Article

LogEDL: Log Anomaly Detection via Evidential Deep Learning

Yunfeng Duan , Kaiwen Xue , Hao Sun , Haotong Bao , Yadong Wei , Zhangzheng You , Yuantian Zhang , Xiwei Jiang , Sangning Yang , Jiaxing Chen , Boya Duan , Zhonghong Ou *

Posted Date: 24 June 2024

doi: 10.20944/preprints202406.1597.v1

Keywords: log anomaly detection; evidential deep learning; uncertainty

Article

LogEDL: Log Anomaly Detection via Evidential Deep Learning

Yunfeng Duan [1,‡], Kaiwen Xue [2,‡], Hao Sun [1], Haotong Bao [2], Yadong Wei [2], Zhangzheng You [2], Yuantian Zhang [2], Xiwei Jiang [2], Sangning Yang [2], Jiaxing Chen [1], Boya Duan [1] and Zhonghong Ou [2,*]

[1] China Mobile Communications Group Co., Ltd, Beijing, China
[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
* Correspondence: zhonghong.ou@bupt.edu.cn
‡ These authors contributed equally to this work.

Abstract: With advancements in digital technologies such as 5G communications, big data, and cloud computing, the components of network operation systems have become increasingly complex, significantly complicating system monitoring and maintenance. Correspondingly, automated log anomaly detection has become a crucial means to ensure stable network operation and protect them from malicious attacks or failures. Conventional machine learning and deep learning methods assume consistent distributions between the training and testing data, adhering to a closed-set recognition paradigm. Nevertheless, in realistic scenarios, systems may encounter new anomalies that were not present in the training data, especially in log anomaly detection. Inspired by evidential learning, we propose a novel anomaly detector called LogEDL, which supervises training of the model through an evidential loss function. Unlike traditional loss functions, the evidential loss function not only focuses on correct classification, but also quantifies the uncertainty of predictions. This enhances the robustness and accuracy of the model in handling anomaly detection tasks, while achieving functionality similar to open-set recognition. To evaluate the proposed LogEDL method, we conduct extensive experiments on three datasets, i.e., HDFS, BGL, and Thunderbird, to detect anomalous log sequences. Experimental results demonstrate that our proposed LogEDL achieves state-of-the-art performance in anomaly detection.

Keywords: log anomaly detection; evidential deep learning; uncertainty

---

1. Introduction

Ensuring the stable operation of large modern systems is critical to business continuity, and a log of runtime status and events reflects the reliability of the system. Logs[1–3], as semi-structured text, typically consist of fixed templates and dynamic parameters that carry rich semantic information. In the event of a system failure, the most straightforward approach for maintenance personnel is to perform log analysis. However, manually identifying anomalies based on massive log data has become impractical[4,5]. Due to the rapid growth in log size, log analysis by experienced experts has become increasingly challenging[6]. As a result, automatic detection of system anomalies from logs has become critical. To address this problem, automated log analysis approaches are proposed that aim to automatically analyze logs using machine learning (ML) or deep learning (DL) techniques. In the field of log anomaly detection, traditional approaches for processing log sequences include word embeddings methods and RNN-based feature embedding methods[7]. However, these methods have difficulty in modeling long-term dependencies in processing lengthy sequences. In recent years, there has been a significant amount of research leveraging the Transformer architecture[8] to tackle natural language processing tasks, because it can model relationships between long sequences using attention mechanisms. Currently, pre-trained language models based on Transformer, such as BERT [9], RoBERT[10], GPT-3[11], BART[12], and T5[13] have demonstrated remarkable performance in various NLP tasks.Among them, BERT is pre-trained with two self-supervised learning methods: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM involves replacing certain tokens in the input sentences with [MASK] or other tokens and predicting their corresponding positions. NSP concatenates two sentences with the token [SEP] in between and predicts whether the two sentences are semantically related based on the [CLS] token positioned at the beginning of the input sequence[9]. While BERT has demonstrated excellent performance in various natural language processing tasks, its application to log anomaly detection, such as MIM tasks, still has certain limitations.

Traditional machine learning and deep learning methods typically assume that training and test data come from the same distribution, meaning the categories seen during training are entirely consistent with those encountered during testing. This approach is known as a closed-set recognition paradigm[14]. In this paradigm, the model classifies and recognizes data by learning features from the training data. However, its performance can be limited in practical applications, especially when encountering new, unseen categories or anomalies.For instance, in the task of log anomaly detection, data often follows a long-tail distribution[15], with anomalies being rare and dynamically changing. In a closed-set recognition paradigm, the model may not effectively discover new categories or anomalies in the logs. This limitation arises because the model is not equipped to handle cases where the test data includes classes or anomalies not present in the training data.

To address this challenge, the introduction of open-set recognition mode is particularly important[16]. In the open-set recognition mode, the model can more flexibly handle different categories or data distributions, meaning it can face different log anomaly situations during both training and testing phases. In addition to open-set recognition mode, there is another important approach called evidential deep learning(EDL)[17,18] for tasks such as anomaly detection. Evidential learning is a method for handling uncertainty, and its core idea is to better model and infer data by introducing uncertainty into the model. Unlike traditional closed-set recognition mode, Evidential learning not only considers the features of known data, but also quantifies the model's confidence in each prediction. In the context of anomaly detection, Evidential learning can help the model identify which data points have higher confidence and which may represent unseen anomalies. In this way, Evidential learning can not only identify known anomaly categories, but also provide inference and modeling for unknown anomaly situations, thereby improving the robustness and accuracy of anomaly detection systems. In summary, Evidential learning is an important method for handling uncertainty and unknown data, similar to the functionality of open-set recognition mode, and can provide a more comprehensive and reliable solution for tasks such as anomaly detection.

In this paper, we pay close attention to the issue of uncertainty estimation in log data and design an evidential learning loss function to supervise the network. Specifically, we replace the output parameters of the network with the parameter set of Dirichlet density. The Dirichlet distribution is a multivariate probability distribution that effectively describes the uncertainty of data. Through this substitution, our model can not only learn the distribution characteristics of normal log sequences but also make anomaly judgments by evaluating uncertainty when facing abnormal logs. To better reflect real-world scenarios, we only use normal log data to train the model, and then evaluate both normal and abnormal log data separately during testing.

