Computer Science and Application 计算机科学与应用, 2025, 15(4), 382-393
Published Online April 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.154111

Hans汉斯

基于StackLog的日志异常检测

徐　克[1,2]，胡　标[1,2]，刘　军[1,2*]

[1]温州大学计算机与人工智能学院，浙江　温州
[2]温州市智能网络重点实验室，浙江　温州

收稿日期：2025年3月22日；录用日期：2025年4月22日；发布日期：2025年4月29日

摘　要

日志信息是一种记录了系统运行状态的重要数据，是进行问题诊断、性能监控以及故障排除的主要依据。当系统出现故障时，通过详细系统地分析日志文本信息，研究人员可以快速准确地定位到系统出现问题的地方。通过分析现有的日志异常检测方法，本文发现当前用于日志异常检测的Text-CNN存在日志文本词向量维度急剧降低导致信息损失的问题。为了更充分地利用日志文本转换的词向量中携带的信息，本文提出了一种基于StackLog的日志异常检测方法。该方法采用堆叠卷积层逐步降低词向量维度的策略，尽可能保留词向量所携带的信息，并通过引入自注意力机制提升模型的检测能力。在HDFS和BGL两个公开数据集上进行对比实验验证了该模型在日志异常检测任务中的有效性。

关键词

深度学习，日志异常检测，StackLog，BERT

Log Anomaly Detection Based on StackLog

Ke Xu[1,2], Biao Hu[1,2], Jun Liu[1,2*]

[1]College of Computer and Artificial Intelligence, Wenzhou University, Wenzhou Zhejiang
[2]Wenzhou Key Laboratory for Intelligent Networking, Wenzhou Zhejiang

Received: Mar. 22nd, 2025; accepted: Apr. 22nd, 2025; published: Apr. 29th, 2025

Abstract

Log information is an important data that records the operating status of a system, and is the main basis for problem diagnosis, performance monitoring, and troubleshooting. When the system malfunctions, researchers can quickly and accurately locate the problem by analyzing the log text information in detail and systematically. By analyzing existing log anomaly detection methods, this paper finds that the current Text-CNN used for log anomaly detection has the problem of

*通讯作者。

文章引用: 徐克, 胡标, 刘军. 基于 StackLog 的日志异常检测[J]. 计算机科学与应用, 2025, 15(4): 382-393.
DOI: 10.12677/csa.2025.154111

information loss caused by a sharp decrease in the dimensionality of log text word vectors. In order to fully utilize the information carried in the word vectors of log text conversion, this paper proposes a log anomaly detection method based on StackLog. This method adopts a strategy of gradually reducing the dimensionality of word vectors by stacking convolutional layers, preserving the information carried by word vectors as much as possible, and improving the detection ability of the model by introducing self attention mechanism. Comparative experiments were conducted on two publicly available datasets, HDFS and BGL, to validate the effectiveness of the model in log anomaly detection tasks.

Keywords

Deep Learning, Log Anomaly Detection, StackLog, BERT

1. 引言

日志信息是一种记录了系统运行状态的重要数据，由时间戳、时序文本数据和其他信息组成。这些记录包括系统的运行状态、事件和错误信息，是进行问题诊断、性能监控以及故障排除的主要依据[1]。通过收集和分析日志，我们能够发现或预知系统中已发生或潜在存在的故障。

日志异常检测的目标是识别可能表明系统运行异常或存在潜在问题的日志条目[2]。传统的日志异常检测手段主要是通过人工分析运行过程中产生的日志数据，随着社会的发展，现代系统正朝着大规模化发展，例如部署在数千台商用机器上的Spark系统，以及拥有数千台处理器的超级计算机Blue Gene/L [3]，这必然会导致难以通过人工分析找出异常日志。

在日志异常检测领域的早期阶段，传统机器学习模型占据主导地位。例如，Xu 等[4]采用主成分分析(PCA)方法，通过为日志文件的不同会话附加会话 ID，实现对日志键的分组，并利用 PCA 检测异常。Liu 等[5]提出的隔离森林(IF)利用基于日志键序列频率构建的计数矩阵进行无监督学习。Lou 等人[6]提出的不变量挖掘(IM)通过挖掘日志事件计数向量中的线性关系来检测异常。

尽管这些传统机器学习模型在异常值识别方面取得了一定成就，但它们仍然存在显著局限性。随着深度学习技术在很多领域均取得不错的成绩，不少学者尝试将其引入到日志异常检测领域。

Lu 等人[7]通过将卷积神经网络(CNN)引入大数据系统日志的异常检测中，该方法首先通过日志解析将日志转换为日志键，然后利用这些日志键生成日志序列。接下来，他们对日志序列进行向量化，并将其输入到 CNN 模型中进行异常检测。曾闽川[8]等人提出一种基于匹配平均的联邦学习算法的日志异常检测模型 LogFTL，该框架在联邦学习场景下的效果达到了超越传统的日志异常检测模型。谢职权[9]提出的一种基于汉明相似度和最长公共子序列的高效在线解析算法 Ham-LCS，以及一种基于生成对抗网络的日志异常检测方法，并搭建了一个基于 ELK 的实时监控系统，解决了云平台上的日志异常检测问题。尹春勇[10]提出了一种基于时间卷积网络的通用日志序列异常检测框架。该方法将日志模板序列建模为自然语言序列，把基于神经网络训练的词嵌入作为模型的输入，以表示目标词在当前日志序列中的语义规则，并通过降维提高整个框架的运算效率。

