGL	OpenStack
DeepLog	64	28	12
LogAnomaly	102	30	28
QLLog	34	16	10
PCLog	**15**	**7**	**6**

RQ3: 反馈数据对 PCLog 异常检测性能有何影响?

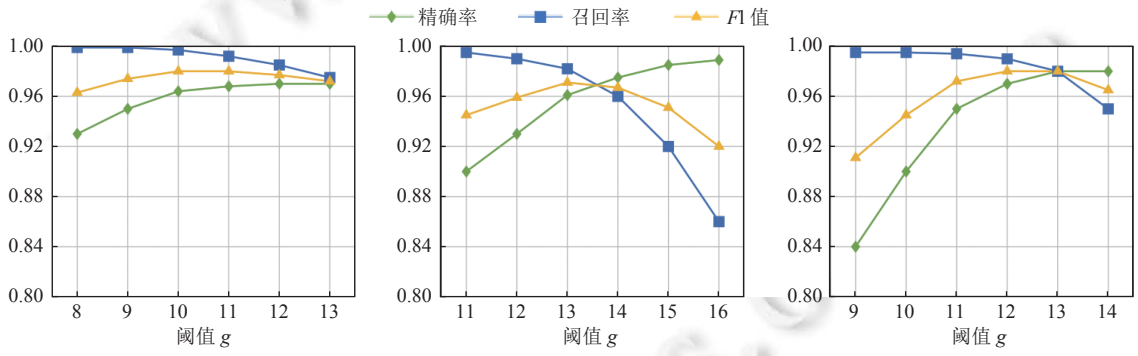
回答: 通过消融实验, 我们可以得出: 当移除反馈数据或减少反馈比例时, PCLog 的精确率和召回率均出现明显下降; 而在引入反馈数据后, 检测性能得到持续提升. 这一结果证明了基于误判样本的行为克隆机制对于缓解模型漂移、提升检测精度具有关键作用.

4.5 RQ4: 不同阈值 g 下 PCLog 的检测性能如何?

● 动机: 阈值 g 是序列异常判定核心参数, g 过小易误判正常事件, g 过大易漏检异常, 需明确各数据集最优 g 值及模型鲁棒性, 为部署提供参数依据.

● 评估方法: 在 HDFS、BGL、OpenStack 数据集上, 将 g 设为 8–16 的连续整数; 以精确率、召回率、$F1$ 值为指标, 每个 g 值重复实验 5 次取均值; 分析 $F1$ 值峰值对应的最优 g 及阈值波动对性能的影响.

本节对不同阈值 g 下的检测性能进行对比分析. 图 9 展示了在 HDFS、BGL 和 OpenStack 这 3 个数据集上, 随着阈值 g 变化的实验结果. 如图 9 所示, 随着阈值 g 的增加, 3 个数据集上的召回率逐渐下降, 而精确率则呈上升趋势. 这是因为阈值 g 决定了被判定为正常日志的概率范围. 当 g 增大时, 模型会将更多日志判定为正常, 从而导致被正确识别为异常的日志数量减少, 召回率随之下降. 同时, 被识别为异常的日志数量减少, 异常样本中真正异常的比例上升, 因此精确率提高. 实验结果表明, 当阈值 g 设置在 11 附近时, 模型在 3 个数据集上均达到了整体最优的检测性能.



(a) HDFS 数据集　(b) BGL 数据集　(c) OpenStack 数据集

图 9　在不同阈值 g 下 3 个数据集的检测性能表现

RQ4: PCLog 在不同实验设置下的有效性如何?

回答: 参数敏感性分析表明, 当 $g\approx11$ 时, PCLog 在 3 个数据集上的性能最优, 且在相邻区间内波动较小. 这说明该方法对阈值设置具有一定的鲁棒性, 能够在合理范围内保持稳定的检测效果.

5 总结与展望

5.1 总　结

本文提出了一种融合强化学习与模仿学习的日志异常检测方法 PCLog. 该方法首先将日志序列转换为适用于强化学习的状态-动作表示形式, 并基于近端策略优化算法构建策略优化模型. 在该框架中, 检测模型被建模为智

能体, 日志的语义向量作为环境状态, 日志事件被视为智能体的动作, 通过学习系统正常运行时的行为模式, 实现对异常行为的精准识别.