The main contributions of this paper are as follows:

1. Analyzing the similarity between the log anomaly detection task and open-set recognition, and introducing the relationship and differences between open-set recognition and evidential learning, clarifying the applicability of evidential learning to log anomaly detection methods.
2. Proposing an evidential learning network, logEDL, suitable for log anomaly detection, enabling the model to identify unknown logs and quantitatively assess the uncertainty of unknown data, thereby assisting in identifying anomalous data.
3. Applying evidential learning algorithms to log anomaly detection tasks on public datasets, demonstrating the effectiveness and generalization ability of the proposed method.

2. Related Work

In this section, we first introduced the application of self-training paradigm in log anomaly detection. Subsequently, we mainly summarized the methods of semi-Supervised log anomaly detection based on deep learning in recent years.

2.1. Paradigm of Self-Training

Self-supervised pre-training is a significant research topic aimed at learning generalizable and shareable knowledge from large-scale data, which further benefits downstream tasks. The pretext tasks of self-supervised learning can be broadly divided into contrastive learning and masking modeling. In recent years, with the maturation of contrastive learning and masking modeling techniques, several self-supervised pre-training methods designed for log sequences have emerged.

In log sequences, the masking modeling paradigm is commonly applied, where the model is optimized by reconstructing the masked content from the unmasked portions. LogBERT[19] enhances the self-supervised training approach based on BERT[9] by introducing two novel self-supervised training tasks. To adapt to log sequences, masked log key prediction (MLKP) is proposed to enable the network to learn patterns of normal log sequences. The NSP of BERT is transformed into the volume of hypersphere minimization (VHM). LAnoBERT[20] utilizes masked language modeling based on BERT to train the model and employs the masked language modeling loss function for each log key during testing, enabling unsupervised anomaly detection. The LogFiT[21] model utilizes the RoBERTa model to handle log datasets containing up to 512 tokens per sample, while the Longformer model is employed for datasets where the token count per sample exceeds 512. Similarly, it employs BERT-based masked language modeling to learn the language patterns of normal log data. The encoder of BERT and its self-supervised paradigm enable it to generate contextually sensitive log keys for the input log sequences and reconstruct the masked log sequences. However, directly masking log sequences will disrupt the inherent structure of log sequences, making reconstruction overly challenging and unable to guide representation learning. In contrast to the direct reconstruction approach used in previous works, MMSM proposes a novel masking modeling task, which involves reconstructing the original log sequence from multiple randomly masked log sequences.

2.2. Semi-Supervised Anomaly Detection

Deeplog models the error between the predicted vector and the actual vector in the validation set as a Gaussian distribution. During testing, if the error falls within the high-confidence interval of the Gaussian distribution, the parameter value vector of the log entry is considered normal. Otherwise, it is deemed anomalous. LogBert establishes a candidate set consisting of g normal log keys. If the log key is in the candidate set, it is considered normal. Otherwise, it is deemed an anomalous log key. Subsequently, when a log sequence consists of more than r anomalous log keys, the sequence is labeled as anomalous. LAnoBERT utilizes computed loss errors and prediction probabilities for anomaly detection. When the probability of predicting a log occurrence is low, it is difficult to find the corresponding context in the normal log context. Thus, it is identified as an anomalous log sequence. LogFiT is trained only on normal log data using masked sentence prediction. When new log data is used, the $top - k$ prediction accuracy of the model can serve as a threshold to judge whether the new log data deviates from normal log data. The aforementioned methods distinguish anomalous log sequences based on data distribution or empirical thresholds. Enabling the neural networks to recognize log sequences belonging to anomalous data distributions is achieved by making the network autonomously determine whether a log sequence is anomalous. Placing a Dirichlet distribution on class probabilities, we treat the neural network predictions as subjective opinions and learn a function from the data that collects evidential leading to these opinions of deterministic neural networks. Leveraging evidential neural networks, the network can autonomously identify normal log sequences and anomalous log sequences.

3. Preliminary

3.1. Background of Open-World Learing

In real-world recognition/classification tasks, it is usually difficult to collect training data for all classes when training a recognizer or classifier due to various objective factors. The dataset is typically closed-set in deep learning, which the training and test data with the same annotations and feature space. Based on this assumption, traditional recognition and classification algorithms have achieved remarkable success in various machine learning tasks. However, real-world scenarios are often open-set and non-smooth, such as log fault detection, where the emergence of invisible situations significantly weakens the robustness of these existing approaches. To address this challenge, this requires a model that can both recognize known normal log sequences and reject unknown log faults.

The previous researchers have proposed open set recognition (OSR), which cannot fully learn all data knowledge during training, but the model can recognize unknown category data during testing. This requires the classifier to be able to accurately recognize not only the visible categories but also the unseen ones. Due to the small number of abnormal log sequences, the previous researches have usually used semi-supervised log anomaly detection methods, where only normal log sequences are targeted during training and abnormal log sequences need to be found during testing. A general framework for open set recognition learning algorithms can help us to continuously recognize unknown category data.

3.1.1. Open World Learning Algorithms

Mohsen Jafarzadeh et al.[16]developed a generalized framework for open-world learning algorithms containing six main elements: feature extraction, known classifications, novelty detection, novelty discovery, novelty management, and modified incremental learning. The implementation of AIOps is basically the same process as the open-world learning algorithm, except for the self-healing. One of the most important processes in AIOps is log anomaly detection, which belongs to the novelty detection in the open-set recognition framework. The complexity of open-world learning lies in the fact that the dataset is a function of time. Specifically, as time progresses, initially unknown data can be labeled to become known.