在日志文本的处理方面，日志解析是数据分析中的早期步骤之一，但不准确的解析可能会影响异常检测模型的性能[11]。Zhu 等人[12]在 16 个数据集上测试了 13 个日志解析器，发现现有的解析器在进行

日志解析过程中存在问题，同一解析器在不同日志数据集上解析结果存在很大的差异。例如，LKE [13]在HDFS 数据集上解析精度高达 1，但在 BGL 数据集上却仅有 0.128 的精度。不仅如此，LogSig [14]在Proxifier 数据集中准确率为 0.967，而在 Linux 数据集中只有 0.169，这种不准确性无法保证解析的准确性。此外，He [15]等人在论文中指出日志挖掘对一些关键事件会十分敏感，解析中出现 4%的错误甚至会导致异常检测的性能下降一个数量级。因此解析精度的不稳定性可能会严重影响模型的预测准确度。

在当前日志异常检测领域，基于深度学习已经取得了一定的成绩，但仍然存在一些不足之处。首先，现有的大多数检测方法借助了日志解析器对日志文本进行处理，这便导致检测方法的准确度在一定程度上会依赖于解析器对日志文本的解析准确度；其次，检测模型在进行文本卷积时，Text-CNN 对由词向量嵌入模型转换后的高维张量卷积时维度降低过快，难以保证从文本转换得到的词向量不会失去信息。

为了克服前述问题，本文提出了一种基于 StackLog 的日志异常检测方法，该方法直接对日志文本进行处理，进而使用 BERT 对处理之后的日志文本进行文本向量化，最后将词向量输入到 StackLog 中进行异常检测，StackLog 通过逐次递减的向量宽度设计实现了特征空间的渐进式聚焦，随着网络深度增加逐步收缩特征维度，迫使模型在信息压缩过程中筛选出区分性强的关键模式，多种不同尺度卷积核的并行应用构建了局部到全局的语义关联网络，两者的结合使用使得该方法在日志异常检测方面表现出更好的性能。

2. 理论基础

2.1. 文本向量化

日志文本是一种用于检测异常的优秀方式，然而日志异常检测模型无法直接处理这些文本数据，因此需要将结构化的日志文本数据转换为能够输入到模型中使用的数据，这一过程被称为文本向量化。

目前主流的文本向量化包括独热编码、TD-IDF、词向量嵌入等方式，这些方法有效地将日志文本表示为词向量，但是也存在一些问题，这些方法大多简单地将其转换为词向量而忽略不同词之间的语义关系，使得转换为孤立的词向量，或者仅静态表示词语含义，而不考虑词语在句子中的上下文关系。为了更好地进行日志文本向量化，本文选择的文本词向量化方法为 BERT [16]。

针对 BERT 模型的训练主要可以分为两个阶段：预训练和微调。预训练阶段通过 Masked LM (MLM)和 Next Sentence Prediction (NSP)这两个无监督训练目标针对大规模无标签文本数据学习上下文信息。

BERT 通过 MLM 随机遮蔽输入序列中 15%的词汇，要求模型基于上下文预测被遮蔽词汇，迫使模型学习词汇间的双向依赖关系；然后使用 NSP 判断两个句子是否为连续段落，增强模型对篇章级语义的理解能力，实现了对文本深层语义的双向编码。

在完成预训练任务后，BERT 通过使用特定任务的标注数据进行参数微调使得更加适用于特定的下游任务，以达到优化特定任务的训练目标。通过预训练和微调这两个步骤的结合，BERT 在自然语言处理问题上发挥着重要作用。

在模型架构上，BERT 模型的核心是 Transformer 的编码器部分，通过使用多个相同的编码器堆叠，每个编码器都包含两个子层，分别为多头自注意力机制(Multi-Head Self-Attention)和位置前馈网络(Position-wise Feed-Forward Network)。多头自注意力机制通过使用多个注意力头并行计算，并且允许模型在不同的位置关注不同表征子空间的信息，通过将每个注意力头独立计算的结果拼接并线性变换得到最终的输出。而位置前馈网络部分则会对每个位置的表示进行非线性变换，以此来增强模型的表达能力。

对于输入序列 $X=\{x_1,x_2,\cdots,x_n\}$，第 l 层 Transformer 的输出计算过程可表示为：

$$Attention(Q,K,V)=soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

$$MultiHead\left(Q,K,V\right)=Concat\left(head_1,\cdots,head_h\right)\cdot W^O \tag{2}$$

$$head_i=Attention\left(QW_i^Q,KW_i^K,VW_i^V\right) \tag{3}$$

其中：Q，K，V 分别为查询、键、值矩阵，d_k 为维度缩放因子，h 为注意力头数。通过堆叠多层注意力机制，BERT 能够逐层抽象不同粒度的语义特征。

2.2. CNN

卷积神经网络(Convolutional Neural Network, CNN)是一种常用于图像处理方向的深度学习模型，一般由输入层、卷积层、池化层和全连接层这四部分构成完整的卷积网络，结构图如图 1 所示。

