计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(10): 2512−2526, 2026

面向物联网终端的云边协同日志异常检测技术

陈大江　安一韬　焦丽华　肖昱璇　邓伏虎
（电子科技大学信息与软件工程学院　成都　611731）
（djchen@uestc.edu.cn）

Cloud-Edge Collaborative Log Anomaly Detection Technology for IoT Terminal Devices

Chen Dajiang, An Yitao, Jiao Lihua, Xiao Yuxuan, and Deng Fuhu
（*School of Information and Software Engineering*, *University of Electronic Science and Technology of China*, *Chengdu* 611731）

Abstract　With the widespread deployment of IoT terminal devices in scenarios such as industrial control, intelligent transportation, smart homes, and smart healthcare, massive amounts of semi-structured logs are continuously generated at the edge. Existing methods either rely on log template IDs and shallow sequence models, making it difficult to model complex semantics, or rely on large-scale pre-trained models, which cannot meet the constraints of real-time performance and resource consumption of terminal devices. To address these issues, we propose a log anomaly detection technology for IoT terminal devices. First, a semantic enhancement detection model based on BERT and Llama is designed, employing regularization replacement and windowing modeling to suppress dynamic parameter noise. A semantic projection module is used to align the representation spaces of the encoder and decoder, and a three-stage fine-tuning method is used to improve the sequence-level anomaly detection capability. Second, a cloud-edge collaborative detection framework for terminal devices is constructed, proposing a two-stage confidence distillation and latency-sensitive dynamic routing strategy. This allows the small edge model to undertake most of the low-cost detection tasks, with only low-confidence samples uploaded to the large cloud model for verification. Our experiments on the public datasets HDFS, BGL, and Thunderbird-mini show that the proposed semantic detection model achieves $F1$ scores of 99.7%, 91.6%, and 96.6%, respectively. The cloud-edge collaborative framework maintains an overall $F1$ score of 95.5% while requiring only an average cloud routing ratio of 26.6%, which is equivalent to reducing the amount of large model calls by 73.4%.

Key words　IoT terminal devices; log anomaly detection; large language model; cloud-edge collaboration; knowledge distillation; dynamic routing

摘　要　随着物联网终端设备在工业控制、智能交通、智能家居和智慧医疗等场景中的广泛部署，边缘侧持续产生海量半结构化日志。现有方法要么依赖日志模板ID和浅层序列模型，难以建模复杂语义；要么依赖大规模预训练模型，难以满足终端设备对实时性与资源消耗的约束。针对上述问题，提出一种面向物联网终端设备的日志异常检测技术。首先，设计基于BERT与Llama的语义增强检测模型，采用正则替换与窗口化建模抑制动态参数噪声，通过语义投影模块实现编码器和解码器表示空间对齐，并利用三阶段微调提升序列级异常判别能力。其次，构建面向终端设备的云边协同检测框架，提出两阶段置信度蒸

收稿日期：2026-04-28；**修回日期**：2026-07-07
基金项目：国家重点研发计划（2023YFB3106402）；四川省杰出青年科技人才项目（2024NSFJQ0030）；国家自然科学基金项目（62471090）
This work was supported by the National Key Research and Development Program of China (2023YFB3106402), the Sichuan Science and Technology Program (2024NSFJQ0030), and the National Natural Science Foundation of China (62471090).
通信作者：邓伏虎（fuhu.deng@uestc.edu.cn）

馏与时延敏感动态路由策略，使边缘小模型承担大部分低成本检测任务，仅将低置信度样本上传云端大模型复核，在 HDFS、BGL 和 Thunderbird-mini 公开数据集上的实验表明，所提语义检测模型的 *F*1 分数分别达到 99.7%、91.6% 和 96.6%；云边协同框架在保持 95.5% 总体 *F*1 分数的同时，仅需平均 26.6% 的云端路由比例，相当于降低 73.4% 的大模型调用量。

关键词 物联网终端设备；日志异常检测；大语言模型；云边协同；知识蒸馏；动态路由

中图法分类号 TP393.08；TP309

DOI: 10.7544/issn1000-1239.202660259 **CSTR:** 32373.14.issn1000-1239.202660259

物联网终端设备已成为动态异构网络中的主要节点类型。与传统服务器日志相比，终端日志具有来源异构、格式不稳定、边缘资源受限和异常样本稀疏等特点，异常行为既可体现为跨日志上下文关系的偏移，也可体现为单条日志内部语义的异常变化[1]。因此，日志异常检测成为终端安全监测、故障诊断和攻击溯源的基础能力。早期工作主要通过控制台日志模式挖掘和统计建模识别异常[2-3]，但在日志模板快速演化和复杂语义场景下易出现泛化不足的问题。

近年来，深度学习方法开始主导日志异常检测研究。DeepLog、LogAnomaly 等[4-5]方法分别从模板预测和序列分布建模角度提升了检测性能，LogRobust[6]进一步通过语义表征提高了不稳定日志场景下的鲁棒性。然而，这类方法大多依赖日志解析器，将原始日志压缩为模板 ID 序列；一旦解析错误或出现词汇表外模板，异常信息就可能在预处理阶段丢失。Drain[7]在线解析方法虽能提高结构化效率，但仍难彻底解决语义缺失与解析漂移问题。

基于预训练的大语言模型为日志检测提供了新的建模思路。BERT 能够学习上下文相关语义表示[8]，Llama 等解码器型模型则具备更强的生成式判别能力[9]。然而，直接将长日志序列拼接后输入大模型会面临序列边界模糊、显存占用高和长序列推理效率低等问题[10]；同时，云端大模型很难直接部署在物联网终端设备上。面向终端侧应用，仅依赖模型压缩仍难同时兼顾检测精度、推理时延和通信成本[11]。

基于此，本文从“高精度语义检测”和“资源受限部署”2 个层面展开研究，构建面向物联网终端的云边协同日志异常检测技术。主要贡献包括：1）提出一种基于 BERT 与 Llama 协同建模的日志异常检测模型，通过正则归一化、语义投影和三阶段微调实现对原始日志序列的端到端异常判别；2）提出一种面向终端设备的云边协同检测框架，通过两阶段置信度蒸馏与时延敏感动态路由，在检测精度与系统开销之间实现动态平衡；3）设计并实现原型系统，给出公开数据集实验和系统测试结果，验证方法在实际应用部署场景中的工程可行性。

1 相关工作

日志异常检测（log anomaly detection, LAD）的目标是从系统运行过程中持续产生的大规模半结构化或非结构化日志中，自动识别偏离正常行为模式的异常事件。该方向已从早期依赖人工规则、阈值统计和浅层机器学习的方法，逐步发展到以深度学习、预训练语言模型和大语言模型为代表的语义建模方法。当前日志异常检测已经覆盖循环神经网络、卷积神经网络、Transformer、图神经网络等多类模型，但仍普遍存在预处理链路复杂、异常类型覆盖不充分、可解释性弱以及复现性不足等问题。除此之外，现有研究发现训练集划分、窗口构造、类别不平衡、数据噪声以及超参数设置都会显著影响模型结论，因此仅以单一数据集上的 *F*1 分数（*F*1-score）评价方法优劣并不充分，检测精度、时间性能与调参敏感性应被综合考量。

传统方法通常遵循“日志预处理—模板解析—特征向量化—异常判定”的技术路线。该类方法先利用 Drain[7]等日志解析器提取日志模板，再将模板序列输入主成分分析（principal component analysis, PCA）、支持向量机（support vector machine, SVM）、长短期记忆网络（long short-term memory, LSTM）等模型进行检测。以 DeepLog[4]、LogAnomaly[5]等为代表的方法推动了序列建模在日志异常检测中的应用，使模型能够学习日志事件的先后依赖关系与频次分布，从而较传统规则方法获得更好的自动化能力。然而，这类方法依赖日志解析器的质量，一旦出现词汇表外单词、模板漂移或变量提取错误，后续检测模型就会受到连锁影响。NeuralLog[12]的研究明确指出，日志解析错误主要来自集外词（out-of-vocabulary, OOV）

与语义误解，并证明绕开显式解析来直接建模原始日志语义能够有效提升检测鲁棒性。

为削弱对模板解析的依赖，研究者开始转向“parser-free”或“end-to-end”建模。LayerLog 从“词—日志—日志序列”3 级层次提取语义信息，在不依赖日志解析器的前提下同时建模词级、日志级和序列级语义关系；LAnoBERT 则采用基于 BERT 掩码语言模型的无监督框架，实现了解析器无关的异常检测；OneLog 进一步将解析、向量化和分类融合为单个端到端神经网络，通过字符级卷积直接处理原始日志文本，并在多项目联合训练下展现出较好的跨数据集泛化能力。因此，日志异常检测正在从“先结构化、后学习”的流水线范式，逐步转向直接从原始日志中学习语义表示的端到端范式。

随着预训练语言模型的进一步发展，日志异常检测的研究重点从“事件顺序”扩展到“日志文本语义”。LAnoBERT、LogFiT 等[13]工作利用 BERT 类模型对日志进行无监督或自监督建模，其中，LogFiT 通过微调预训练语言模型学习正常日志的语言模式，在无需异常标签的条件下实现检测，并在引入日志内容变化时表现出较好的鲁棒性[13]。与此同时，训练自由(training-free)的检索式方法也开始出现，相关研究将预训练语言模型与相似日志检索结合，在无需针对特定数据集训练的情况下获得具有竞争力的效果；LogOW[14]则面向开放世界场景，将不确定性估计与增量预训练结合，以缓解未见类别和冷启动问题。上述研究说明，预训练语言模型显著增强了日志语义表示能力，并推动检测方法从封闭世界、静态模板向开放世界、动态演化场景延伸。