针对实际系统中异常检测模型在长期运行过程中缺乏有效反馈、易出现性能退化的问题, PCLog 引入了一种基于现有检测网络的自适应微调机制. 当系统检测到模型性能下降时, 自动收集部分模型预测结果构建专家示范数据, 通过行为克隆方法对模型进行精调, 提升其鲁棒性和泛化能力, 从而增强系统在动态环境下的持续检测能力.

在 HDFS、BGL 和 OpenStack 这 3 个公开日志数据集上的实验证明, PCLog 相较于主流的基线方法, 在精确率、召回率和 $F1$ 值等指标上均取得了更优性能. 同时, 该方法在应对日志模式变化方面展现出良好的适应性, 具备较高的实用价值和推广潜力, 能够为实际系统中的日志异常检测任务提供一种高效、可靠的解决方案. 任务提供一种高效、可靠的解决方案.

5.2 未来展望

尽管 PCLog 在多个真实数据集上展现出优异的检测性能与较强的自适应能力, 但本文工作仍存在一些不足: (1) 反馈数据依赖人工标注, 当前反馈样本需人工确认假阳性, 日均 10 亿条日志场景中, 0.5% 样本仍需 500 万条标注, 成本较高; (2) 极端模板变更下性能波动, 系统新增 50% 以上新模板时, 反馈样本短期内难以覆盖, 模型微调后召回率仍下降 1%–2%, 需 2–3 轮反馈恢复; (3) 超参数需人工调整: 阈值 g (序列异常判定) 与 λ (模仿学习权重) 需按数据集调整, HDFS 最优 g=11、λ=0.5, BGL 最优 g=12、λ=0.4, 无统一自适应机制.

针对上述不足, 我们未来的研究可从以下几个方面展开. 第一, 探索更加自动化和低成本的反馈数据收集方式, 减少对人工标注的依赖. 结合规则引擎, 自动筛选高置信度正常样本 (如参数格式合规、时间间隔正常的假阳性样本), 目标将人工标注比例从 0.5% 降至 0.1%, 降低标注成本与运维负担. 第二, 设计更具鲁棒性的自适应机制, 以提升模型在高噪声与开放环境下的稳定性. 引入日志领域预训练模型 (如 LogBERT), 先在大规模通用日志数据集上完成预训练, 再在目标数据集上进行针对性微调, 缩短极端日志模板变更场景下的模型适应时间, 增强对动态环境的适配能力. 第三, 引入自适应阈值或动态调参方法, 降低人工设定超参数对性能的影响. 设计基于强化学习的超参数优化器, 实时根据模型输出的精确率、召回率动态调整序列异常判定阈值 g 与模仿学习平衡因子 λ, 实现超参数的端到端优化, 避免人工调参的主观性与繁琐性. 第四, 结合预训练模型或图神经网络等新兴技术, 进一步提升对未见模板和复杂日志模式的检测能力. 通过预训练模型的语义理解优势与图神经网络对事件关联关系的建模能力, 强化对新型日志模板、复杂交互序列的异常识别效果. 我们相信, 上述方向的探索将有助于推动日志异常检测方法向更加智能化、鲁棒化和可部署化的方向发展.

References

[1] He SL, Zhu JM, He PJ, Lyu MR. Experience report: System log analysis for anomaly detection. In: Proc. of the 27th IEEE Int'l Symp. on Software Reliability Engineering. Ottawa: IEEE, 2016. 207–218. [doi: 10.1109/ISSRE.2016.21]

[2] You Y, Wang H, Ren T, Gu SH, Sun JL. Storage design of tracing-logs for application performance management system. Ruan Jian Xue Bao/Journal of Software, 2021, 32(5): 1302–1321 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6234.htm [doi: 10.13328/j.cnki.jos.006234]

[3] Jia T, Li Y, Wu ZH. Survey of state-of-the-art log-based failure diagnosis. Ruan Jian Xue Bao/Journal of Software, 2020, 31(7): 1997–2018 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6045.htm [doi: 10.13328/j.cnki.jos.006045]

[4] Bao HY, Yin KL, Cao L, Li SN, Sun YJ, Yin HF, Tang RM, Hou Y, Wang SQ, Pei D, Yang XQ, Wang LX. AIOps in practice: Status Quo and standardization. Ruan Jian Xue Bao/Journal of Software, 2023, 34(9): 4069–4095 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6876.htm [doi: 10.13328/j.cnki.jos.006876]