3.1.2. Open-World Learning Processes

Figure 1 shows the open-world learning process. The open-world stream is defined as $S_k^O = \{x_k \in Q_k | Q_k \subseteq \mathcal{W}_k\}$, which flow is a time sequence. Each member is a single or batch of data points x_k drawn from the query set Q_k at time step $k \geq 1$. $\mathcal{W}_k$ corresponds to the data of the whole world. The whole process of training the model at time step k is $Agent_k$. During open-world learning, the model trains a feature extractor and a decoder on a closed-set dataset. In the open-set world stream S^O, the sample x_k undergoes feature extraction to obtain the feature representation f_k through the feature extractor E_k. Subsequently, the decoder D_k is used to compute the probability P_k of the data belonging to unknown, known, and discovered classes, $P_k = \left[p_u,\ p_{K_1},\ p_{K_2}, \ldots, p_{K_{n-1}},\ p_{K_n},\ p_{D_1},\ p_{D_2}, \ldots, p_{D_{m_k-1}},\ p_{D_{m_k}}\right]^T$ where n is the number of known categories in the training set, m_k is the number of discovery categories. If the unknown probability p_u is less than a threshold τ, the buffer B_k is equal to the memory bank M_k. Otherwise, the buffer B_k is equal to the cascade of features f_k and memory banks M_k. Each instance of the buffer B_k is given a supervised (human or other agent) or unsupervised label by novelty discovery using the function C_k . Finally, the $Agent_k$ is updated by adapting to new data based on the buffer B_k and supervised/unsupervised labels. In the subsequent time step $k+1$, the agent $Agent_{k+1}$ will be utilized for improved incremental learning, continuing this cycle until all classes are discovered.

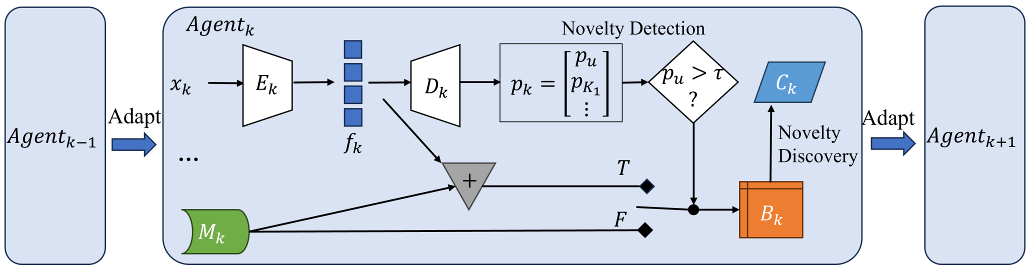


Figure 1. Open-world learning process

3.2. Background of Evidential Deep Learning

Existing deep learning-based models typically use a softmax layer at the top of deep neural networks (DNNs) for classification, using maximum likelihood estimation (MLE) to find the best network parameters. However, these softmax-based DNNs are unable to estimate predictions for classes outside of the closed set, because the softmax scores are inherently point estimates of the predictive distribution. For out-of-distribution categories, the outputs of softmax layer tend to be overconfident in false predictions. Out-of-distribution anomaly detection can be implemented in a variety of ways, with EDL [17,18]networks being a natural approach that allows the network to effectively say "no".

EDL overcomes the limitations of softmax-based DNNs by introducing dempster's evidential framework, shafer theory (DST)[22] and subjective logic (SL). EDL combines multi-class classification and uncertainty modelling to provide interpretable methods. Specifically, EDL interprets the standard output of the classification network as a distribution over any possible belief state categories. Assuming that category probabilities follow a prior dirichlet distribution, the set of distribution parameters is replaced with the parameters of the Dirichlet density for novelty detection.

4. Methodology

In this section, the main networks used in the proposed method and the architecture of the proposed model are explained. Section 4.1 gives the description and importance of the Masked Language Model (MLM), Section 4.2 presents Uncertainty in Log Sequence Detection, Section 4.3 introduces log preprocessing, Section 4.4 offers the problem definition, and Section 4.5 introduces the LogEDL model.

4.1. Masked Language Model

The operating mechanism of LogEDL proposed in this study is shown in Figure 2. Since LogEDL performs anomaly detection through the pre-training method of MLM of BERT and the EDL method, a detailed description of MLM is given in this section, and a detailed description of the EDL method in the log anomaly detection process is given in Section 4.2.

The reasons for using MLM for system log anomaly detection are as follows. Due to the existence of a large amount of log data, there is enough data to effectively pre-train BERT using the MLM approach, resulting in a large number of contextual and structural features of the log sequences.MLM is inspired by the completion task, which replaces certain tokens of the input sentence with [MASK] and then predicts the token in the [MASK] tokens. Bert is a large-scale bidirectional (fully contextual) trained model, which means that the model takes into account the contextual information of the entire input sequence when generating the representation of each word, unlike the pre-training of left- or right-directed language models.The LogEDL model uses the MLM-trained model for anomaly detection, which is expected to improve the generalization performance of the logs for effective anomaly detection.

Second, the task of labeling log sequences is very difficult, and the number of normal log sequences is much larger than the abnormal log data. Anomaly detection is a method based on unsupervised learning that uses only normal data for training to detect anomalies. While MLM is a self-training task, it does not need to label the data, and it is suitable for the case where the normal data is much larger than the abnormal data, which meets the purpose of anomaly detection. Therefore. Therefore, it is more suitable than supervised binary classification based methods for situations where there is a large amount of normal data.

Finally, from a practical operations perspective, experts detect anomalies by comparing the actual log keys with the generated log keys. While MLM methods can focus on patterns in context, they tend to have lower prediction probabilities for abnormal patterns[20]. However, MLM training methods do not yet have the ability to express anomalies.

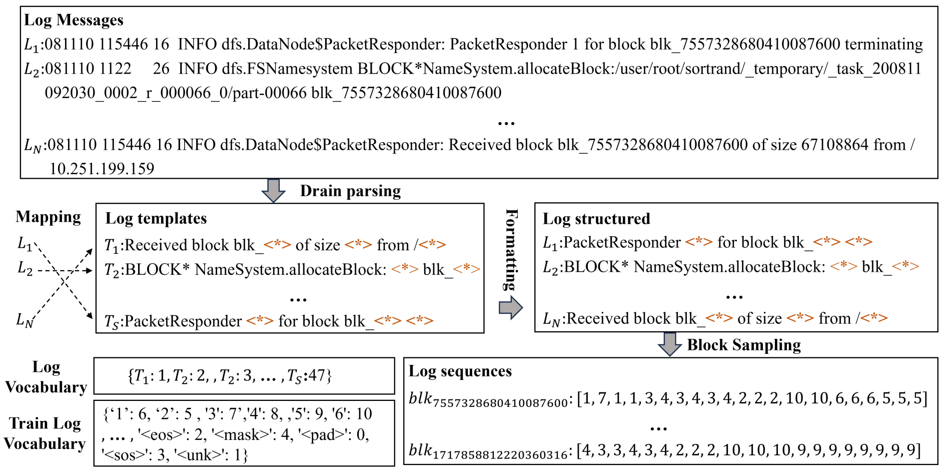
4.2. Uncertainty in Log Sequence Detection

In this paper, we apply EDL combined with MLM method to construct anomaly log sequence detection.