在此基础上，日志异常检测开始进入大语言模型驱动阶段。一方面，有研究尝试利用 ChatGPT 承担日志解析任务，再结合 BERT 完成异常检测，结果表明基于大模型的语义解析能够为后续检测提供更有效的模板与语义信息；另一方面，近期工作进一步直接将大语言模型用于异常判别与解释生成，例如 LLM-LADE 通过“数据增强—参数高效微调—在线优化”的三阶段框架，同时完成异常检测与原因解释，并利用知识库机制提升动态环境适应能力。文献 [15] 也指出，在智能运维(artificial intelligence for IT operations, AIOps)场景下，提示工程与检索增强生成(retrieval-augmented generation, RAG)正成为提高日志异常检测准确性与可解释性的重要方向。尽管如此，大语言模型方法仍面临显著挑战：其一，模型参数规模大、训练和推理开销高；其二，日志数据高度不平衡，异常样本稀缺；其三，日志结构持续演化，单次离线微调难以长期适应真实运行环境。

对于物联网终端设备而言，日志异常检测不仅要追求检测精度，还必须满足低时延、低功耗和低通信开销等部署约束。围绕这一问题，已有研究开始探索轻量化与边缘部署方案。LightLog[16]面向边缘设备提出轻量级时序卷积网络，通过低维语义向量和轻量时间卷积网络(temporal convolutional network, TCN)结构兼顾检测性能与实时处理能力，并明确指出“高精度日志异常检测模型难以直接部署到边缘侧”是物联网(Internet of things, IoT)场景落地的关键瓶颈。与此类似，国内研究也开始关注隐私保护和分布式协同，有作者提出基于联邦学习的异常日志检测架构，在本地节点完成训练，仅上传模型参数聚合，以降低日志集中汇聚带来的隐私泄露风险。

LogHGAD[17]在 AIoT(artificial intelligence of things)场景下的日志特征显式纳入模型设计，通过超图建模同时融合日志头信息、参数信息及多元交互关系，在 5 个真实数据集上取得了优于 7 种基线方法的效果，并兼顾了实时安全场景下的效率需求。因此，面向物联网和 AIoT 场景的日志异常检测正在从单纯依赖模板或语义序列，进一步走向对日志上下文、实体关系和结构信息的联合建模。

现有研究已经在日志语义建模、无解析检测、预训练迁移以及边缘轻量化等方面取得了明显进展，但仍存在以下不足：大量方法仍默认日志先被稳定解析，难以充分适应物联网终端日志中常见的模板漂移、参数噪声和格式异构问题；大模型方法虽具备更强语义理解能力，但部署成本高，难以直接适配资源受限终端；轻量化方法虽然推理效率较高，但对复杂上下文异常和隐式语义异常的识别能力仍有限；现有协同研究更多面向传感器数据、网络流量或一般异常检测任务，真正面向“物联网终端设备日志”的云边协同检测研究仍然不足。因此，有必要围绕物联网终端日志的语义表达、轻量部署和协同推理机制开展进一步研究，以实现检测精度、实时性与工程可用性的统一。

2 预备知识

2.1 日志解析

日志解析(log parsing)技术作为日志挖掘与异常检测系统中的关键预处理步骤，通过结构化转换机制将原始非结构化日志文本映射为规范化日志模板

与参数序列，从而为后续的统计分析与深度学习建模提供结构化输入[18]。其核心思想在于将日志消息分解为日志模板（log template）与变量参数（parameters）形式，实现日志语义的压缩表达与模式抽象。

日志解析通常包括日志预处理、模板生成与参数提取 3 个核心模块。首先，预处理阶段对原始日志进行分词、去噪和格式标准化；其次，通过模板抽取算法识别日志中的固定模式；最后，将变化字段替换为通配符标记（如⟨∗⟩），形成结构化日志模板。在设原始日志集合为$L=\{l_1,l_2,\cdots,l_n\}$，日志解析的目标是构建映射函数$f:l_i\to(T_j,P_i)$，T_j表示日志模板，P_i表示对应参数集合。解析后的模板集合为$T=\{T_1,T_2,\cdots,T_m\},m\ll n$。通过模板抽象，日志序列可表示为模板 ID 序列，为后续序列建模提供离散结构化输入。日志解析方法多基于规则与正则表达式，通过人工定义模式实现字段匹配，但在面对复杂系统日志时可扩展性较差。随后，研究者提出基于数据驱动的自动化解析算法。例如，Drain 采用固定深度解析树（parse tree）结构，通过逐层匹配词元进行模板聚类，其核心思想是根据日志长度与前缀信息快速定位候选模板，从而实现在线高效解析。设日志被分词为序列$w_1,w_2,\cdots,w_k$，Drain 通过层级匹配函数

$$Sim(l_i,T_j)=\frac{\sum_{t=1}^{k}\prod(w_t=T_{j,t})}{k}。\tag{1}$$

式（1）计算日志与候选模板之间的相似度，当相似度高于阈值θ时，将其归入对应模板，否则生成新模板。该方法在保证解析准确率的同时显著提升了解析效率。

2.2 预训练大语言模型原理

预训练大语言模型[19]作为当前自然语言处理领域的重要技术范式，其核心思想在于通过大规模无监督语料进行统一建模，学习通用的语言表示能力，再通过针对具体任务的调优策略实现下游应用的高效适配。该方法遵循“预训练—微调”两阶段范式，在计算资源与数据规模的支撑下，实现了跨任务、跨领域的知识迁移。

从认知角度来看，人类在语言学习过程中通常先通过大量阅读与交流积累通用语言知识，再在具体情境中进行专项训练。预训练大语言模型正是借鉴了这一“先泛化、再专精”的学习机制。其第 1 阶段通过对海量文本数据进行自监督学习，使模型掌握语法结构、语义关系以及世界知识；第 2 阶段则根据具体任务目标，通过参数更新或结构适配，使模型的表达能力向目标任务空间收敛。

预训练大语言模型通常基于自回归或自编码目标函数进行训练。以自回归语言建模为例，给定一个长度为 n 的序列如式（2）所示。

$$X=(x_1,x_2,\cdots,x_n)。\tag{2}$$

模型通过最大化条件概率来学习参数 θ，其目标函数如式（3）所示。

$$L_{\mathrm{AR}}=-\sum_{t=1}^{n}\log P_\theta(x_t|x_{<t})，\tag{3}$$

其中$x_{<t}$表示位置 t 之前的所有词元，模型通过逐步预测下一个词来学习语言分布。这类方法被广泛应用于生成式模型，如 OpenAI 提出的 GPT-3。另一类常见的预训练方式是掩码语言建模（masked language modeling, MLM），其目标是在输入序列中随机遮蔽部分词元，并预测被遮蔽的内容。设掩码集合为 M，则目标函数如式（4）所示。

$$L_{\mathrm{MLM}}=-\sum_{i\in M}\log P_\theta(x_i|X_{\setminus M})。\tag{4}$$

该方法通过双向上下文建模提升语义理解能力，典型代表为 Google 提出的 BERT。预训练大语言模型通常基于多层 Transformer 堆叠结构，通过自注意力机制实现全局依赖建模。设隐藏层表示为 H^l，则每一层的更新过程如式（5）所示。

$$H^{l+1}=LayerNorm\left(H^l+FFN\left(Attention\left(H^l\right)\right)\right)，\tag{5}$$

其中 *Attention* 表示多头自注意力运算，*FFN* 表示前馈网络模块。通过多层非线性变换与残差连接，模型逐层抽象语义表示，从而形成高维语义空间中的知识表达。在完成预训练后，模型通常需要通过调优（fine-tuning）策略适配具体任务。传统全参数微调方法通过最小化下游任务损失函数来更新全部参数，损失计算如式（6）所示。

$$L_{\mathrm{task}}=-\sum_{i=1}^{N}y_i\log\hat{y}_i，\tag{6}$$

其中y_i为真实标签，$\hat{y}_i$为模型预测结果。该方法能够充分利用模型容量，但在参数规模达到数十亿甚至千亿级时，计算与存储成本较高。为提升效率，参数高效微调技术被广泛提出。例如低秩适配（LoRA）通过将权重更新分解为低秩矩阵形式。

提示学习通过在输入空间中引入可学习提示向量，将下游任务转化为语言建模问题，使模型无需大规模参数更新即可完成任务适配。进一步地，指令微调通过构建“指令—输入—输出”格式数据，使模型在统一框架下学习任务泛化能力。此类方法在

Meta 发布的 Llama 等模型中得到了广泛应用。

预训练大语言模型通过大规模自监督学习构建通用语言表示空间,再通过多种调优策略实现任务适配与行为对齐。这种分阶段训练范式不仅提升了模型的泛化能力,也显著降低了特定任务开发的成本,为自然语言处理系统提供了统一而高效的技术基础。