[5] Jiang ZM, Hassan AE, Flora P, Hamann G. Abstracting execution logs to execution events for enterprise applications (short paper). In: Proc. of the 8th Int'l Conf. on Quality Software. Oxford: IEEE, 2008. 181–186. [doi: 10.1109/QSIC.2008.50]

[6] Makanju AAO, Zincir-Heywood AN, Milios EE. Clustering event logs using iterative partitioning. In: Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Paris: ACM, 2009. 1255–1264. [doi: 10.1145/1557019.1557154]

[7] Du M, Li FF. Spell: Streaming parsing of system event logs. In: Proc. of the 16th Int'l Conf. on Data Mining. Barcelona: IEEE, 2016. 859–864. [doi: 10.1109/ICDM.2016.0103]

[8] Di Gennaro G, Buonanno A, Palmieri FAN. Considerations about learning Word2Vec. The Journal of Supercomputing, 2021, 77(1–2): 12320–12335. [doi: 10.1007/s11227-021-03743-2]

[9] Brochier R, Guille A, Velcin J. Global vectors for node representations. In: Proc. of the 2019 World Wide Web Conf. San Francisco: ACM, 2019. 2587–2593. [doi: 10.1145/3308558.3313595]

[10] Joulin A, Grave E, Bojanowski P, Douze M, Jégou H, Mikolov T. FastText.zip: Compressing text classification models. arXiv:1612.03651, 2016.

[11] Guo HX, Yuan SH, Wu XT. LogBERT: Log anomaly detection via BERT. In: Proc. of the 2021 Int'l Joint Conf. on Neural Networks. Shenzhen: IEEE, 2021. 1–8. [doi: 10.1109/IJCNN52387.2021.9534113]

[12] Almodovar C, Sabrina F, Karimi S, Azad S. LogFit: Log anomaly detection using fine-tuned language models. IEEE Trans. on Network and Service Management, 2024, 21(2): 1715–1723. [doi: 10.1109/TNSM.2024.3358730]

[13] Liang Y, Zhang YY, Xiong H, Sahoo R. Failure prediction in IBM BlueGene/L event logs. In: Proc. of the 7th IEEE Int'l Conf. on Data Mining. Omaha: IEEE, 2007. 583–588. [doi: 10.1109/ICDM.2007.46]

[14] Liu ZL, Qin T, Guan XH, Jiang HZ, Wang CX. An integrated method for anomaly detection from massive system logs. IEEE Access, 2018, 6: 30602–30611. [doi: 10.1109/access.2018.2843336]

[15] Li T, Ma JF, Pei QQ, Shen YL, Lin C, Ma SQ, Obaidat MS. AClog: Attack chain construction based on log correlation. In: Proc. of the 2019 IEEE Global Communications Conf. Waikoloa: IEEE, 2019. 1–6. [doi: 10.1109/GLOBECOM38437.2019.9013518]

[16] Ying S, Wang BM, Wang L, Li QS, Zhao YS, Shang JG, Huang H, Cheng GL, Yang Z, Geng JY. An improved KNN-based efficient log anomaly detection method with automatically labeled samples. ACM Trans. on Knowledge Discovery from Data (TKDD), 2021, 15(3): 34. [doi: 10.1145/3441448]

[17] Landauer M, Wurzenberger M, Skopik F, Stanni G, Filzmoser P. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. Computers & Security, 2018, 79: 94–116. [doi: 10.1016/j.cose.2018.08.009]

[18] Lin QW, Zhang HY, Lou JG, Zhang Y, Chen XW. Log clustering based problem identification for online service systems. In: Proc. of the 38th Int'l Conf. on Software Engineering Companion. Austin: IEEE, 2016. 102–111.

[19] Lou JG, Fu Q, Yang SQ, Xu Y, Li J. Mining invariants from console logs for system problem detection. In: Proc. of the 2010 USENIX Annual Technical Conf. Boston: ACM, 2010. 24.