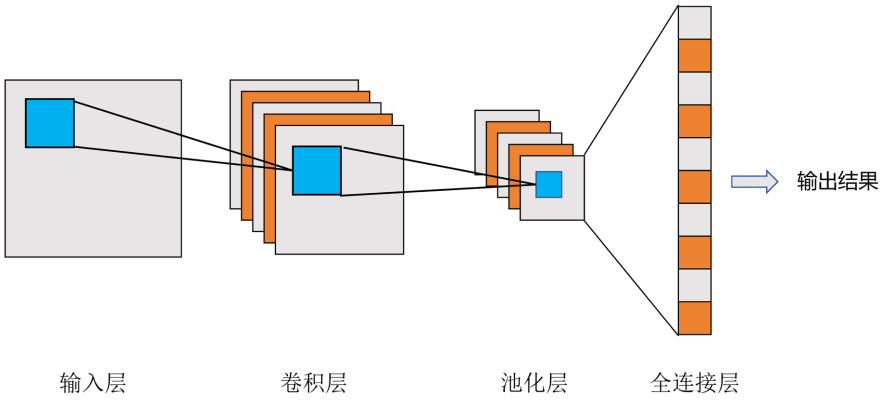

Figure 1. CNN structure diagram
图 1. CNN 结构图

输入层作为模型的起点，主要是将经过文本化的词向量输入到模型内部，进一步进行计算。

卷积层是其核心部分，卷积层的核心操作是通过卷积核在输入数据上进行滑动，进而计算局部区域的加权和的卷积运算。下面给出卷积运算的公式(以二维图像为例)：

$$O_{i,j}=\sum_{m,n}K_{m,n}\times I_{i+m,j+n} \tag{4}$$

其中 $O_{i,j}$ 是输出特征图在(i, j)位置的值，$K_{m,n}$ 是卷积核在(m, n)位置的权重，$I_{i+m,j+n}$ 是输入图像在$(i+m, j+n)$位置的像素值。

池化层作用是在保持主要特征不变的情况下，通过降低特征图的分辨率以减少计算量。常见的池化操作有最大池化(Max Pooling)和平均池化(Average Pooling)。下面以最大池化为例，它将输入特征图划分为多个不重叠的子区域，每个子区域中取最大值作为输出。假设池化窗口大小为 2×2，则池化操作的公式为：

$$O_{i,j}=\max_{m,n\in[0,1]}I_{2i+m,2j+n} \tag{5}$$

其中 $O_{i,j}$ 是输出特征图在(i, j)位置的值，$I_{2i+m,2j+n}$ 是输入特征图在$(2i+m, 2j+n)$位置的值。

全连接层的作用是对经过卷积计算之后提取的特征进行分类或回归。在经过一个或者多个卷积层和池化层后，再将最后一个池化层的输出展开成一维向量，并输入到全连接层得到最终结果，其运算公式为：

$$y=\sigma\left(Wx+b\right) \tag{6}$$

其中 y 是输出，x 是输入向量，W 是权重矩阵，b 是偏置向量，σ 是激活函数。

3. 日志异常检测模型

3.1. 方法框架

本文提出的基于 StackLog 的日志异常检测模型框架如图 2 所示。首先，对原始日志进行预处理，将文本序列转换为结构化文本。随后利用 BERT 模型进行词向量嵌入，从而获取日志词的语义向量。接下来，将经过处理的语义向量引入堆叠卷积网络和自注意力机制网络中进行训练，将训练完成的模型对新日志进行异常检测。

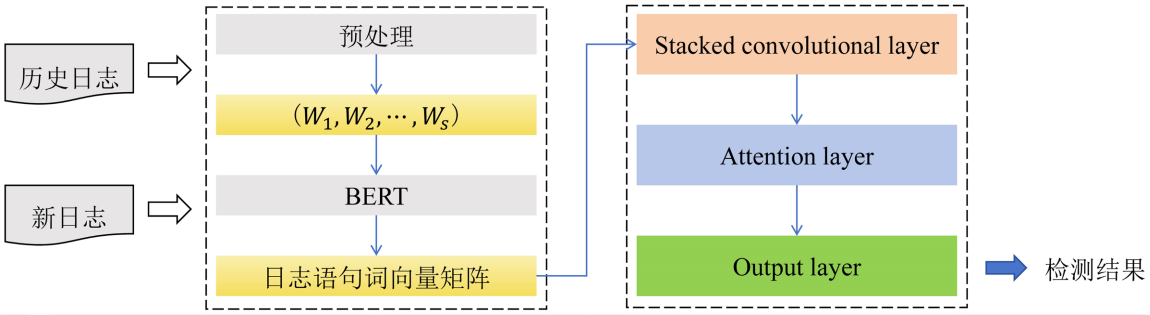

Figure 2. StackLog method framework diagram
图 2. StackLog 方法框架图

3.2. 数据预处理

上文指出本文对于日志的使用方式为直接使用原始日志，那么必然需要对原始日志进行一系列的处理使得日志可以进入下一步的文本向量化，这一步骤被称作日志文本数据的预处理，预处理步骤如算法 1 所示：