2.3 大小模型协同技术

大小模型协同(large–small model collaboration)通过构建“大模型-小模型”的分工协作机制,在保证预测性能的同时显著降低计算开销与响应延迟[20]。大模型具备强大的表示能力与泛化能力,但计算成本高;小模型结构轻量、响应迅速,但表达能力有限。通过合理设计路由机制与知识传递策略,使 2 类模型在不同难度样本上分工处理,从而实现性能与效率的平衡。

大小模型协同框架由 3 个核心模块构成:小模型(small model)、大模型(large language model)以及样本路由器(router)。其中,小模型负责处理大部分“易样本”,大模型负责处理“难样本”或不确定样本,路由器根据输入样本的特征或模型置信度决定样本的分配路径。整体推理流程可表示为式(7)。

$$y=\begin{cases} f_s(x), & \text{if } R(x)=0, \\ f_l(x), & \text{if } R(x)=1, \end{cases} \tag{7}$$

其中,$f_s(x)$表示小模型预测函数,$f_l(x)$表示大模型预测函数,$R(x)$为路由函数,其输出决定样本由哪个模型处理。在训练阶段,大小模型协同通常结合知识蒸馏(knowledge distillation)机制实现能力迁移。设大模型输出的软标签如式(8)所示。

$$p_i^{\mathrm{Te}}=\frac{\exp\left(\frac{z_i^{\mathrm{Te}}}{\tau}\right)}{\sum_j \exp\left(\frac{z_j^{\mathrm{Te}}}{\tau}\right)}, \tag{8}$$

其中,z_i^{Te}为大模型在类别 i 上的 logit 输出,τ为温度系数。小模型通过最小化与大模型输出分布之间的 KL 散度(Kullback-Leibler divergence)进行学习。

$$L_{\mathrm{KD}}=KL(p^{\mathrm{l}}\|p^{\mathrm{s}}), \tag{9}$$

其中,p^{s}为小模型输出分布,p^{l}为大模型输出分布。通过蒸馏训练,小模型能够在保持轻量结构的前提下吸收大模型的判别知识。在样本路由策略方面,常见方法包括基于置信度的动态路由与显式训练路由网络 2 种范式。基于置信度的方法通常利用小模型输出的最大概率或预测熵作为不确定性指标。当预测熵超过阈值δ时,将样本转交大模型处理。该策略实现简单,推理开销低,但阈值选择对系统性能影响较大。

随着应用场景复杂度提升,多级协同结构被进一步提出,即构建“多小一大”或“级联式”模型架构。在级联框架中,样本逐层经过不同容量的模型,当目前模型置信度不足时再进入更大规模模型,从而形成逐级筛选机制。这种结构能够有效降低平均计算开销,尤其适用于实时性要求较高的场景。此外,在生成式任务中,大小模型协同还可以采用推理辅助机制。例如小模型先生成候选结果,大模型进行重排序或验证;或由大模型提供策略指导,小模型负责高频推理执行。这类协同方式在大规模语言模型系统中具有广泛应用价值。

3 面向物联网终端的日志异常检测方法

3.1 系统模型

面向物联网终端的日志异常检测的核心思想是充分利用大语言模型在自然语言理解方面的优势,通过构建“编码—对齐—生成式判别”的统一框架实现对原始日志序列的语义建模与异常判别。与传统依赖日志解析器提取模板 ID 方法不同,本方案基于日志文本内容进行语义表示学习,引入编码器型大模型与解码器型大模型的协同机制,在嵌入空间中完成跨模型语义对齐,从而实现高精度的日志异常检测。

如图 1 所示,预处理阶段采用正则表达式对日志消息中的动态参数进行统一替换,保留稳定的语义结构,避免传统日志解析方法在不稳定日志场景下带来的语义丢失问题。随后,将处理后的日志消息按照会话窗口或滑动窗口策略构建为日志序列来作为模型输入。模型由 3 部分构成:语义编码模块、向量对齐模块以及序列判别模块。语义编码模块负责对单条日志消息进行上下文建模,生成其语义向量表示;向量对齐模块通过线性映射将编码得到的语义表示投影至生成判别模块的嵌入空间,实现 2 类模型表示空间的一致性;最终,将对齐后的日志序列嵌入向量与预设提示模板进行拼接,输入至生成判别模块,通过自回归方式输出对应的异常判定结果。在训练阶段,算法采用三阶段渐进式优化策略以增强模块间的协同能力。第 1 阶段仅对生成判别模块进行微调,使其掌握异常检测任务的回答范式,建立基础判别能力;第 2 阶段冻结生成判别模块,训练语义编码模块及向量对齐模块,使日志语义表示能够

有效匹配目标嵌入空间分布；第 3 阶段联合微调整体模型，进一步强化各模块之间的协同优化，从而提升整体检测性能与稳定性。同时，为缓解异常检测中的类别不平衡问题，算法引入少数类过采样策略，提高模型对异常样本的识别能力。在推理阶段，模型输入经过正则处理的日志序列，通过已训练完成模型进行前向传播，即可输出序列级别的异常判定结果。

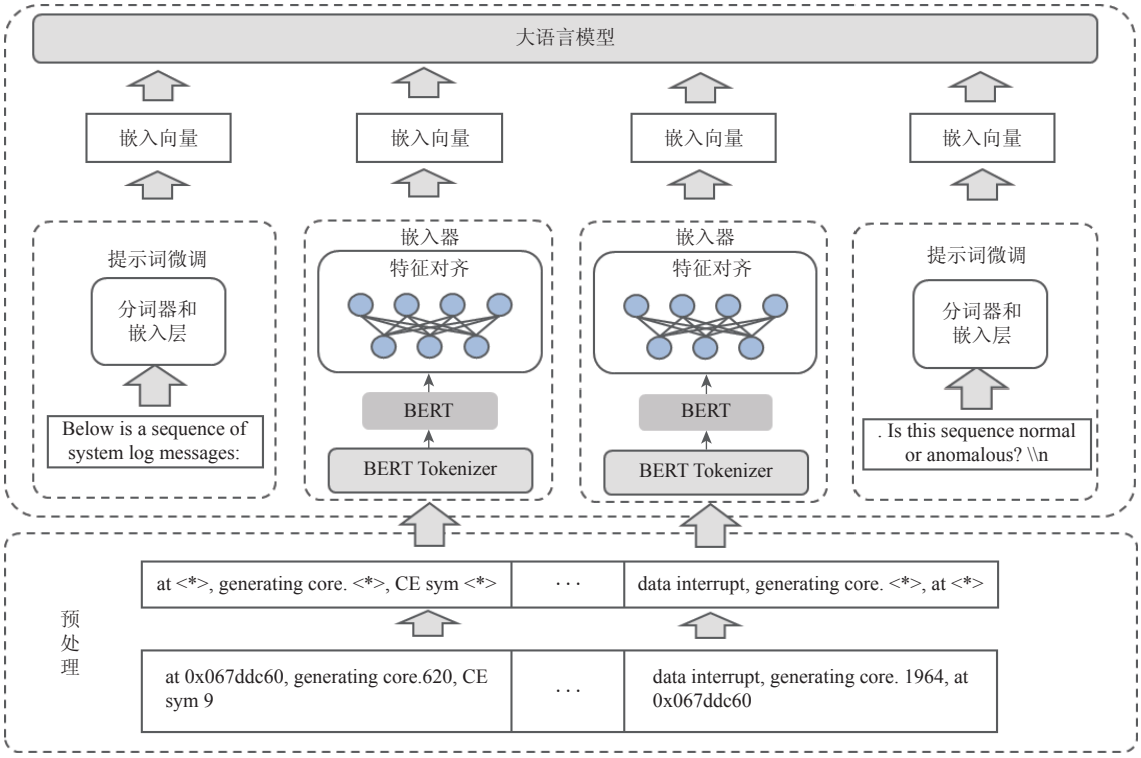


Fig. 1 Diagram of system model

图 1 系统模型图

3.2 基于大语言模型的语义增强检测

3.2.1 日志预处理

日志消息可以基于会话或固定/滑动窗口分组为日志序列。会话窗口分区根据其会话 ID 对日志消息进行分组，从而生成在每个会话内包含日志消息的序列。考虑到日志消息内容包括携带动态运行时信息的可变参数，这总是与异常无关，并使深度模型训练复杂化，需要一种技术来识别这些参数并用常量标记替换它们。常用的日志解析器，如 Drain 和 Spell，会自动将每条日志消息转换为特定的事件模板，方法是移除参数并保留关键字，在基于日志的异常检测方法中被广泛采用，并且似乎是一种有效的技术。

然而，最近的日志解析研究表明，现有的日志解析器会产生大量的解析错误，这直接降低了异常检测的性能。日志解析错误主要源于以下 2 个原因：1)在软件的演进过程中，日志语句可能频繁变化，导致出现训练数据中未包含的新日志事件，常用的日志解析方法难以处理新日志消息中的词汇表外(OOV)单词，从而导致语义信息丢失，当日志不稳定时，这些解析器随着时间的推移变得越来越无效；2)在将日志消息解析为日志事件的过程中，可能会丢失重要信息，从而导致对日志消息语义的误解。日志解析错误会影响后续的异常检测任务，并降低检测准确率。

由于结构化日志的特点，表示特定日志对象的参数的文本格式可以使用正则表达式轻松识别。因此，本文将每个变量参数，如账户名、目录路径和 IP 地址等可变参数替换为“<*>”。图 2 给出了本文处理预处理 BGL 数据集的流程，尽管它很简单，但这种方式提供了显著的性能优势。与使用日志解析器相比，这种预处理技术更有效，且不需要训练。