[20] Zhang B, Zhang HY, Moscato P, Zhang AZ. Anomaly detection via mining numerical workflow relations from logs. In: Proc. of the 2020 Int'l Symp. on Reliable Distributed Systems. Shanghai: IEEE, 2020. 195–204. [doi: 10.1109/SRDS51746.2020.00027]

[21] Debnath B, Solaimani M, Gulzar MAG, Arora N, Lumezanu C, Xu JW, Zong B, Zhang H, Jiang GF, Khan L. Loglens: A real-time log analysis system. In: Proc. of the 38th IEEE Int'l Conf. on Distributed Computing Systems. Vienna: IEEE, 2018. 1052–1062. [doi: 10.1109/ICDCS.2018.00105]

[22] Jin YJ, Qiu CL, Sun L, Peng X, Zhou JN. Anomaly detection in time series via robust PCA. In: Proc. of the 2nd IEEE Int'l Conf. on Intelligent Transportation Engineering. Singapore: IEEE, 2017. 352–355. [doi: 10.1109/ICITE.2017.8056937]

[23] Salman M, Husna D, Apriliani SG, Pinem JG. Anomaly based detection analysis for intrusion detection system using big data technique with learning vector quantization (LVQ) and principal component analysis (PCA). In: Proc. of the 2018 Int'l Conf. on Artificial Intelligence and Virtual Reality. Nagoya: ACM, 2018. 20–23. [doi: 10.1145/3293663.3293683]

[24] Carter J, Mancoridis S, Galinkin E. Fast, lightweight IoT anomaly detection using feature pruning and PCA. In: Proc. of the 37th ACM/SIGAPP Symp. on Applied Computing. ACM, 2022, 133–138. [doi: 10.1145/3477314.3508377]

[25] Jin MX, Lv AR, Zhu YP, Wen ZJ, Zhong YB, Zhao ZX, Wu J, Li HJ, He HH, Chen FY. An anomaly detection algorithm for microservice architecture based on robust principal component analysis. IEEE Access, 2020, 8: 226397–226408. [doi: 10.1109/ACCESS.2020.3044610]

[26] Du M, Li FF, Zheng GN, Srikumar V. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 1285–1298. [doi: 10.1145/3133956.3134015]

[27] Du M, Chen Z, Liu C, Oak R, Song D. Lifelong anomaly detection through unlearning. In: Proc. of the 2019 ACM SIGSAC Conf. on Computer and Communications Security. London: ACM, 2019. 1283–1297. [doi: 10.1145/3319535.3363226]

[28] Zhou JW, Qian YJ, Zou QT, Liu P, Xiang JW. DeepSyslog: Deep anomaly detection on Syslog using sentence embedding and metadata. IEEE Trans. on Information Forensics and Security, 2022, 17: 3051–3061. [doi: 10.1109/TIFS.2022.3201379]

[29] Bursic S, Cuculo V, D'Amelio A. Anomaly detection from log files using unsupervised deep learning. In: Proc. of the 2020 Int'l Symp. on Formal Methods. Porto: Springer, 2020. 200–207. [doi: 10.1007/978-3-030-54994-7_15]

[30] Baril X, Coustié O, Mothe J, Teste O. Application performance anomaly detection with LSTM on temporal irregularities in logs. In: Proc.

of the 29th ACM Int'l Conf. on Information & Knowledge Management. ACM, 2020. 1961–1964. [doi: 10.1145/3340531.3412157]

[31] Li XY, Chen PF, Jing LX, He ZL, Yu GB. SwissLog: Robust and unified deep learning based log anomaly detection for diverse faults. In: Proc. of the 31st IEEE Int'l Symp. on Software Reliability Engineering. Coimbra: IEEE, 2020. 92–103. [doi: 10.1109/ISSRE5003.2020.00018]

[32] Li Y, Sun JC, Huang WG, Tian XH. Detecting anomaly in large-scale network using mobile crowdsourcing. In: Proc. of the 2019 IEEE Conf. on Computer Communications. Paris: IEEE, 2019. 2179–2187. [doi: 10.1109/INFOCOM.2019.8737541]

[33] Le VH, Zhang HY. Log-based anomaly detection without log parsing. In: Proc. of the 36th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: IEEE, 2021. 492–504. [doi: 10.1109/ASE51524.2021.9678773]

[34] Fu YY, Liang K, Xu J. MLog: Mogrifier LSTM-based log anomaly detection approach using semantic representation. IEEE Trans. on Services Computing, 2023, 16(5): 3537–3549. [doi: 10.1109/TSC.2023.3289488]