First, valuable semantic information is mainly contained in log sequence variations such as trends, period, and peaks and valleys, which may correspond to the system state. For log sequences, there is a correlation between the logs at each location in the log sequence and the nearby logs. Anomalous log sequences are often the result of unexpected logs appearing at log locations where they should be.

Second, recent models using the MLM approach in BERT do not have the ability to detect anomalies autonomously. However, researchers have been able to detect anomalies through some phenomena such as low prediction probability of anomalous logs and large loss values[20]. Because the logs on the mask can be in multiple situations, normal logs can be mistaken for anomalous logs by simple presentation methods.

We propose a log anomaly detection system, LogEDL, that exploits the fact that EDL can be characterized by high uncertainty values for unseen patterns in order to overcome the limitations of existing methods. Uncertainty modeling is performed for each [mask] location, and the uncertainty of the [mask] location responds to the likelihood of an anomaly occurring here.In the training phase, by applying EDL to the MLM learning of normal logs, we are essentially trying to collect evidence of the classes reflected by the context at each mask. In the testing phase, the assumption of contextual uncertainty at each mask allows the model to "know the unknown". Therefore, uncertainty is utilized to detect anomalies in the logs.



Log Messages

```
L1:081110 115446 16  INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_7557328680410087600 terminating
L2:081110 1122     26  INFO dfs.FSNamesystem BLOCK*NameSystem.allocateBlock:/user/root/sortrand/_temporary/_task_200811
   092030_0002_r_000066_0/part-00066 blk_7557328680410087600
...
LN:081110 115446 16 INFO dfs.DataNode$PacketResponder: Received block blk_7557328680410087600 of size 67108864 from /
   10.251.199.159
```

Log templates

```
T1:Received block blk_<*> of size <*> from /<*>
T2:BLOCK* NameSystem.allocateBlock: <*> blk_<*>
...
TS:PacketResponder <*> for block blk_<*> <*>
```

Log structured

```
L1:PacketResponder <*> for block blk_<*> <*>
L2:BLOCK* NameSystem.allocateBlock: <*> blk_<*>
...
LN:Received block blk_<*> of size <*> from /<*>
```

Log Vocabulary: $\{T_1: 1, T_2: 2, , T_2: 3, \dots, T_S{:}47\}$

Train Log Vocabulary: {'1': 6, '2': 5 , '3': 7','4': 8, ,'5': 9, '6': 10 , ... , '<eos>': 2, '<mask>': 4, '<pad>': 0, '<sos>': 3, '<unk>': 1}

Log sequences

$blk_{7557328680410087600}$: [1, 7, 1, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 6, 6, 6, 5, 5, 5]
...
$blk_{1717858812220360316}$: [4, 3, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9]

Figure 2. Examples of preprocessing system log (HDFS dataset).

4.3. Preprocessing

Log parsing methods, such as Spell[23], IPLoM[24], Drain[25], and so on. According to previous studies[26], weak log parsing methods can negatively affect the performance of subsequent anomaly detection tasks.Drain has become a common method used by researchers with its superior accuracy and efficiency [27,28]. Therefore, in this work, we use Drain for log message parsing.As shown in

Figure 2, the HDFS dataset is used as an example. Each raw log message consists of a constant part (also known as "log keys" or "log events") and a variable part (also known as "parameters"). The logs are first parsed into log keys using drain, then each log is mapped to its key to obtain structured logs, and finally a vocabulary of log keys is constructed, and a log sequence is obtained by sorting the log data in time order according to the block of the log data.

4.4. Problem definition

Log anomaly detection in this study can be defined as follows. Given an unstructured sequence of log messages, we aim to detect whether this sequence is normal or abnormal. We can then define the log sequence as an ordered sequence of log keys $S = \{k_1, \ldots, k_t, \ldots, k_T\}$, where $k_t \in V$ represents the log key at position t , and V denotes the set of unique log keys extracted from the log messages. The goal of this task is to predict whether a new log sequence S is abnormal or not based on a training dataset $Dataset_{train} = \{S_i\}_{i=1}^{N}$ that contains only normal log sequences. To achieve this goal, LogEDL models normal log sequences and further derives an anomaly detection criterion to identify anomalous sequences. In the training phase, normal log key sequence keys are replaced by [MASK] tokens with a certain probability for network optimization; in the testing phase, the replacement is performed with the same probability to determine whether a log sequence is normal or not by the number of uncertainty log keys.