3.2.2 语义编码

在日志异常检测任务中，日志消息通常表现为离散的自然语言形式序列。现有方法大多依赖日志解析器提取模板 ID，或基于词袋模型构建统计特征，再通过浅层神经网络进行建模。然而，日志异常往往并非由单一关键词触发，而是由多个语义单元之间的组合关系所决定。例如，某些状态词与特定操

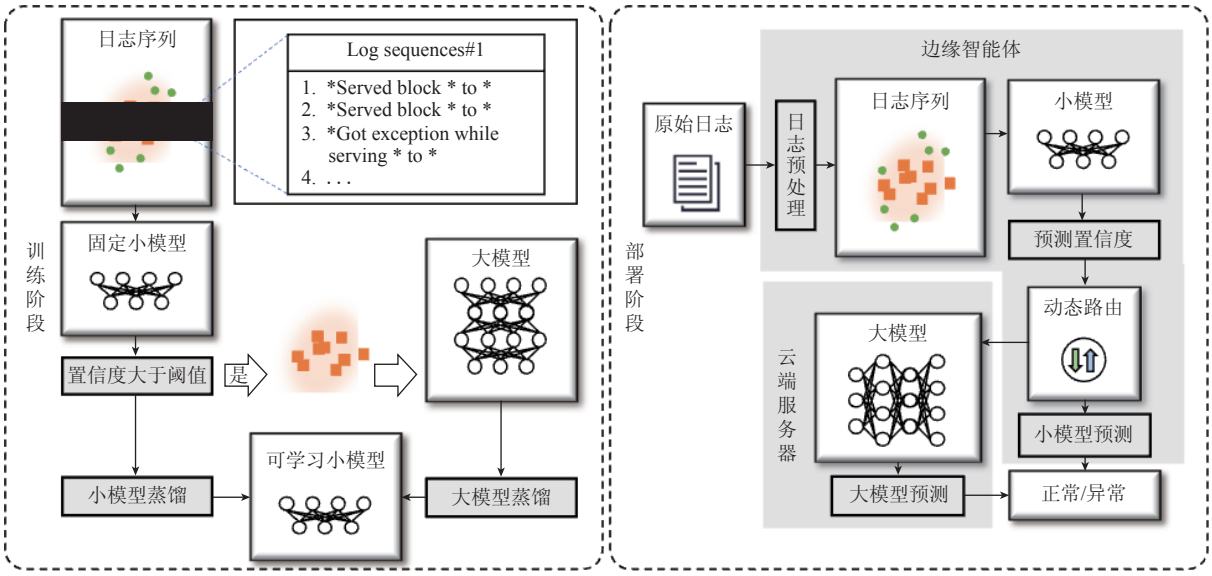


Fig. 2 Cloud-edge collaboration-based log anomaly detection framework

图 2 基于云边协同的日志异常检测框架

作对象共同出现时才构成异常。这些具有明显语义依赖的异常信息对于那些难以捕捉这些语义信息的方法，在面对日志模板不稳定或出现新模板时表现不佳。同时，直接使用大语言模型处理原始日志序列面临内存溢出和语义边界模糊等问题。因此，本文的目标是构建一种能够建模全局语义关系的编码结构，能够明确日志消息之间的边界，使日志文本在连续嵌入空间中形成具有判别性的语义表示。

BERT 模型利用 Transform 的编码器部分，设计多层双向变压器编码器，在保持并行计算优势的同时，采用预训练的方式，从而对日志文本的语义学有了更深刻的理解，利用多层自注意机制替代循环神经网络(recurrent neural network, RNN)和卷积神经网络(convolutional neural network, CNN)，关注输入日志文本中不同部分词之间的相关性，并对该日志中每个词的权重进行调整，进一步获得单个日志更高维度的含义，更好地表征日志文本中的句法语义信息，从而可以获得更好的日志词向量表示。

通过在大规模通用语料集上预训练，BERT 具备良好的自然语言理解能力，适用于日志这种半结构化文本。相较于解码器模型，BERT 参数量较小，适合作为前置编码模块，不会显著增加整体计算开销。对于任意日志 $L_i \in L$，首先使用 BERT 分词器对其进行分词(tokenizer)，如式(10)所示。

$$Tokens_i = Tokenizer(L_i)\ , \tag{10}$$

其中，$Tokens_i$ 可以表示为式(11)。

$$Tokens_i = \left([\mathrm{CLS}], t_1, t_2, \cdots, t_{L_i}, [\mathrm{SEP}]\right)\ 。 \tag{11}$$

然后，将每条日志的 $Tokens_i$ 输入 BERT 模型，以生成语义向量 $\boldsymbol{H}_i = BERT(Tokens_i) \in \mathbb{R}^{d_{\mathrm{BERT}}}$。其中，$\boldsymbol{H}_i$ 表示 BERT 输出的隐状态序列，d_{BERT} 表示 BERT 的隐藏层维度。从 $\boldsymbol{H}_i$ 中获取 [CLS] 的隐藏层表示，并使用 tanh 激活函数生成语义向量 $\boldsymbol{h}_i = \boldsymbol{H}_i[0]$。由于日志语义编码层和基于大模型的表示空间(维度、分布等)存在差异，直接将 BERT 输出的语义向量作为后续输入可能导致语义偏移。为此，本文引入一个可训练的投影层，将日志语义编码层的表示空间映射到后续大模型的表示空间。投影层是一个线性层，它将从 BERT 处获取的语义向量映射到 Llama 接受的 token 嵌入向量维度，表示为式(12)所示。

$$\boldsymbol{E} = Projection(C) \in \mathbb{R}^{N \times d_{\mathrm{Llama}}}\ , \tag{12}$$

其中 $\boldsymbol{E}$ 表示向量映射的结果，d_{Llama} 是 Llama 的隐藏层维度。投影网络旨在对齐 BERT 和 Llama 的向量表示空间。

3.2.3 异常模式学习

在获得日志消息的语义向量表示后，如何有效利用这些向量进行序列级别的异常检测成为关键问题。传统的序列模型(如 LSTM 等)在一定程度上建模时序依赖关系，但在理解复杂语义模式和捕捉长距离依赖方面存在固有局限。大语言模型，特别是基于 Transformer 解码器架构的模型，凭借其强大的上下文建模能力和丰富的预训练知识，能够更好地捕捉日志序列中的深层异常模式。基于此，本文以 Llama 作为核心模型设计出一个基于大模型的日志序列异常检测模块，通过精心设计的提示词引导模

型理解任务目标，并采用生成式分类策略输出异常检测结果。

本方案根据语义嵌入后的日志序列生成结构化的文本查询，每个查询由 3 个模块组成。第 1 个模块为任务描述模块，使用提示词“Below is a sequence of system log messages:”引导模型理解接下来将输入日志序列。第 2 个模块是由投影网络输出的语义嵌入 $\boldsymbol{E}$。第 3 个模块为查询模块，使用提示词“Is this sequence normal or anomalous?”明确询问序列是否异常。第 1 个和第 3 个模块首先经过 Llama tokenizer 和 Llama embedding 层，得到对应的嵌入表示 $\boldsymbol{E}_1 \in \mathbb{R}^{A\times d_{\text{Llama}}}$ 和 $\boldsymbol{E}_3 \in \mathbb{R}^{Q\times d_{\text{Llama}}}$，其中 A 和 Q 分别为 2 个模块经 tokenizer 后生成的 token 数量。然后，将 3 个模块的嵌入表示按顺序连接，构成 Llama 的完整输入序列，表示为式(13)所示。

$$\boldsymbol{E}_{\text{total}} = [\boldsymbol{E}_1][\boldsymbol{E}_2][\boldsymbol{E}_3]\ 。 \tag{13}$$

该完整嵌入序列作为 Llama 的输入，参与后续的序列建模与异常分类。

在输出端，为使 Llama 能够以统一的格式生成分类结果，本文定义标准化的输出模板。模型被训练为根据输入序列生成“The sequence is anomalous.”或“The sequence is normal.”形式的回答，分别对应异常样本和正常样本。这种生成式输出方式不仅使结果自然可解释，便于人工验证，同时也充分发挥了 Llama 作为生成式模型的优势。在序列建模过程中，Llama 通过多层 Transformer 解码器对输入序列进行处理，每一层执行自注意力机制(配合因果掩码保持自回归特性)、前馈网络变换、层归一化与残差连接等操作，最终以自回归方式生成输出序列。训练过程中，模型以最大化生成正确输出序列的概率为目标，使用交叉熵损失函数计算生成文本与目标模板之间的差异。

为降低大模型微调的计算成本，本文采用量化低秩适配(quantized low-rank adaptation, QLoRA)技术进行高效微调。具体而言，将 Llama 的权重量化到 4-bit NF4 格式以大幅减少显存占用，同时在量化模型的基础上插入可训练的 LoRA，仅更新少量参数即可实现与全精度微调相当的性能。通过该技术，本文能够在有限显存内微调 Llama-3-8B 模型，同时保持优异的检测性能。综上所述，该模块通过 Llama 与 QLoRA 的结合，以及精心设计的提示模板和输出格式，实现了高效、准确、可解释的日志序列异常检测，为整体框架提供了强大的序列建模能力。