算法 1：日志数据预处理

输入：原始日志数据集合 $L = \{L1, L2, \cdots, Ln\}$
输出：预处理后的日志数据集合 L'
1) 定义常量：最大长度 MAX_LENGTH，填充字符 PAD_CHAR，无意义标点符号集合 PUNCTUATION_SET
2) 初始化空的日志语句集合 processed_logs
3) FOR log in log_line:
4) IF log in PUNCTUATION_SET:
5) 删除该字符 log
6) 用合适的分隔符(如空格、制表符等)将 log_line 分割成单词列表 words 拼接单词为语句：
7) 使用空格将 words 列表重新拼接成一个日志语句 new_log_line
8) IF len(new_log_line) > MAX_LENGTH：
9) MAX_LENGTH = len(new_log_line)
10) ELSE:
11) padding_length = MAX_LENGTH - len(new_log_line)
12) 在结尾填充 padding_length 个 PAD_CHAR
13) 将处理后的 new_log_line 添加到 processed_logs 集合中
14) 返回 processed_logs 集合，作为预处理后的日志语句集合

如图 3 给出了预处理具体示例，预处理的主要目的包括删除日志数据中的定界符、标点符号等一系列无意义的特殊字符和无意义的数字文本。除此以外，对于日志数据中使用驼峰规则串联起来的较长变量名，例如“DataNode”这种较长的变量名会被拆分为“Data”和“Node”，通过这种操作会明显降低

词汇表的大小，使一些合成词回归原本的含义，降低文本向量化的难度，具有相似含义的单词之间的联系性更加紧凑，模型在进行异常检测的时候会更加容易理解词语之间的关系，最后需要对处理成单词列表的日志文本进行大小写规范化。

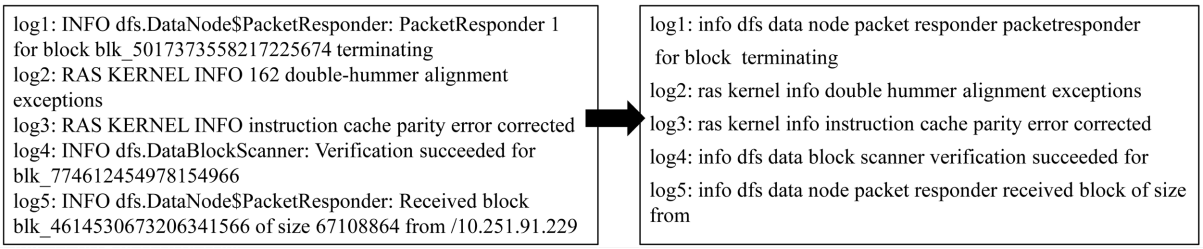

Figure 3. Preprocessing diagram
图 3. 预处理示意图

下面给出一条 HDFS 数据集的日志 L“2014-09-28 00:00:00,464 INFO org.apache.hadoop.hdfs.server.DataNode:Registered FS Dataset#0”，用来演示处理过程为例，针对这条日志，需要除去日志中包含的时间戳，日期等信息，这些信息无法反馈出日志异常情况，除此之外还需要将这里存在的逗号、冒号、杠、句号等标点符号去除，同时还需要将一些特殊的长单词拆分，这时会得到一组单词集合 $W_i\left(i\in\left[1,2,3,\cdots,p\right]\right)$，其中 i 表示日志中的第 i 个词，p 表示一条日志语句中的单词总数。完成之后，需要根据最大长度 MAX_LENGTH 和 p 的值判断是否需要进行填充，完成之后就会得到一条处理完成的日志{info, org, apache, hadoop, hdfs, server, data, node, registered, fsdataset, PAD_CHAR, ⋯, PAD_CHAR}。当将全部语句处理完毕后就完成日志语句预处理这部分内容。

3.3. 异常检测

CNN 通过使用卷积核在输入数据上进行滑动操作，提取局部特征，特别适用于图像和文本等数据的特征提取。在文本数据处理中，CNN 常用于捕捉文本的局部特征，例如单词或短语的组合，以获取语义信息。在日志异常检测方法中，Text-CNN [17]广泛应用于文本数据的处理。