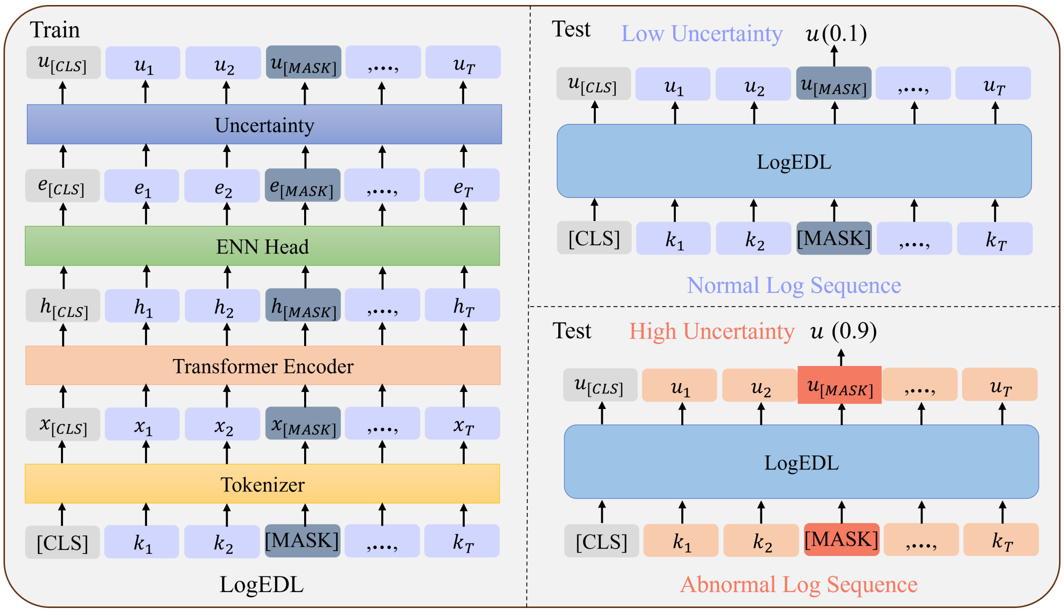

Train
$u_{[CLS]}$ u_1 u_2 $u_{[MASK]}$,..., u_T
Uncertainty
$e_{[CLS]}$ e_1 e_2 $e_{[MASK]}$,..., e_T
ENN Head
$h_{[CLS]}$ h_1 h_2 $h_{[MASK]}$,..., h_T
Transformer Encoder
$x_{[CLS]}$ x_1 x_2 $x_{[MASK]}$,..., x_T
Tokenizer
[CLS] k_1 k_2 [MASK] ,..., k_T
LogEDL
Test Low Uncertainty $u(0.1)$
LogEDL
Normal Log Sequence
Test High Uncertainty u (0.9)
LogEDL
Abnormal Log Sequence

Figure 3. The framework of logEDL

4.5. LogEDL

In this section, we present LogEDL, a semi-supervised log representation framework for anomaly detection.An overview of the proposed LogEDL framework is shown in Figure 3. It consists of three main components, Transformer Encoder, ENN head, and Uncertainty. Transformer Encoder is used after the log sequence is tokenized to extract the contextual information of the log sequence and transform it into semantic vectors. In the second step, ENN Head is used to convert the semantic vectors into evidential of the logs, which is the core of LogEDL, and finally Uncertainty is applied to make the judgment of anomaly detection based on the computed uncertainty. For this purpose, we build an evidential deep learning anomaly detection model. It mainly solves the problem of how to extract evidence of separable features with contextual information, which means that normal and abnormal sequences have boundaries in the feature space. Finally, the uncertainty of the extracted evidence is computed for anomalous log sequence detection. We describe each component in detail in the following subsections.

4.5.1. Transformer Encoder

We treat log sequences as sentences in natural language and use Transformer Encoder to extract the semantic vectors of log keys and capture the input log sequence contextual semantic information through a multi-head attention mechanism.Transformer Encoder consists of multiple transformer layers. Each transformer layer consists of a multi-head self-attention and a position-wise feed forward sub-layer using residual connection around the two sub-layers followed by layer normalization[8]. The sequence of logs $S = \{k_1, \ldots, k_t, \ldots, k_T\}$, where T is the sequence length. then represents each log key k_t as an input representation x_t, where the representation x_t are obtained by using the tokenizer, resulting in $X = \{x_1, \ldots, x_t, \ldots, x_T\}$ Secondly it is used as input to transformer encoder to obtain a d dimendional semantic vetor h_t, where d is set to 256.Finally we obtain the sequence of semantic vectors, written as $H = \{h_1, h_2, \cdots, h_t\}$. To summarize, the transformer encoder represents the log keys with semantic vectors, which preserves the semantic information of the events similar log keys also have similar semantic vectors, mitigating the effect of unstable logs and noise.The sequence of logs $S = \{k_1, \ldots, k_t, \ldots, k_T\}$, where T is the sequence length. then represents each log key k_t as an input representation x_t, where the representation x_t are obtained by using the tokenizer, resulting in $X = \{x_1, \ldots, x_t, \ldots, x_T\}$ Secondly it is used as input to transformer encoder to obtain a d dimendional semantic vetor h_t, where d is set to 256.Finally we obtain the sequence of semantic vectors, written as $H = \{h_1, h_2, \cdots, h_t\}$. To summarize, the transformer encoder represents the log keys with semantic vectors, which preserves the semantic information of the events similar log keys also have similar semantic vectors, mitigating the effect of unstable logs and noise.

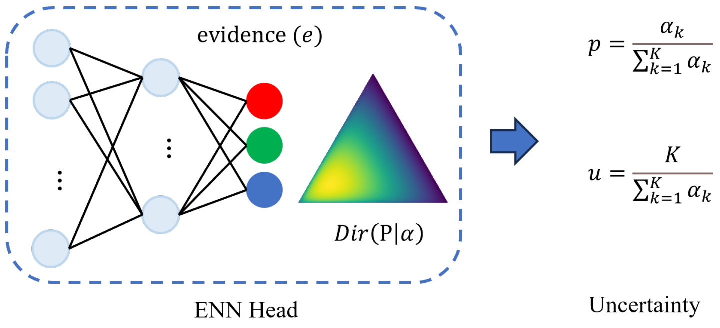

evidence (e)
$Dir(\text{P}|\alpha)$
ENN Head
$p = \frac{\alpha_k}{\sum_{k=1}^{K}\alpha_k}$
$u = \frac{K}{\sum_{k=1}^{K}\alpha_k}$
Uncertainty

Figure 4. ENN head

4.5.2. ENN head

After the Transformer Encoder, each log key is represented as a semantic vector. To detect anomalous sequences, we propose an evidential neural network head(ENN head) based on an uncertainty mechanism, as shown in Figure 4. This module includes two fully connected (FC) layers, which are used to learn the evidence of contextual patterns across the entire sequence.Given a sequence of semantic vectors, $H = \{h_1, h_2, \cdots, h_t\}$, the evidence $E = \{e_1, e_2, \cdots, e_t\}$ for this sequence is learned in two layers of FC after ENN head. Note that we have the last layer of hidden states as output, denoted as:

$$e_t = \sigma(w_2\sigma(w_1 h_t + b_1) + b_2) \tag{1}$$

where w_1, w_2, b_1, b_2 are the trainable parameters of the input h_t, and σ ( ) is the Relu function.

When we obtain the evidence e_t, it can efficiently learn the information about the mask position of the input log sequence in relation to the context and use the last step of the hidden state of the output as a representation of the context pattern of this log key position. To achieve this goal, we propose MMSM to model normal sequences, followed by log sequences-wise contrastive learning for representation learning, and further utilize an evidential neural network module to derive an anomaly detection criterion for identifying anomalous sequences.