3.2.4 三阶段微调

为有效训练本文提出的深度模型，实现各模块的协同优化与最佳性能，本文设计了三阶段训练算法。该算法通过渐进式参数更新策略，依次优化模型的不同组成部分，最终实现整体模型的端到端的调优。

1)微调 Llama 以生成答案模板阶段。这阶段的目标是使 Llama 模型掌握预期的输出格式，能够根据输入提示生成标准化的答案模板。本文训练 Llama 响应任务提示，即给定查询“Is this sequence normal or anomalous?”，模型应生成“The sequence is anomalous.”或“The sequence is normal.”形式的回答。在该阶段中，冻结 BERT 编码器和投影网络的参数，仅更新 Llama 模型的参数。这种策略将训练聚焦于让 Llama 学习输出模板的语法结构和语义表达，而不受日志语义编码模块的干扰。由于答案模板的格式相对固定且语义简单，本阶段仅需少量数据样本即可完成有效微调，大幅降低了初期训练成本。

2)训练日志消息编码器阶段。这阶段的目标是训练日志消息编码器，即 BERT 模型和投影网络，使其能够将每条日志消息映射到 Llama 表示空间中最合适的嵌入向量，从而为 Llama 提供富含语义且易于辨别的输入表示。这个阶段分 2 个步骤：首先冻结 BERT 编码器和 Llama 模型的参数，仅训练投影网络。此步骤的目标是让投影网络学习将 BERT 输出的语义向量空间映射到 Llama 的嵌入空间，实现 2 个异构表示空间的对齐。通过最小化投影后的向量与 Llama 理想输入表示之间的差异，投影网络逐步掌握跨空间映射能力。在投影网络初步对齐的基础上，解冻 BERT 编码器的参数，允许其参与训练。此步骤训练整个日志消息编码流程，即从原始日志输入的 BERT 编码到投影网络的跨空间映射，最终到 Llama 的序列建模与异常分类。通过端到端的梯度回传，BERT 模型得以根据下游异常检测任务的反馈，调整其语义提取策略，生成更有利于 Llama 辨别的日志表示。

3)全模型联合微调阶段。此阶段的目标是实现所有模块的协同优化，确保整体模型的内聚性与准确性能。在此阶段，解冻所有模块的参数，对整个模型进行端到端的联合微调。

3.3 基于云边协同的日志异常检测算法

3.3.1 算法总览

本文提出的云边协同日志异常检测算法核心思想是在保证整体检测精度的前提下，通过两阶段置

信度引导蒸馏提升小模型判别能力,并结合代价感知路由机制,实现精度与时延之间的动态权衡。算法整体由“蒸馏增强模块”与“代价感知路由模块”2部分构成,分别对应训练阶段与部署阶段的优化目标。方案整体由“蒸馏增强模块”与“代价感知路由模块”2部分构成,分别对应训练阶段与部署阶段。

基于云边协同的日志异常检测框架如图2所示。具体而言,在训练阶段,算法构建由大模型、固定小模型与可学习小模型组成的协同蒸馏框架。大模型作为教师模型提供高精度预测分布;固定小模型为仅基于真实标签训练得到的稳定模型,不参与教师知识更新;可学习小模型用于吸收多源知识并逐步扩展判别能力。对于输入日志样本x,首先计算固定小模型预测分布与置信度,根据置信度信息构建分阶段蒸馏机制。当固定小模型对样本预测不确定而大模型置信度较高时,执行从大模型到可学习小模型的知识蒸馏,通过最小化两者预测分布之间的KL散度实现决策边界扩展;当特定小模型对样本预测置信度较高时,则执行特定小模型到可学习小模型的自蒸馏,以保持其在已良好拟合区域的稳定性,避免教师模型潜在噪声对学生模型产生干扰。

在训练过程中,2类蒸馏损失交替优化,使可学习小模型在吸收大模型知识的同时保持原有判别分布的稳定性,从而缓解灾难性遗忘问题。经过迭代训练后,得到性能增强的小模型,其对复杂日志模式的建模能力显著提升,能够独立处理更多困难样本,以降低后续部署阶段对大模型的依赖。在部署阶段,算法仅保留优化后的小模型与代价感知路由模块。对于每一个输入日志样本,首先由小模型输出预测分布与置信度信息。随后,路由模块基于推理时延代价与精度损失代价构建综合评分函数,对“本地小模型处理”与“云端大模型处理”2种决策进行代价评估。若小模型预测带来的综合代价低于预设阈值,则直接输出小模型结果;否则将样本路由至大模型进行精细分析。其中,推理时延代价通过小模型与大模型平均推理时间差进行建模,精度损失代价则通过小模型置信度进行估计。通过在统一评分框架下进行权衡,系统能够根据实时需求动态调整精度与时延之间的平衡关系。具体过程如算法1所示。

3.3.2 置信度蒸馏

在云边协同日志异常检测框架中,小模型承担着边缘侧的实时检测任务,而大模型则提供高精度的深度分析能力。此前的协同方案大多采用基于置信度或不确定性的路由策略:当小模型预测置信度较高时直接输出结果,当置信度较低时将样本上传至云端大模型处理。这类机制能够在一定程度上平衡检测精度与系统开销。然而,当小模型表达能力有限时,大量“困难样本”会被频繁路由至云端,导致通信开销上升与时延增加,从而削弱云边协同架构的优势。因此,单纯依赖路由机制难以从根本上降低系统成本,必须进一步提升小模型自身的判别能力。通过知识蒸馏将大模型的判别知识迁移至小模型,可增强小模型对复杂日志模式的理解能力,使其能够独立处理更多样本,从而降低路由比例并提升整体检测精度与系统效率。

知识蒸馏的基本思想是通过最小化教师模型与学生模型输出分布之间的差异,使学生模型学习教师模型中蕴含的“软知识”。与仅利用真实标签进行监督不同,蒸馏利用教师模型输出的概率分布刻画类别之间的相对关系,从而传递更丰富的结构信息。通常采用KL散度来度量2个分布之间的差异,通过优化式(9)所示目标函数实现知识迁移。

通过这一机制,学生模型能够在保持较小参数规模的同时逼近大模型的决策边界,从而提升检测精度。

算法1. 云边协同自适应日志异常检测算法。

输入:日志数据集$D=\{(x_i,y_i)\}_{i=1}^{N}$,系统状态信息$L_{\text{net}}$(网络时延),云端时延$L_{\text{cloud}}$,真实标签,$\delta$,$T$,$\eta$,$\lambda$,$F_1$,$F_{\text{s1}}$;

输出:训练好的增强小模型F_{s2e},样本异常判定结果$\hat{y}$,路由决策$a\in\{0:\text{边缘处理},1:\text{云端处理}\}$。

① 从F_{s1}复制得到F_{s2};

② for x_i in Dx

③ 计算F_{s1}的预测置信度$C_{\text{s1}}(x_1)$;

④ 计算F_{s2}的$logits_{\text{learn}}$;

⑤ if $C_{\text{s1}}(x_1)<\delta$ then

⑥ 转向大模型学习

$$loss_{\text{kd}}=KLDivergence\left(probs_{\text{large}},Softmax\left(\frac{logits_{\text{learn}}}{T}\right)\right);$$

⑦ else

⑧ 固定小模型很确信,进行自蒸馏

$$loss_{\text{kd}}=KLDivergence\left(probs_{\text{fixed}},Softmax\left(\frac{logits_{\text{learn}}}{T}\right)\right);$$

⑨ 反向传播更新可学习小模型$Backward(loss)$,$Update(F_{\text{s2}},lr=\eta)$;

⑩ end if

⑪ end for

⑫ 使用 F_{s2e}进行推理 $conf_{small} = Max(probs_{small})$, $y_{small} = \mathrm{ArgMax}(probs_{small})$;

⑬ 计算动态阈值: $\delta(t) = 1 - \dfrac{L_{net}(t) + L_{cloud}(t)}{\lambda}$;

⑭ if $\delta(t) > \delta$ then

⑮ $\delta(t) = \delta$;

⑯ else if $\delta(t) < 0.5$ then

⑰ $\delta(t) = 0.5$;

⑱ end if

⑲ if $C_s(x) < \delta(t)$ then $a = 1$(路由至云端);

⑳ else $a = 0$(边缘输出)。

㉑ end if

然而, 在实际应用中可以观察到, 学生模型在进行知识蒸馏后往往会出现“灾难性遗忘”现象, 即在学习教师模型知识的过程中逐渐偏离原有数据分布上已良好拟合的区域。当教师模型预测存在偏差或置信度不稳定时, 学生模型可能被迫学习错误或噪声知识, 导致原有优势被削弱。此外, 在异常检测这一高度不平衡且边界敏感的任务中, 学生模型原本对部分简单样本具有较好判别能力, 若过度依赖教师模型进行统一蒸馏, 可能破坏其在特定分布上的稳定性。因此, 如何在吸收教师模型知识的同时保留学生模型原有优势, 成为蒸馏过程中的关键问题。