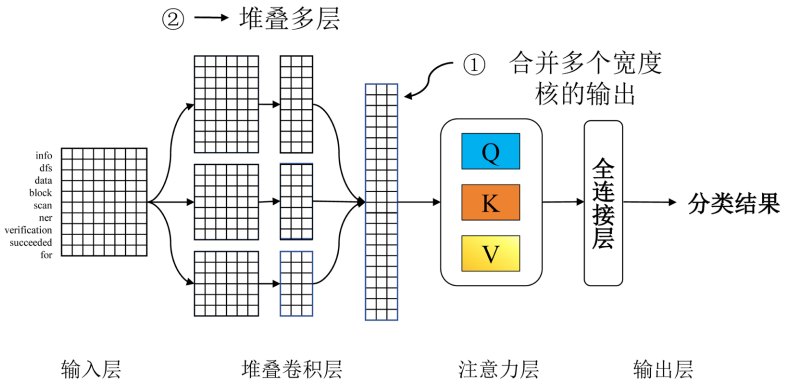

Figure 4. StackLog model structure diagram
图 4. StackLog 模型结构图

然而，将 Text-CNN 方法应用于文本向量的卷积时存在一个问题，使用与文本向量维度相同的卷积核直接压缩维度，比如日志文本的嵌入维度为 e，此时的卷积维度往往等于嵌入维度，导致高维词向量所携带的信息在经过一次文本卷积操作后丢失。这种维度的迅速压缩可能导致日志文本词向量中携带的信

息大量丢失。为了克服这个问题，本文通过改进 Text-CNN 的使用方法提出基于堆叠卷积网络和注意力机制构建的 StackLog，结构如图 4 所示，堆叠卷积网络负责提取转换后的日志文本词向量中的信息，通过注意力机制赋予不同部分不同的权重，以更有力地突显对日志异常检测有益的信息。StackLog 主要由四个部分构成，包括输入层、堆叠卷积层、注意力层和输出层，各层协同工作，实现对日志文本的深入分析与异常判断。

堆叠卷积层作为模型的核心组件之一，主要负责从输入的日志文本数据中提取特征。本文提出的日志异常检测模型在卷积层的设计上存在独特的策略，具体介绍如下：

1) 卷积核与维度调整

模型采用较小尺寸的卷积核，并通过堆叠多层卷积层来逐步降低词向量的维度 e。假设第 i 层卷积层的输入为 X_{i-1}，其维度为 (b,s,e_{i-1})，经过该层卷积操作后，输出为 X_i，维度变为 (b,s,e_i)，其中 $e_i < e_{i-1}$。具体的二维卷积操作计算过程如下：

$$X_i(m,n,p)=\sum_{u=0}^{k_{h_i}-1}\sum_{v=0}^{k_{w_i}-1}W_i(u,v,q,p)\cdot X_{i-1}(m+u,n+v,q)+b_i(p) \tag{7}$$

其中：W_i 是第 i 层的卷积核权重张量，其维度为(k_{h_i}, k_{w_i}, e_{i-1}, e_i)， k_{h_i} 和 k_{w_i} 分别是卷积核在高度和宽度方向上的尺寸。b_i 为第 i 层的偏置向量，m 和 n 是特征图在空间维度上的坐标，分别表示高度和宽度方向上的位置。p 和 q 是通道维度上的坐标，分别对应输出通道和输入通道。

这种维度的逐步降低是一个对数据进行抽象和提炼的过程。较小的卷积核能够捕捉到日志文本中局部的、细致的特征，类似于在图像识别中捕捉边缘、纹理等微观信息。而多层卷积层的堆叠则可以从不同层次和尺度对这些局部特征进行组合和整合，从而形成更具代表性的高层特征。在日志异常检测的情境下，这种方式有助于模型从原始的词向量信息中挖掘出与异常相关的关键特征，去除冗余信息，提升模型的处理效率和特征提取的准确性。

2) 多宽度卷积核捕捉单词关系

在卷积核的宽度维度上，模型使用多个不同大小的卷积核(分别表示为 $K_{w_1},K_{w_2},\cdots,K_{w_n}$)。这一设计源于日志文本中单词之间关系的多样性和复杂性。不同宽度的卷积核在对同一输入进行卷积操作时，能够捕捉到日志文本中不同范围和层次的单词之间的关系。例如，较窄的卷积核可以专注于捕捉相邻单词之间的紧密语义联系，可能对应着日志中一些固定的短语或具有强语义关联的词汇组合；而较宽的卷积核则能够跨越更大的范围，捕捉到更广泛的单词之间的潜在逻辑关系，有助于发现日志中一些隐含的语义模式。

将这些由不同宽度卷积核得到的输出结果进行合并(例如采用拼接操作)，记为 $Output_{concat}$，其过程可表示为：

$$Output_{concat}=Concat\left(Conv\left(X,K_{w_1}\right),Conv\left(X,K_{w_2}\right),\cdots,Conv\left(X,K_{w_n}\right)\right) \tag{8}$$

其中：$Conv(X,K)$ 表示输入 X 与卷积核 K 进行卷积操作。X 是经过前面处理后的日志文本特征表示。K 为不同宽度的卷积核，其维度根据具体宽度和输入输出通道数而定。Concat 表示拼接操作，它将不同卷积核得到的输出特征图在通道维度上进行拼接，从而得到一个包含了多种尺度特征信息的综合特征表示。