4.5.3. Uncertainty

After obtaining the evidence, the Dirichlet prior distribution is utilized to compute the uncertainty, which is then used for subsequent anomaly detection. The Dirichlet distribution is a probability density function (pdf) that defines the possible values of a probability mass function (pmf) p. It is characterized by K parameters $\alpha = [\alpha_1, \ldots, \alpha_k, \ldots, \alpha_K]$ and is expressed as follows:

$$Dir(p|\alpha) = \begin{cases} \frac{1}{B(\alpha)} \prod_{k=1}^{K} p_k^{\alpha_k - 1}, & for \;\; p \in P_K \\ 0, & otherwise \end{cases} \tag{2}$$

where P_K is the K-dimensional unit simplex, $P_K = \left\{ p \mid \sum_{i=1}^{K} p_i = 1 \; and \; 0 \leq p_1, \cdots, p_K \leq 1 \right\}$ α_k represents the concentration parameter of the Dirichlet distribution,K denotes the number of log key classes.

According to belief theory [29], subjective logic provides a belief mass b_k for each singleton $k = 1, \cdots, K$, along with an overall uncertainty mass u. These $K + 1$ mass values are non-negative and sum to 1, i.e.

$$u + \sum_{k=1}^{K} b_k = 1 \tag{3}$$

where $u \geq 0$ and $b_k \geq 0$,for $k = 1, \ldots, K$.

A belief mass b_k for a singleton k is computed using evidence for singleton.Let $e_k \geq 0$ be the evidence derived for the k^{th} Singleton, then the belif b_k and the uncertainty u are computed as

$$b_k = \frac{e_k}{\mathcal{S}} \; and \; u = \frac{K}{\mathcal{S}} \tag{4}$$

where $\mathcal{S} = \sum_{k=1}^{K} \alpha_k$ is referred to as the Dirichlet strength,and a belief mass assignment, i.e., subjective opinion, corresponds to a Dirichlet distribution with parameters $\alpha_k = e_k + 1$.

The probability is given by

$$p_k = \frac{\alpha_k}{\mathcal{S}} \tag{5}$$

We optimized the network by setting a suitable loss function, enabling it to effectively learn normal log sequences. By leveraging uncertainty, we identified unknown anomalous logs, thus determining whether a log sequence is anomalous.We use the K-simplex (illustrated as a triangular heatmap in Figure 5) to jointly represent the token classification probability and uncertainty at the masked positions. The uncertainty detection at [MASK] is particularly suitable for log anomaly detection. Using a 3-class classification as an example and assuming the first class is the correct label, a well-calibrated model should give Accurate and Certain (AC) predictions (Figure 5(a)), indicating that the context for the [MASK] token is very certain. Accurate and Uncertain (AU) predictions (Figure 5(b)) indicate that the model is somewhat confused by the context and cannot classify it well. Inaccurate and Certain (IC) predictions (Figure 5(c)) suggest that the [MASK] token could have multiple possible values, which aligns well with the contextual variability in log sequences. Inaccurate and Uncertain (IU) predictions (Figure 5(d)) indicate that the context for the [MASK] token is completely unfamiliar, suggesting a high likelihood of the log sequence being anomalous.

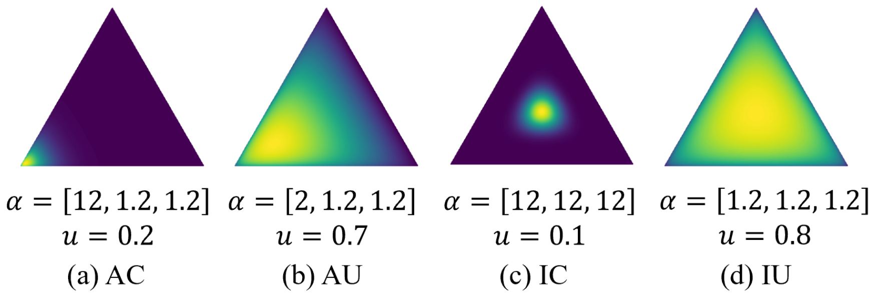


Figure 5. Examples of Probability Simplex.

4.5.4. anomaly detection

After training, we can deploy logEDL for detecting anomalous log sequences. The idea behind applying logEDL for log anomaly detection is that "there are differences in the context between normal system logs and anomalous system logs."In other words, when a language model trained solely on normal log data encounters the context of anomalous logs during testing, it is expected to exhibit higher uncertainty.

In this study, the defined uncertainty refers to the logit values produced by the model when predicting the [MASK] token, indicating the network's inability to make a judgment on this contextual pattern. Furthermore, when the uncertainty is high, it suggests that the corresponding context is difficult to find in normal contexts, thus identifying it as an anomaly.

Therefore, we can derive the anomaly score of a log sequence based on the predicted logits for the [MASK] token. To achieve this, given a test log sequence, we first randomly replace a certain proportion of log keys with the [MASK] token, similar to the training process, and use the randomly masked log sequence as input to logEDL. The uncertainty of the k^{th} log key refers to the degree of unknown represented by u_k when the log key at the [MASK] position is considered as the answer. For each [MASK] token, we compute u_k , which reflects the uncertainty at this masked position. For n [MASK] tokens, we select the $top - k\ u_k$ values and compute the final anomaly score to detect anomalies. The formula for calculating the anomaly score of a test log sequence is shown in Equation 6.

$$abnormal = \frac{1}{n} \sum_{k \in Top-n\ indices} u_k \tag{6}$$

4.5.5. Evidential Uncertainty Loss

To train the logEDL model, we follow the MLM paradigm to capture patterns in normal log sequences. The neural network architecture of logEDL is very similar to that of a classical neural network. The only difference is that we replace the softmax layer with an activation layer, such as ReLU, to ensure non-negative outputs, which serve as the evidence vector for predicting the Dirichlet distribution.