算法 1 是一种两阶段置信度引导的蒸馏方案。该方法基于教师模型与学生模型的置信度信息进行分阶段知识迁移: 当学生模型对样本预测不确定而教师模型置信度较高时, 引导学生模型从教师模型学习以扩展决策边界; 而当学生模型对样本具有较高置信度时, 则强化其原有判别分布, 以保持稳定性并避免错误知识干扰。具体而言, 为保留学生模型原有优势, 需要维护一个不接受教师模型知识更新的原始学生模型版本, 记为特定学生模型 F_{s1}。同时, 构建一个可学习学生模型 F_{s2}, 用于进行知识吸收与能力增强。为了减轻不正确知识对学生模型的负面影响, 可以使可学习学生模型能够同时基于它们各自的置信度从特定学生模型和教师模型中学习。对于一个输入样本, 当特定学生模型的置信度较低而教师模型的置信度较高时, 可学习学生模型将从教师模型中获取预测, 以引入额外的知识并促进成本降低。这样的过程使可学习学生模型能够处理日益多样化的样本。相反, 可学习学生模型将继续从特定学生模型中学习高置信度样本, 以减轻教师模型扭曲知识的影响。

对于给定输入样本 x, 通过对特定学生模型 F_{s1} 的输出进行 Softmax 运算得到预测置信度, 如式(14)所示。

$$C_{s1} = \frac{e^{z_i}}{\sum e^{z_d}},\ z_i \in F_{s1}(x)\ , \tag{14}$$

其中 z_i 表示特定小模型在第 i 类上的输出 logit 值, C_{s1} 表示对应类别的预测概率。某一维度上的第 i 类的预测概率越大, 表示模型对该类别判断的置信程度越高。

当 $C_T < \delta$ 时, 认为学生模型对该样本不确定, 此时计算教师模型 F_T 的预测置信度, 如式(15)所示。

$$C_T = \frac{e^{z_i}}{\sum e^{z_d}},\ z_i \epsilon F_T(x)\ , \tag{15}$$

其中 F_T 是大模型的函数, C_T 是预测置信度。如果 $C_T > \delta$, 用 KL 散度执行从教师模型到学生模型的蒸馏过程, 如式(16)所示。

$$L_{TS} = KL(F_{s2}(x), C_T)\ 。 \tag{16}$$

相反, 当 $C_T < \delta$ 时, 认为小模型对该样本具有较高置信度, 为避免大模型潜在噪声影响, 则执行特定小模型对可学习小模型的自蒸馏, 如式(17)所示。

$$L_{s1s2} = KL(F_{s2}(x), C_{s1})\ 。 \tag{17}$$

在训练过程中, 交替迭代执行上述 2 类蒸馏损失, 使可学习学生模型在吸收教师模型知识的同时保持原有判别分布的稳定性。通过这种两阶段置信度引导机制, 学生模型能够逐步扩展对复杂日志模式的建模能力, 减少对云端大模型的依赖, 从而在保证检测精度的同时有效降低路由比例与系统开销。

3.3.3 动态路由设计

在完成两阶段置信度蒸馏后, 就获得性能增强的小模型与高精度大模型。为在部署阶段实现云边协同系统的高效运行, 需要设计一个时延敏感的路由机制, 在保证检测精度的同时最小化系统推理时延。本节从优化目标出发, 对路由问题进行形式化建模, 并提出一种基于期望代价最小化的时延敏感路由算法。

对于输入日志样本 x, 系统需要在式(18)的 2 种处理方式之间进行选择。本节的目标是最小化样本的期望系统代价。令 L_{edge} 表示小模型推理时延, $C_s(x)$ 表示学生模型对样本 x 的预测置信度。学生模型输出通过 Softmax 计算得到预测置信度如式(18)所示。

$$C_{s1}=\frac{e^{z_i}}{\sum e^{z_d}},\ z_i\in F_{s1}(x)\ , \tag{18}$$

其中，$F_s(x)$表示学生模型输出的 logits，z_i表示预测类别对应的 logit。由于部署阶段真实标签未知，本文使用置信度估计错误概率，见式(19)。

$$P(error|x)\approx 1-C_s(x)\ 。 \tag{19}$$

设错误带来的风险权重为λ，表示单位错误概率对应的系统损失，则边缘处理的总代价表示如式(20)所示。

$$Cost_{edge}(x)=L_{edge}+\lambda(1-C_s(x))\ 。 \tag{20}$$

式(20)体现了精度风险与推理时延之间的权衡关系。

令L_{edge}表示边缘预处理时间，L_{net}表示网络传输时延，L_{cloud}表示云端推理时延，则云端总代价$Cost_{cloud}(x)$可以表示为式(21)。

$$Cost_{cloud}(x)=L_{edge}+L_{net}+L_{cloud}\ 。 \tag{21}$$

系统应该选择代价更小的处理方式。当$Cost_{edge}(x)>Cost_{cloud}(x)$时选择云端，否则直接在边缘进行处理。带入式(20)(21)得到式(22)。

$$L_{edge}+\lambda(1-C_s(x))>L_{edge}+L_{net}+L_{cloud}\ 。 \tag{22}$$

整理得到最终决策如式(23)所示。

$$C_s(x)<1-\frac{L_{net}+L_{cloud}}{\lambda}\ 。 \tag{23}$$

可以定义时延敏感阈值$\delta(t)$表示如式(24)所示。

$$\delta(t)=\delta_0-\frac{L_{net}+L_{cloud}}{\lambda}\ , \tag{24}$$

其中，δ_0表示系统设置的初始分流置信度阈值。该阈值直观上受网络状况和云端处理繁忙程度影响，当网络拥塞或云端处理资源紧张时，系统减少路由；当网络通常或云端资源空闲时，运行更多样本给云端处理。

4 性能与评价

4.1 数据集与实验设置

实验选取HDFS、BGL(Blue Gene/L)和Thunderbird-mini 三个公开日志数据集，如表 1 所示。其中，HDFS 数据集通过在 200 多个 Amazon EC2 节点上运行基于 Hadoop 的 MapReduce 作业生成，共包含 11 172 157 条日志消息。这些日志消息根据其 block_id 分组到不同的日志窗口中，这些日志窗口反映了 HDFS 中的程序执行情况，产生 575 061 个 block。其中，16 838 个 block(2.93%)表示系统异常。BGL 数据集[21]是从劳伦斯利弗莫尔国家实验室(LLNL)的 BlueGene/L 超级计算机系统收集的超级计算系统日志数据集。该数据集包含 4 747 963 条日志消息，每条消息都被手动标记为正常或异常，其中有 348 460 条日志消息(7.34%)被标记为异常。Thunderbird-mini 数据集[21]是来自桑迪亚国家实验室(SNL)Thunderbird 超级计算机的日志数据的公开访问集合。该数据集由正常和异常消息组成，每个消息都经过手动分类。为提升计算效率，本文从超过 2 亿日志消息中选取 2 000 万连续日志消息子集。该子集包括 4 937 条异常日志消息，约占总数的 0.049%。针对不同数据集，按照会话窗口或滑动窗口构造日志序列，并采用训练集/测试集划分方式进行评估。对于大模型检测实验，使用 BERT-base 与 Llama-3-8B 作为主干模型；对于云边协同实验，边缘小模型采用 BERT+LSTM，云端大模型采用 BERT+Llama。为降低类别不平衡影响，对少数类样本执行过采样。实验中使用到的软硬件环境配置如表 2 所示。

Table 1 Evaluation Dataset Statistics

表 1 评估数据集统计

数据集	日志条目数	日志序列数
HDFS	11 172 157	575 061
BGL	4 747 963	47 135
Thunderbird-mini	20 000 000	99 997

Table 2 Hardware and Software Environment

表 2 软硬件环境

软件/硬件	配置
操作系统	Ubuntu Linux（version 20.04）
训练框架	PyTorch（version 2.2.1）
Thunderbird-mini CPU	AMD 7950X
GPU	NVIDIA GeForce RTX 4090
Cuda	cuda_12.8

4.2 预处理技术评估

为全面评估不同预处理技术的有效性，本节针对当前日志异常检测领域常用的几种预处理方案开展了对比实验，实验结果详见表 3。在表 3 中，“Raw”代表日志消息内容未经过任何预处理，直接输入至所提出的模型框架；“Template”表示利用日志解析器 Drain 生成的日志模板序列作为所提模型框架的输入；“Template ID”则指通过嵌入层(而非 BERT)将 Drain 获取的日志模板 ID 简单编码为数字向量。显然，仅采用“Template ID”的预处理技术，使得模型无法有

Table 3 Impact of Different Data Preprocessing Schemes on the Performance of Model Framework

表 3 不同数据预处理方案对模型框架的性能影响

方法	HDFS			BGL		
	P	*R*	*F*1	*P*	*R*	*F*1
Raw	0.994	0.991	0.993	0.943	0.767	0.846
Template ID	0.995	0.945	0.969	0.775	0.286	0.418
Template	0.991	1.000	0.995	0.861	0.919	0.889
本文	0.994	1.000	0.997	0.861	0.979	0.916

注：*P* 表示精确率，*R* 表示召回率。

效捕获日志消息中的语义信息。需特别说明的是，Drain 解析器应用于整个数据集，而非仅针对训练集，以此避免因未词汇表外问题导致的性能下降。本文采用正则表达式进行预处理。