通过这种多宽度卷积核的方式，模型能够从多个角度全面地挖掘日志文本中的各种特征关系，最大程度地保留日志语句中的丰富信息，为后续的分析提供更充足、更具代表性的特征依据。

4. 实验结果与分析

4.1. 实验设置

为了验证本文提出的模型的有效性，在两个真实的数据集上进行实验：HDFS 数据集和 BGL 数据集，

数据集均来源于公开的日志数据集 Loghub (https://github.com/logpai/loghub)。日志数据集的汇总统计信息如表 1 所示：

Table 1. Summary information of log dataset
表 1. 日志数据集汇总信息

	Category	Size	Messages	Anomalies	Duration
HDFS	Distributed system	1.5 G	11,175,629	16,838	38.7 hours
BGL	Supercomputer	743 M	4,747,963	348,460	214.7 days

4.2. 评价指标

日志异常检测被视为一个二分类问题，为了评估模型在异常检测任务中的性能，本文采用了 Precision (精度)、Recall (召回率)和 F1-Score (F1 值)作为主要衡量指标：

1) 精度(Precision)：正确识别的正类别在所有被模型识别为正类别中的百分比：

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

2) 召回率(Recall)：正确识别为正类别占所有真实正类别的百分比：

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

3) F1 值(F1-Score)：Precision 和 Recall 的调和平均值。

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{11}$$

其中，TP (True Positive)是模型正确检测到的正类别的数量。FP (False Positive)是被模型错误地识别为正类别的数量。FN (False Negative)是模型错误地判别为负类别的数目。

4.3. 异常检测效果对比实验

本文选择三个机器学习方法 PCA [4]，SVM [18]，IM [6]和两个深度学习方法 LogRobust [11]，LogAnomaly [19]作为基准模型进行对比，HDFS 数据集的实验结果如表 2 所示：

Table 2. HDFS dataset experimental results
表 2. HDFS 数据集实验结果

对比方法	准确率	召回率	F1 值
PCA	0.63	0.96	0.76
SVM	0.93	0.94	0.93
IM	1.00	0.88	0.94
LogRobust	0.94	0.97	0.95
LogAnomaly	0.96	0.94	0.95
StackLog	0.97	0.95	0.96

从表中可明显观察到，在 HDFS 数据集上，与传统机器学习方法相比，各种深度学习方法表现更为优秀。尽管 IM 方法具有相对高的精度，但其相对较低的召回率表明在异常检测任务中性能较差。在异

常检测中，高召回率对于识别异常值至关重要，因为漏检可能导致巨大损失和潜在风险。几种深度学习方法中，StackLog 在精度和召回率方面均表现出色，在 F1 分数上取得最佳效果。各种模型的表现如图 5 所示。

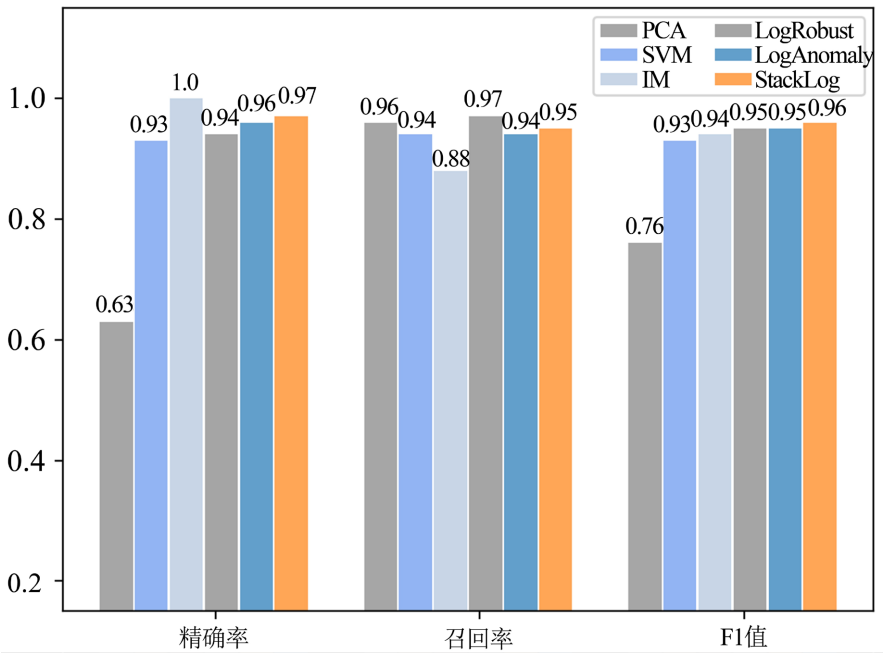

Figure 5. HDFS dataset experimental results
图 5. HDFS 数据集实验结果

在 BGL 数据集上的表现如表 3 所示：

Table 3. BGL dataset experimental results
表 3. BGL 数据集实验结果

对比方法	准确率	召回率	F1 值
PCA	0.50	0.61	0.55
SVM	0.93	0.30	0.45
IM	0.23	0.46	0.31
LogRobust	0.76	0.99	0.85
LogAnomaly	0.95	0.91	0.92
StackLog	0.96	0.97	0.97