Given a sample k_i, let $f\ (\ k_i\ |\Theta)$ represent the evidence vector e_i predicted by the network for classification, where Θ denotes the network parameters. The corresponding Dirichlet distribution has parameters $\alpha_i = f\ (k_i\ |\Theta) + 1$. Once these distribution parameters are computed, their mean, $k_i / \mathcal{S}$, can be used as an estimate of the class probabilities. Let the ground truth y_i be the one-hot vector encoding the observation k_i , with $y_{ij} = 1$ and$y_{ij} = 0$ for all $i \neq j$. Here, α_i is the parameter of the Dirichlet density for the predictor.Assuming that class probability follows a prior Dirichlet distribution, the evidential mean squared error loss(EMSE) to be minimized for learning eventually reduces to the following form

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2024 doi:10.20944/preprints202406.1597.v1

$$
\begin{aligned}
\mathcal{L}^{i}_{EMSE}(\Theta) &= \int \|y_i - \mathrm{p}_i\|_2^2 \frac{1}{B(\alpha_i)} \prod_{i=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathrm{p}_i \\
&= \sum_{j=1}^{K} \mathbb{E}[y_{ij}^2 - 2y_{ij}p_{ij} + p_{ij}^2] \\
&= \sum_{j=1}^{K} (y_{ij}^2 - 2y_{ij}\mathbb{E}[p_{ij}] + \mathbb{E}[p_{ij}^2]) \\
&= \sum_{j=1}^{K} (y_{ij} - \alpha_{ij}/\mathcal{S}_i)^2 + \frac{\alpha_{ij}(\mathcal{S}_i - \alpha_{ij})}{\mathcal{S}_i^2(\mathcal{S}_i + 1)} \\
&= \sum_{j=1}^{K} (y_{ij} - p_{ij})^2 + \frac{p_{ij}(1 - p_{ij})}{(\mathcal{S}_i + 1)}
\end{aligned}
\tag{7}
$$

By decomposing the first and second moments, the loss function aims to achieve a joint objective of minimizing both the prediction error and the variance of the Dirichlet distribution generated by the neural network for each sample in the training set. In doing so, it prioritizes data fitting over variance estimation. Given that the patterns of normal log sequences and anomalous log sequences are different, we expect that once logEDL can correctly predict the masked log keys, it will be able to distinguish between normal and anomalous log sequences.

we can obtain the negative log marginal likelihood by integrating out the class probability, the evidential negative log likelihood loss (ENL) takes

$$
\begin{aligned}
\mathcal{L}^{i}_{ENL}(\Theta) &= -\log(\int \prod_{j=1}^{K} p_{ij}^{y_{ij}} \frac{1}{B(\alpha_i)} \prod_{i=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathrm{p}_i) \\
&= \sum_{j=1}^{K} y_{ij}(\psi(S_i) - \psi(\alpha_{ij}))
\end{aligned}
\tag{8}
$$

The same approach can be applied also to the cross-entropy loss, the evidential cross-entropy loss(ECE) will read

$$
\begin{aligned}
\mathcal{L}^{i}_{ECE}(\Theta) &= \int \left[\sum_{j=1}^{K} -y_{ij} \log(p_{ij}) \right] \frac{1}{B(\alpha_i)} \prod_{i=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathrm{p}_i \\
&= \sum_{j=1}^{K} y_{ij}(\psi(S_i) - \psi(\alpha_{ij}))
\end{aligned}
\tag{9}
$$

5. Experiment and Analysis

5.1. Experiment Setting

5.1.1. Dataset

We benchmark our log anomaly detection method on three log datasets: HDFS[30], BGL[27], and Thunderbird[31]. These datasets are sourced from the large-scale log dataset LogHub, published by Jieming Zhu et al. [32]. The statistics of these datasets are shown in Table 1. These datasets have been preprocessed as described earlier, with log data grouped into log sequences. For the HDFS dataset, data is partitioned using Block ID, while BGL and Thunderbird datasets are partitioned using a time window. The preprocessing results are shown in Table 2.

Table 1. Statistics of evaluation datasets

Dataset	Log Messages	Normal	Anomaly
HDFS	11,175,629	10,887,339	288,290
BGL	4,747,963	4399503	348,460
Thunderbird	20,000,000	1,241,438	758,562

Table 2. Preprocessing on HDFS, BGL, and Thunderbird Datasets

Dataset	Log Sequences	Log Keys	Train Dataset	Test Dataset		Average length
			Normal	Normal	Anomaly	
HDFS	742,527	47 (18)	167,466	558,223	16,838	19
BGL	37,315	334 (175)	13,718	20,579	3,018	562
Thunderbird	122,540	1,165 (866)	46,293	30,862	45,385	326

HDFS dataset was generated by running benchmark workloads in a private cloud environment and was manually labeled using hand-crafted rules to identify anomalies. Log data was divided into log sequences based on block IDs. Each log sequence associated with a block ID was assigned a ground truth label of either normal or anomalous. The HDFS dataset contains 11,175,629 log messages, grouped into 742,527 log sequences, with 725,689 normal sequences and 16,838 anomalous sequences. The average sequence length is 19. A total of 47 log keys were extracted from the dataset, with 18 log keys appearing in normal sequences. The model was trained on this dataset for 200 epochs with a masking ratio of 0.65.

BGL dataset comes from the BlueGene/L supercomputer system at the Lawrence Livermore National Laboratory (LLNL) in the United States. The logs include both alert and non-alert messages, with alert messages considered as anomalies. The BGL dataset contains 4,747,963 log messages, grouped into 37,315 log sequences, with 10,131 normal sequences and 2,630 anomalous sequences. For BGL, data was partitioned using a sliding time window of 5 minutes to generate log sequences, with an average sequence length of 562. A total of 334 log keys were extracted from the dataset, with 175 log keys appearing in normal sequences. The model was trained on this dataset for 200 epochs with a masking ratio of 0.50.

Thunderbird dataset is another large log dataset collected from a supercomputer system, specifically the Thunderbird microprocessor. The logs contain both alert and non-alert messages, identified by alert category labels. In the first column of the logs, "-" indicates a non-alert message, while other entries indicate alert messages. This labeling information can be used for alert detection and prediction research. The Thunderbird dataset contains 20,000,000 log messages, grouped into 122,540 log sequences, with 77,155 normal sequences and 45,385 anomalous sequences. For the Thunderbird microprocessor dataset, we used a sliding time window of 1 minute to generate log sequences, with an average sequence length of 326. A total of 1,165 log keys were extracted from the dataset, with 866 log keys appearing in normal sequences. The model was trained on this dataset for 200 epochs with a masking ratio of 0.50.

5.1.2. Implementation Details

The model was implemented using the open-source deep learning framework PyTorch, and it was run on a server equipped with an AMD EPYC 7773X 64-Core Processor and an Nvidia GeForce RTX 3090 GPU. The network was optimized using the Adam optimizer with a learning rate of 1e-4. A validation set was used to select the best model for testing.