如表 3 所示，本文提出的的预处理方案在所有数据集中产生最高的 *F*1 分数。与之相对应的是，仅仅使用“Template ID”的预处理技术在所有数据集中始终取得最低的 *F*1 分数，比本文方法评价要低 0.368。这可以归因于“Template ID”直接使用嵌入层生成模板 ID 的向量表示，而不是去捕获日志消息中的语义信息，最终导致了其从自然语言的角度检测异常的能力。另外 2 个预处理技术“Raw”和“Template”性能相对较好，但总体来看，它们的 *F*1 分数仍然比本文方案低 6.4 个百分点和 5.7 个百分点。对于直接使用原始日志的方案，看似可以最大限度地保留日志消息中的全部语义信息，但它忽略了一个关键事实：每个日志消息内容中的可变部分(即机器运行时产生的参数)对于日志异常检测几乎没有影响。然而，由于它们的高随机性，它们会混淆模型，使得模型注意力分散，最终导致难以识别异常。对于使用 Drain 解析器解析的模板的预处理技术，解析器并不总是可靠，有时会错误地将常量解析为变量或将变量解析为常量，这会导致模型信息丢失或出现混淆，最终导致识别异常困难。

4.3 语义增强检测模型效果

表 4 呈现了所提出的语义增强检测模型与典型的日志异常检测方法在 3 个数据集上的 *F*1 分数对比情况。从表中数据可以清晰看出，本文方法在 HDFS、BGL 和 Thunderbird-mini 数据集上分别取得了 99.7%、91.6% 和 96.6% 的优异成绩，其平均 *F*1 分数高达 95.9%，在整体性能上显著优于 DeepLog、LogAnomaly、PLELog、FastLogAD、LogBERT、LogRobust、CNN、NeuralLog 以及 RAPID 等基线方法。特别地，在 BGL 和 Thunderbird-mini 这类数据结构更为复杂、异常模式更为丰富的数据集上，本文方法借助 BERT 语义编码与 Llama 生成式分类的协同作用，有效缓解了因依赖模板 ID 而导致的性能损耗问题，展现出强大的性能优势。

进一步的消融实验表明，BERT 语义编码与投影层显著降低了直接拼接原始日志输入大模型带来的显存压力，并通过显式保持日志边界提升了异常模式学习能力；三阶段训练流程则保证了解码器模板学习、嵌入空间对齐与端到端优化之间的衔接，是模型取得较高检测精度的重要原因。

Table 4 Compared with Representative Baselines for *F*1 Scores

表 4 与代表性基线方法的 *F*1 分数对比 %

方法	HDFS	BGL	Thunderbird-mini	平均 *F*1
DeepLog	90.8	28.5	3.3	50.6
LogAnomaly	96.6	29.9	5.0	52.1
PLELog	93.4	71.0	76.4	81.0
FastLogAD	79.8	28.7	1.7	34.1
LogBERT	75.8	28.3	3.9	45.6
LogRobust	98.0	81.0	48.2	77.1
CNN	98.2	81.0	76.9	81.8
NeuralLog	97.9	83.5	85.7	89.3
RAPID	92.4	54.8	20.3	60.2
本文	99.7	91.6	96.6	95.9

4.4 大语言模型的影响

本节选取 4 种参数规模相近但类型各异的大语言模型展开对比研究，包括 Qwen2.5-7B、Llama-3.1-8B、Mistral-7B 以及 GLM-4-9B 模型，旨在深入探究使用不同大型语言模型对检测性能的影响。同时，针对具有相同架构的 Llama 系列模型，进一步考察了其不同参数规模对日志异常检测结果的潜在影响。表 5 详细展示了这些模型在 2 个日志数据集上的性能表现。实验结果表明，Llama-3.1-8B 模型在 2 个数据集上均展现出了卓越的性能，分别取得了 0.997 和 0.916 的 *F*1 分数，在所有参与测试的模型中表现最优。Qwen2.5-7B 模型的性能紧随其后，在 HDFS 数据集上的表现与 Llama-3.1-8B 模型几乎不相上下，但在 BGL 数据集上却出现了性能下滑的现象。原因是由于 Qwen2.5-7B 模型在判别边界的确定上不够精准、果断，从而影响了其在该数据集上的性能发挥。实验还发现，GLM-4-9B 模型的输出极不稳定，常常输出与预设标签不一致的结果，这导致其在当前分类架构上性能受到影响。Mistral-7B 模型在 2 个数据集

上的表现相对逊色,仅分别取得了 0.969 和 0.850 的 $F1$ 分数。其原因是该模型在指令对齐方面的能力较弱,对日志这种半结构化文本的适应性较差,进而导致其检测性能不佳。此外,相较于小型模型 Llama-3.2-1B,采用参数量更大的 Llama-3.1-8B 模型在多个数据集上的 $F1$ 分数均有提升。这一显著提升充分彰显了大规模模型在语义理解能力和泛化能力方面所具备的巨大优势,为后续日志异常检测模型的选择与优化提供了重要的参考依据。

Table 5 Impact of Different LLMs on the Performance of Model Framework

表 5 不同 LLM 对模型框架的性能影响

大语言模型	HDFS			BGL		
	P	R	$F1$	P	R	$F1$
Qwen2.5-7B	0.982	0.993	0.987	0.835	0.961	0.894
Mistral-7B	0.962	0.976	0.969	0.787	0.924	0.850
GLM-4-9B	0.979	0.987	0.983			
Llama-3.2-1B	0.996	0.996	0.996	0.734	0.734	0.734
Llama-3.1-8B	0.994	1.000	0.997	0.861	0.979	0.916

注:P 表示精确率,R 表示召回率。

4.5 云边协同框架效果

为验证本文所引入的大小模型协同框架在提升整体模型精度、削减整体计算开销方面的有效性,本节于 HDFS、BGL 和 Thunderbird -mini 三个公开数据集上开展了实验。该实验对每个数据集进行了统一预处理,利用过采样技术确保正负样本数量达到平衡状态。将数据划分为训练集、验证集和测试集,分别用于模型训练、参数调优以及最终的模型性能评估。本实验中,小模型采用 LSTM 作为序列分类器,大模型采用 Llama 模型作为分类器。

表 6 的实验结果可以观察到,在 3 个数据集上,大模型的平均 $F1$ 分数是 0.959,小模型的平均 $F1$ 分数是 0.893。预训练的大模型的异常检测能力明显优于特定的小模型(这里主要是 LSTM)。这是由于其广泛的预训练使其具备了丰富的知识,使其能够处理更广泛的场景。为实现和大模型相当的检测结果,我们的协同结构需要分别在 3 个数据集上路由 17.9%、36.9% 和 31.8% 的数据。在小模型本就能够表现良好的数据集(HDFS)上,获得与大模型相当性能所需要的路由比例下降。总体而言,本文所提出架构能够在显著节省系统整体开销同时,保持较高检测检测能力。

Table 6 Overall Performance of the Edge-Cloud Collaborative Framework

表 6 云边协同框架整体性能 %

数据集	边缘小模型 $F1$	云端大模型 $F1$	协同框架 $F1$	云端路由比例
HDFS	97.9	99.7	99.6	17.9
BGL	83.5	91.6	90.8	36.9
Thunderbird-mini	85.7	96.6	96.2	31.8
平均	89.3	95.9	95.5	26.6

为系统评估本文提出的两阶段蒸馏策略在大小模型协同框架中的有效性,我们设计了消融实验。实验在 BGL 数据集上进行,严格控制其他变量以保证对比公平性。所有实验均采用相同的置信度阈值 $\delta = 0.9$,其中 w/o KD 表示基线模型,小模型不经过任何蒸馏;w/o Two-stage KD 是对比模型,小模型仅接受常规单阶段蒸馏。本实验重点关注 2 个关键评价指标:$F1$ 和路由比例(routing ratio)。精确率衡量系统整体检测准确度;路由比例表示需要调用大模型处理的样本比例,该指标直接反映系统的计算开销。两阶段蒸馏消融实验结果如表 7 所示。

Table 7 Ablation Study of the Two-Stage Distillation on BGL Dataset

表 7 BGL 数据集上的两阶段蒸馏消融实验 %

数据集	边缘小模型 $F1$	云端大模型 $F1$	协同框架 $F1$	云端路由比例
w/o KD	86.5	92.9	89.6	39.1
w/o Two-stage KD	88.1	93.2	90.5	37.5
本文	88.4	93.4	90.8	36.9

从表 7 可以发现,相较于 w/o KD, w/o Two-stage KD 的 $F1$ 由原来的 86.5% 提高到 88.1%,同时在相同置信度阈值下,路由比例下降了 1.6 个百分点。这一结果表明,通过蒸馏大模型的知识,小模型的判别能力得到增强,能够更准确地处理更多样本,从而降低对大模型的调用需求。

此外,本文提出的两阶段蒸馏策略进一步放大了这一优势,与 w/o Two-stage KD 相比,本文方法将 $F1$ 提升至 88.4%,同时将路由比例显著降低至 36.9%。这意味着在相同的置信度阈值设定下,需要调用大模型处理的样本数量进一步减少。通过两阶段的蒸馏过程,小模型不仅习得了大模型的输出分布,还逐步掌握了更精细的决策边界,从而在保持高 $F1$ 的同时,有效降低了对大模型的依赖。

4.6 动态路由有效性

为验证所提出时延敏感动态路由机制的有效性,

本文针对典型应用场景设置了 3 种不同的时延约束参数 λ=50 ms(高时延敏感场景)、λ=100 ms(中等时延敏感场景)以及 λ=1 000 ms(宽松时延场景)。从检测准确率与系统总体检测时延 2 个维度对模型性能进行了系统评估，实验结果如图 3 所示。