在 BGL 数据集上，尽管 LogRobust 在召回率上表现较好，但其精度不够高，这意味着会将正常日志误判为异常，导致大量误报。这可能与 LogRobust 使用 Drain [20]日志解析器获取日志模板有关，而 Drain 解析器在解析 BGL 数据集上的准确性不足，这进一步验证了解析器的不准确性可能影响模型的检测性能。相反，本文的方法没有使用解析器，通过使用原始日志克服了这一问题，在检测效果保持不错的情况下，召回率也达到了 0.98，尤其在 F1 分数上达到了 0.97，充分展现了模型卓越的检测性能。各种模型的表现如图 6 所示。

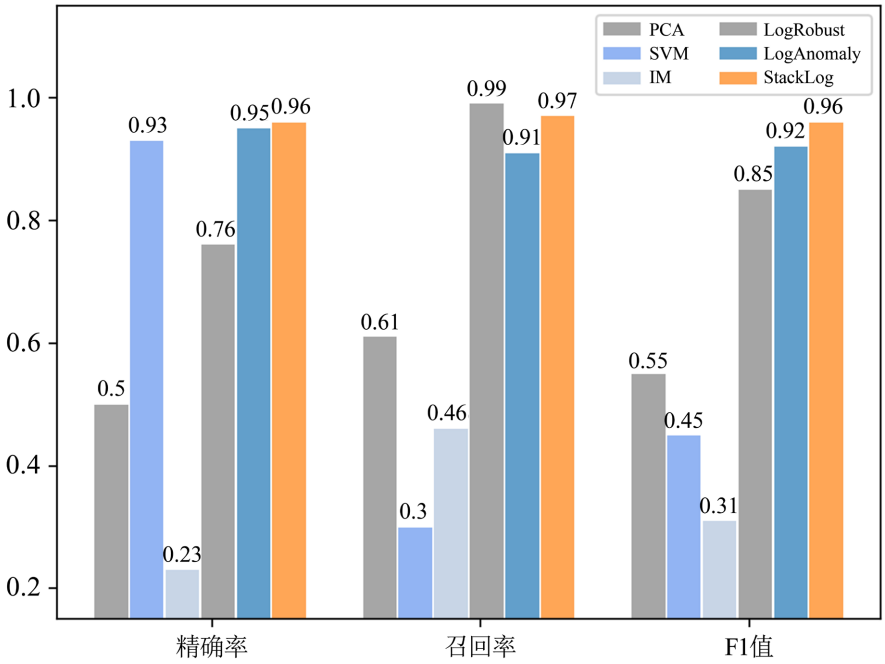

Figure 6. BGL dataset experimental results
图 6. BGL 数据集实验结果

4.4. 消融实验

为了评估多卷积核和堆叠卷积层对于日志异常检测模型性能的影响，本文在 BGL 数据集上进行了消融实验。其中 A 模型为去除堆叠卷积层，退化为 Text-CNN；B 模型将堆叠卷积层的层数减少为 1 层；C 模型是保证堆叠卷积层为 3，但是仅使用宽度为 2 的卷积核而非使用多种不同宽度的卷积核。其他参数保持不变，结果如表 4 所示。

Table 4. Results of ablation experiment
表 4. 消融实验结果

		A	B	C	StackLog
HDFS	准确率	0.912	0.953	0.946	0.976
	召回率	0.854	0.937	0.913	0.952
	F1 值	0.882	0.945	0.929	0.964
BGL	准确率	0.863	0.941	0.954	0.963
	召回率	0.704	0.936	0.928	0.974
	F1 值	0.775	0.938	0.941	0.968

从实验结果可以清晰地看出，当降低卷积层的数量时，也就意味着模型对于日志文本转换后的高维张量的维度降低速度加快。从表中可以看出，随着卷积层数量的减少，模型的检测能力降低，特别是当卷积层的数量减少到 0 时，本文的改进堆叠卷积网络退化为原始的 Text-CNN。在这种情况下，采用直接缩减所有维度的方式，严重损失了文本中携带的信息，进而导致模型的性能急剧下降，这可以从 F1 分数上清晰地表现。

另一方面，在去除多种不同卷积和处理时，可以看出模型在三个方面的检测能力都有所下降，这说明使用多个卷积核是明显可以捕捉到日志语句中相邻单词或一些隐含的单词之间的潜在逻辑关系的，当使用多卷积核相对于固定卷积核时，显然更好地发现了日志语句之间的关系，提升了日志异常检测模型对日志语句的检测准确度。

综合而言，实验结果清晰地说明了卷积层数量对于模型性能的重要性，以及多卷积对于模型性能的积极影响，说明了本文提出的日志异常检测模型的有效性。

5. 结语

本文旨在解决日志异常检测中存在的问题，并通过一系列创新性的方法提升检测模型性能。在数据预处理阶段，本文选择使用原始日志文本进行处理，避免了使用解析器方法可能导致的信息丢失问题。在检测模型方面，本文改变 Text-CNN 在文本卷积方面的使用，提出堆叠卷积网络，逐步降低词向量维度并且使用多卷积探索不同个数词语之间的关系，更好地保留了日志文本中的语义信息，提高了模型的检测性能。实验表明，本文提出的基于 StackLog 的日志异常检测模型可以有效地针对日志文本进行异常检测。