[35] Guo HC, Yang J, Liu JH, Bai JQ, Wang BY, Li ZJ, Zheng TQ, Zhang B, Peng JR, Tian Q. LogFormer: A pre-train and tuning pipeline for log anomaly detection. In: Proc. of the 38th AAAI Conf. on Artificial Intelligence. Vancouver: AAAI, 2024. 135–143. [doi: 10.1609/aaai.v38i1.27764]

[36] Yan LJ, Luo C, Shao R. Discrete log anomaly detection: A novel time-aware graph-based link prediction approach. Information Sciences, 2023, 647: 119576. [doi: 10.1016/j.ins.2023.119576]

[37] He ZY, Tang YN, Zhao KQ, Liu JM, Chen W. Graph-based log anomaly detection via adversarial training. In: Proc. of the 9th Int'l Symp. on Dependable Software Engineering: Theories, Tools, and Applications. Nanjing: Springer, 2023. 55–71. [doi: 10.1007/978-981-99-8664-4_4]

[38] Li Z, Shi JY, van Leeuwen M. Graph neural networks based log anomaly detection and explanation. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering: Companion Proc. Lisbon: ACM, 2024. 306–307. [doi: 10.1145/3639478.3643084]

[39] Chu GJ, Wang JY, Qi Q, Sun HF, Zhuang ZR, He B, Jing YH, Zhang L, Liao JX. Anomaly detection on interleaved log data with semantic association mining on log-entity graph. IEEE Trans. on Software Engineering, 2025, 51(2): 581–594. [doi: 10.1109/TSE.2025.3527856]

[40] Xie YX, Yang K. Log anomaly detection by adversarial autoencoders with graph feature fusion. IEEE Trans. on Reliability, 2024, 73(1): 637–649. [doi: 10.1109/TR.2023.3305376]

[41] Xia B, Yin JJ, Xu J, Li Y. LogGAN: A sequence-based generative adversarial network for anomaly detection based on system logs. In: Proc. of the 2nd Int'l Conf. on Science of Cyber Security. Nanjing: Springer, 2019. 61–76. [doi: 10.1007/978-3-030-34637-9_5]

[42] Tao SM, Liu YL, Meng WB, Ren ZM, Yang H, Chen X, Zhang L, Xie YM, Su C, Oiao X, Tian WN, Zhu YC, Han T, Qin Y, Li Y. Biglog: Unsupervised large-scale pre-training for a unified log representation. In: Proc. of the 31st IEEE/ACM Int'l Symp. on Quality of Service. Orlando: IEEE, 2023. 1–11. [doi: 10.1109/IWQoS57198.2023.10188759]

[43] Xu JJL, Yang RC, Huo YT, Zhang CY, He PJ. DivLog: Log parsing with prompt enhanced in-context learning. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 199. [doi: 10.1145/3597503.3639155]

[44] Liu YL, Tao SM, Meng WB, Wang JY, Ma WB, Chen YH, Zhao YQ, Yang H, Jiang YF. Interpretable online log analysis using large language models with prompt strategies. In: Proc. of the 32nd IEEE/ACM Int'l Conf. on Program Comprehension. Lisbon: ACM, 2024. 35–46. [doi: 10.1145/3643916.3644408]

[45] Ma LP, Yang WD, Xu B, Jiang SH, Fei B, Liang JQ, Zhou MJ, Xiao YH. KnowLog: Knowledge enhanced pre-trained language model for log understanding. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 32. [doi: 10.1145/3597503.3623304]

[46] Wang XH, Song JX, Zhang X, Tang JS, Gao WH, Lin QW. LogOnline: A semi-supervised log-based anomaly detector aided with online learning mechanism. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Luxembourg: IEEE, 2023. 141–152. [doi: 10.1109/ASE56229.2023.00043]

[47] Han SB, Wu QH, Zhang H, Qin B, Hu JK, Shi XG, Liu LF, Yin X. Log-based anomaly detection with robust feature extraction and online learning. IEEE Trans. on Information Forensics and Security, 2021, 16: 2300–2311. [doi: 10.1109/TIFS.2021.3053371]

[48] Yuan YL, Adhatarao SS, Lin MK, Yuan YC, Liu ZL, Fu XM. ADA: Adaptive deep log anomaly detector. In: Proc. of the 2020 IEEE Conf. on Computer Communications. Toronto: IEEE, 2020. 2449–2458. [doi: 10.1109/INFOCOM41043.2020.9155487]