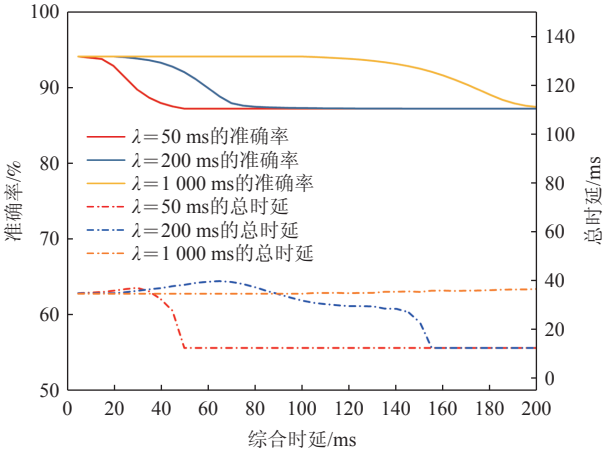


Fig. 3 Dynamic routing latency adaptation

图 3 动态路由时延适应

由图 3 可以发现，在不同 λ 设定下，系统整体检测准确率均随网络时延的增加呈下降趋势，其中在 λ = 50 ms 的高敏感场景下，准确率下降最为显著。这一现象表明，在严格时延约束下，系统更倾向于选择低时延的小模型进行推理，从而在一定程度上牺牲了准确率。然而，系统最终准确率稳定在 87.23%，该数值与小模型的独立检测准确率一致，说明小模型在极端网络条件下为系统提供了稳定的“兜底”能力，保障了最低可接受精度水平。在时延表现方面，动态路由机制为系统提供了相对稳定的时延预期。以 λ = 1 000 ms 场景为例，系统总体时延随网络时延增加仅呈现小幅波动，并在较大区间内保持稳定。当网络时延上升至不可接受水平时，系统通过降低对大模型的调用频率，转而依赖低时延的小模型进行推理，以牺牲部分准确率为代价换取时延优势，从而整体上保证了系统在不同网络条件下的可用性与稳定性。

5 结 论

针对物联网终端设备日志异常检测面临的语义建模不足、边缘资源受限与云端调用成本高等问题，本文提出一种集“语义增强检测—云边协同推理—系统化部署”于一体的技术方案。本方案所提 BERT-Llama 日志检测模型通过正则归一化、语义投影和三阶段微调有效提升了原始日志序列的异常识别能力；所提云边协同框架通过两阶段蒸馏和时延敏感动态路由，在保持总体检测准确率的同时显著降低了大模型调用比例。实验结果显示，语义增强检测模型在 HDFS、BGL 和 Thunderbird-mini 数据集上分别获得 99.7%、91.6% 和 96.6% 的 $F1$ 分数；协同框架在总体 $F1$ 分数保持 95.5% 的同时，将平均云端路由比例降低到 26.6%。系统测试进一步验证了方法在实时性、吞吐性和稳定性方面的优势。后续工作将进一步研究未知异常迁移、多源日志融合以及基于强化学习的自适应路由优化方法，以增强模型的跨场景泛化能力。

作者贡献声明：陈大江提出了云端协同的日志异常检测架构，进行了实验指导和论文修改；安一韬设计了基于 BERT 与 Llama 的语义增强检测模型，进行了实验验证和论文撰写；焦丽华设计了三阶段微调方案，进行了实验验证和 4.2.4 章节撰写和修改；肖昱璇设计了两阶段置信度蒸馏方案，进行了实验验证和 4.3.2 章节撰写和修改；邓伏虎设计了时延敏感动态路由策略，进行了实验指导和论文修改。

参 考 文 献

[1] Li Jingwei, Sun Jia, Yang Jingyuan, et al. Cloud-edge integrated secure storage architecture and technical challenges[J]. Scientia Sinica Informationis, 2025, 55(3): 516−527 (in Chinese)
(李经纬, 孙嘉, 杨劲远, 等. 云边融合安全存储架构及技术挑战[J]. 中国科学 (信息科学), 2025, 55(3): 516−527)

[2] Xu Wei, Huang Ling, Fox A, et al. Online system problem detection by mining patterns of console logs[C]//Proc of 2009 9th IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2009: 588−597

[3] Oliner A, Stearley J. What supercomputers say: A study of five system logs[C]//Proc of 37th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks (DSN'07). Piscataway, NJ: IEEE, 2007: 575−584

[4] Du Min, Li Feifei, Zheng Guineng, et al. DeepLog: Anomaly detection and diagnosis from system logs through deep learning[C]//Proc of the 2017 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 1285−1298

[5] Meng Weibin, Liu Ying, Zhu Yichen, et al. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs[C]//Proc of the 28th Int Joint Conf on Artificial Intelligence. 2019, 19(7): 4739−4745

[6] Zhang Xu, Xu Yong, Lin Qingwei, et al. Robust log-based anomaly detection on unstable log data[C]//Proc of the 2019 27th ACM Joint Meeting on European Software Engineering Conf and Symp on the

Foundations of Software Engineering. New York: ACM, 2019: 807−817

[7] He Pinjia, Zhu Jieming, Zheng Zibin, et al. Drain: An online log parsing approach with fixed depth tree[C]//Proc of 2017 IEEE Int Conf on Web Services (ICWS). Piscataway, NJ: IEEE, 2017: 33−40

[8] Devlin J, Chang M W, 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

[9] Touvron H, Martin L, Stone K, et al. Llama 2: Open foundation and fine-tuned chat models[J]. arXiv preprint, arXiv: 2307.09288, 2023

[10] Zhou Yingjie, Yang Min, Wu Di, et al. An intelligent lightweight model construction solution for anomaly detection in cloud platforms[J]. Journal of Computer Research and Development, 2025, 62(10): 2481−2494 (in Chinese)
(周颖杰, 杨敏, 吴迪, 等. 一种云平台中的异常检测轻量化模型智能构建方案[J]. 计算机研究与发展, 2025, 62(10): 2481−2494)

[11] Chen Xiaoping, Li Tao, Li Lin, et al. A software automatic vulnerability repair method based on large language models[J]. Computer Technology and Development, 2025, 35(7): 63−70 (in Chinese)
(陈霄萍, 李涛, 李琳, 等. 一种基于大型语言模型的软件漏洞自动修复方法[J]. 计算机技术与发展, 2025, 35(7): 63−70)

[12] Le V H, Zhang Hongyu. Log-based anomaly detection without log parsing[C]//Proc of the 36th IEEE/ACM Int Conf on Automated Software Engineering (ASE). Piscataway, NJ: IEEE, 2021: 492−504

[13] Lee Y, Kim J, Kang P. LAnoBERT: System log anomaly detection based on bert masked language model[J]. Applied Soft Computing, 2023, 146: 110689

[14] Ye Jingwei, Liu Chunbo, Gu Zhaojun, et al. LogOW: A semi-supervised log anomaly detection model in open-world setting[J]. Journal of Systems and Software, 2025, 222: 112305

[15] De la Cruz Cabello M, Prince Sales T, Machado M R. AIOps for log anomaly detection in the era of LLMs: A systematic literature review[J]. Intelligent Systems with Applications, 2025, 28: 200608

[16] Wang Zumin, Tian Jiyu, Fang Hui, et al. LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge[J]. Computer Networks, 2022, 203: 108616

[17] He Jiewei, Song Chao, Hu Ruilin, et al. LogHGAD: A hypergraph-based log anomaly detection method for AIoT systems[J]. Journal of Systems Architecture, 2026, 170: 103630

[18] Zhang Tianzhu, Qiu Han, Castellano G, et al. System log parsing: A survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(8): 8596−8614

[19] Min B, Ross H, Sulem E, et al. Recent advances in natural language processing via large pre-trained language models: A survey[J]. ACM Computing Surveys, 2023, 56(2): 1−40

[20] Chen Dong, Zhuang Yueting, Zhang Shuo, et al. Data shunt: Collaboration of small and large models for lower costs and better performance[C]//Proc of the AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2024, 38(10): 11249−11257

[21] Zhu Jieming, He Shilin, He Pinjia, et al. Loghub: A large collection of system log datasets for AI-driven log analytics[C]//Proc of the 2023 IEEE 34th Int Symp on Software Reliability Engineering (ISSRE). Piscataway, NJ: IEEE, 2023: 355−366



Chen Dajiang, born in 1982. PhD, associate professor. His main research interests include Internet of things security, trustworthy artificial intelligence, and artificial intelligence security.
陈大江, 1982 年生。博士, 副教授。研究方向为物联网安全、可信人工智能、人工智能安全。



An Yitao, born in 2001. Master candidate. His main research interests include Internet of things security and edge agents.
安一韬, 2001 年生。硕士研究生。研究方向为物联网安全、边缘智能体。



Jiao Lihua, born in 2002. Master candidate. Her main research interests include data security and trustworthy large language models.
焦丽华, 2002 年生。硕士研究生。研究方向为数据安全、可信大语言模型。



Xiao Yuxuan, born in 2003. Master candidate. Her main research interests include Internet security, vulnerability detection, and risk assessment.
肖昱璇, 2003 年生。硕士研究生。研究方向为互联网安全、漏洞检测、风险评估。



Deng Fuhu, born in 1984. PhD, associate professor. His main research interests include data security, artificial intelligence, and image processing.
邓伏虎, 1984 年生。博士, 副教授。研究方向为数据安全, 人工智能, 图像处理。