本文提出的 StackLog 日志异常检测模型通过使用 BERT 进行文本向量化，使用 StackLog 进行模型的搭建，使得模型的参数量较大，未来可以考虑轻量级的 BERT 或者探索知识蒸馏或量化技术降低模型的大小，以期望可以适配到边缘计算设备，用来满足工业场景的实时性需求。

参考文献

[1] Landauer, M., Onder, S., Skopik, F. and Wurzenberger, M. (2023) Deep Learning for Anomaly Detection in Log Data: A Survey. *Machine Learning with Applications*, **12**, Article 100470. https://doi.org/10.1016/j.mlwa.2023.100470

[2] 张颖君, 刘尚奇, 杨牧, 等. 基于日志的异常检测技术综述[J]. 网络与信息安全学报, 2020, 6(6): 1-12.

[3] BlueGene/L Message Types. https://www.usenix.org/cfdr-data#hpc4

[4] Xu, W., Huang, L., Fox, A., Patterson, D. and Jordan, M.I. (2009) Detecting Large-Scale System Problems by Mining Console Logs. *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, Big Sky Montana, 11-14 October 2009, 117-132. https://doi.org/10.1145/1629575.1629587

[5] Liu, F.T., Ting, K.M. and Zhou, Z. (2008) Isolation Forest. 2008 *Eighth IEEE International Conference on Data Mining*, Pisa, 15-19 December 2008, 413-422. https://doi.org/10.1109/icdm.2008.17

[6] Lou, J.G., Fu, Q., Yang, S., *et al*. (2010) Mining Invariants from Console Logs for System Problem Detection. *Proceedings of 2010 USENIX Annual Technical Conference*, Boston, 23-25 June 2010, 1-14.

[7] Lu, S., Wei, X., Li, Y. and Wang, L. (2018) Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network. 2018 *IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress*(*DASC/PiCom/DataCom/CyberSciTech*), Athens, 12-15 August 2018, 151-158. https://doi.org/10.1109/dasc/picom/datacom/cyberscitec.2018.00037

[8] 曾闽川, 方勇, 许益家. 基于联邦迁移学习的应用系统日志异常检测研究[J]. 四川大学学报(自然科学版), 2023, 60(3): 79-86.

[9] 谢职权. 云平台日志异常检测技术研究与实现[D]: [硕士学位论文]. 镇江: 江苏大学, 2023.

[10] Yin, C.Y. and Kong, X. (2024) Semi-Supervised Log Anomaly Detection Based on Bidirectional Temporal Convolutional Network. *Journal of Computer Applications Research*, **41**, 2110-2117.

[11] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., *et al*. (2019) Robust Log-Based Anomaly Detection on Unstable Log Data. *Proceedings of the* 2019 27*th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Tallinn, 26-30 August 2019, 807-817. https://doi.org/10.1145/3338906.3338931

[12] Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z. and Lyu, M.M. (2018) Tools and Benchmarks for Automated Log Parsing. arXiv:1811.03509.

[13] Fu, Q., Lou, J., Wang, Y. and Li, J. (2009) Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis. 2009 *Ninth IEEE International Conference on Data Mining*, Miami Beach, 6-9 December 2009, 149-158. https://doi.org/10.1109/icdm.2009.60

[14] Tang, L., Li, T. and Perng, C.-S. (2011) LogSig: Generating System Events from Raw Textual Logs. *Proceedings of the* 20*th ACM International Conference on Information and Knowledge Management*, Glasgow, 24-28 October 2011, 785-794.

[15] He, P., Zhu, J., He, S., Li, J. and Lyu, M.R. (2016) An Evaluation Study on Log Parsing and Its Use in Log Mining. 2016 46*th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN*), Toulouse, 28 June 2016-1 July 2016, 654-661. https://doi.org/10.1109/dsn.2016.66

[16] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2018) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

[17] Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, 25-29 October 2014, 1746-1751. https://doi.org/10.3115/v1/d14-1181

[18] Liang, Y., Zhang, Y., Xiong, H. and Sahoo, R. (2007) Failure Prediction in IBM Bluegene/l Event Logs. *Seventh IEEE International Conference on Data Mining* (*ICDM* 2007), Omaha, 28-31 October 2007, 583-588. https://doi.org/10.1109/icdm.2007.46

[19] Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., *et al*. (2019) Loganomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, 10-16 August 2019, 4739-4745. https://doi.org/10.24963/ijcai.2019/658

[20] He, P., Zhu, J., Zheng, Z. and Lyu, M.R. (2017) Drain: An Online Log Parsing Approach with Fixed Depth Tree. 2017 *IEEE International Conference on Web Services* (*ICWS*), Honolulu, 25-30 June 2017, 33-40. https://doi.org/10.1109/icws.2017.13