[49] Duan XY, Ying S, Yuan WL, Cheng HL, Yin X. QLLog: A log anomaly detection method based on Q-learning algorithm. Information Processing & Management, 2021, 58(3): 102540. [doi: 10.1016/j.ipm.2021.102540]

[50] Zhou JW, Gao YY, Zhu Y, Yu XT, Yang YC, Tan C, Xiang JW. DRLLog: Deep reinforcement learning for online log anomaly detection. IEEE Trans. on Network and Service Management, 2025, 22(3): 2382–2395. [doi: 10.1109/tnsm.2025.3542595]

[51] Liu Quan, Yan Jie, Wu Lan. Offline reinforcement learning method with diffusion model and expectation maximization. Ruan Jian Xue

Bao/Journal of Software, 2025, 36(10): 4695–4709 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7296.htm [doi: 10.13328/j.cnki.jos.007296]

[52] Wang C, Erfani S, Alpcan T, Leckie C. Detecting anomalous agent decision sequences based on offline imitation learning. In: Proc. of the 23rd Int'l Conf. on Autonomous Agents and Multiagent Systems. Auckland: ACM, 2024. 2543–2545.

[53] Oh MH, Iyengar G. Sequential anomaly detection using inverse reinforcement learning. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Anchorage: ACM, 2019. 1480–1490. [doi: 10.1145/3292500.3330932]

[54] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

[55] Ghazi MR, Gangodkar D. Hadoop, MapReduce and HDFS: A developers perspective. Procedia Computer Science, 2015, 48: 45–50. [doi: 10.1016/j.procs.2015.04.108]

[56] Oliner A, Stearley J. What supercomputers say: A study of five system logs. In: Proc. of the 37th annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. Edinburgh: IEEE, 2007. 575–584. [doi: 10.1109/DSN.2007.103]

[57] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: Toward an open-source solution for cloud computing. Int'l Journal of Computer Applications, 2012, 55(3): 38–42. [doi: 10.5120/8738-2991]

[58] Xu W, Huang L, Fox A, Patterson D, Jordan MI. Detecting large-scale system problems by mining console logs. In: Proc. of the 22nd ACM SIGOPS Symp. on Operating Systems Principles. ACM, 2009. 117–132. [doi: 10.1145/1629575.1629587]

[59] Meng WB, Liu Y, Zhu YC, Zhang SL, Pei D, Liu YQ, Chen YH, Zhang RZ, Tao SM, Sun P, Zhou R. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. 2019. 4739–4745. [doi: 10.24963/ijcai.2019/658]

附中文参考文献

[2] 尤勇, 汪浩, 任天, 顾胜晖, 孙佳林. 一种监控系统的链路跟踪型日志数据的存储设计. 软件学报, 2021, 32(5): 1302–1321. http://www.jos.org.cn/1000-9825/6234.htm [doi: 10.13328/j.cnki.jos.006234]

[3] 贾统, 李影, 吴中海. 基于日志数据的分布式软件系统故障诊断综述. 软件学报, 2020, 31(7): 1997–2018. http://www.jos.org.cn/1000-9825/6045.htm [doi: 10.13328/j.cnki.jos.006045]

[4] 包航宇, 殷康璘, 曹立, 李世宁, 孙永敬, 尹汇锋, 汤汝鸣, 侯岳, 王士强, 裴丹, 杨晓勤, 王立新. 智能运维的实践: 现状与标准化. 软件学报, 2023, 34(9): 4069–4095. http://www.jos.org.cn/1000-9825/6876.htm [doi: 10.13328/j.cnki.jos.006876]

[51] 刘全, 颜洁, 乌兰. 扩散模型期望最大化的离线强化学习方法. 软件学报, 2025, 36(10): 4695–4709. http://www.jos.org.cn/1000-9825/7296.htm [doi: 10.13328/j.cnki.jos.007296]

作者简介

周俊伟, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为计算机视觉, 系统安全.

胡淼, 硕士生, 主要研究领域为日志异常检测.

王春龙, 硕士生, 主要研究领域为日志异常检测.

杜亚娟, 博士, 副教授, CCF 专业会员, 主要研究领域为计算机存储系统, 软件安全性.

谈诚, 博士, 讲师, CCF 专业会员, 主要研究领域为攻击溯源, 云安全, 区块链安全, 智能合约安全.