5.1.3. Evaluation Metrics

In this study, similar to previous research, we use F1 score, Precision, and Recall as evaluation metrics to validate the effectiveness of the logEDL model for anomaly detection. Precision represents the percentage of true anomalies among all detected anomalies, while Recall represents the percentage of detected anomalies in the anomalous data set. The F1 score is calculated as the harmonic mean of these two metrics, as shown below:

$$precision = \frac{TP}{TP + FP} \tag{10}$$

$$recall = \frac{TP}{TP + FN} \tag{11}$$

$$F1\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{12}$$

where TP is true positive, FP is false positive, FN is false negative. The F1 score ranges from 0 to 1, with higher values indicating better performance of the classifier. A higher F1 score signifies that the model has a good balance between precision and recall, effectively capturing the overall performance in anomaly detection. This makes the F1 score a comprehensive metric for evaluating the model's effectiveness in identifying anomalies.

5.1.4. Baselines

To evaluate the performance of LogEDL, we compare it with the following log anomaly detection methods:

Principal Component Analysis(PCA) [30]: A traditional machine learning method for dimensionality reduction-based anomaly detection, combined with item weighting techniques from the data retrieval field. Suitable for simpler data scenarios.

IsolationForest (iForest)[33]: An unsupervised learning algorithm for anomaly detection that represents features as tree structures.

One-Class SVM (OCSVM) [34]: A well-known model for one-class classification, widely used for anomaly detection in log data where only normal data is observed.

LogCluster[4]: Based on similarity clustering methods, LogCluster automatically identifies similar log patterns and clusters them to achieve automatic anomaly recognition.

DeepLog[7]: An anomaly detection model based on log templates, which uses LSTM to model log data, automatically learns patterns of anomalous behavior, and compares predicted logs with real logs for anomaly detection.

LogAnomaly[35]: Integrates log templates by analyzing synonyms in logs, encodes log templates into semantic vectors, and inputs them into an LSTM model for anomaly detection.

LogBERT [19]: A method based on BERT, LogBERT constructs two self-supervised training tasks to learn patterns of normal log sequences and detects anomalies deviating from the normal model.

5.2. Results and Analysis

The comparative experiments is shown in Table 3, traditional anomaly detection methods, such as PCA, Isolation Forest, and OCSVM, have demonstrated suboptimal performance across the three datasets. Although these methods sometimes achieve high precision or recall,for instance, PCA's recall reaches 100% in all datasets,their F1 scores remain exceedingly low. This indicates an inability to balance precision and recall effectively, likely due to their failure to capture the temporal sequence information inherent in log data, resulting in inadequate anomaly detection capabilities.

LogCluster, a method specifically designed for log anomaly detection, shows improvements over traditional methods, particularly in the HDFS and BGL datasets. However, its performance in the Thunderbird dataset remains unsatisfactory, with F1 scores failing to reach high levels.

Table 3. Performance Comparison of LogEDL and Baseline methods.

Method	HDFS			BGL			Thunderbird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
PCA	5.89	100.00	11.12	9.07	98.23	16.61	37.35	100.00	54.39
iForest	53.60	69.41	60.49	99.70	18.11	30.65	34.45	1.68	3.20
OCSVM	2.54	100.00	4.95	1.06	12.24	1.96	18.89	39.11	25.48
LogCluster	99.26	37.08	53.99	95.46	64.01	76.63	98.28	42.78	59.61
DeepLog	88.44	69.49	77.34	89.74	82.78	86.12	87.34	99.61	93.08
LogAnomaly	94.15	40.47	56.19	73.12	76.09	74.08	86.72	99.63	92.73
LogBERT	87.02	78.10	82.32	89.40	92.32	90.83	96.75	96.52	96.64
LogEDL(Ours)	**90.06**	**92.80**	**91.41**	**99.84**	**97.26**	**98.53**	**97.80**	**98.08**	**97.91**

Table 4. Ablation experiments on different evidence loss functions

Method	HDFS			BGL			Thunderbird		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
logbert	87.02	78.10	82.32	89.40	92.32	90.83	96.75	96.52	96.64
LogEDL(w. ENL)	89.14	84.59	86.81	97.43	92.54	94.92	96.70	96.96	96.83
LogEDL(w. EMSE)	89.93	84.62	87.19	97.46	92.34	94.83	96.99	97.80	97.39
LogEDL(w. ECE)	**90.06**	**92.80**	**91.41**	**99.84**	**97.26**	**98.53**	**97.80**	**98.08**	**97.91**

DeepLog and LogAnomaly, two deep learning-based methods, significantly outperform traditional methods and LogCluster, showcasing the advantages of deep learning in capturing log sequence patterns. Both methods achieve high F1 scores across the three datasets, especially excelling in the HDFS and Thunderbird datasets.

LogBERT, a method based on pre-trained models and self-supervised learning tasks, demonstrates superior performance across all three datasets, with F1 scores markedly higher than those of traditional methods and other deep learning approaches. LogBERT successfully models normal log sequences and accurately identifies anomalous sequences through self-supervised pre-training tasks.

Our proposed method, LogEDL, surpasses all other methods across the three datasets, exhibiting exceptional performance. In the HDFS dataset, LogEDL achieves an F1 score of 91.41, significantly higher than LogBERT's 82.32. In the BGL dataset, LogEDL's F1 score of 98.53 substantially exceeds LogBERT's 90.83. In the Thunderbird dataset, LogEDL's F1 score of 97.91 outperforms LogBERT. This demonstrates that LogEDL maintains high precision and recall while achieving the best F1 scores.

The comparative analysis indicates that LogEDL outperforms other methods in log anomaly detection tasks. Traditional methods (PCA, Isolation Forest, OCSVM) exhibit significant shortcomings in balancing precision and recall, resulting in low F1 scores. The specialized method LogCluster shows some improvements but still lags behind deep learning methods. DeepLog and LogAnomaly demonstrate the strengths of deep learning, while LogBERT further enhances performance through self-supervised learning. Ultimately, LogEDL consistently achieves the best overall performance across the three datasets, proving its efficiency and reliability in log anomaly detection tasks.

5.2.1. Ablation Studies

The ablation experiments is shown in Table 4 Across all three datasets, the LogEDL model optimized with the ECE loss function demonstrates the best performance. ECE outperforms other methods in terms of precision, recall, and F1 score. Notably, in the BGL dataset, its precision reaches an almost perfect 99.84%.Compared to LogBERT, the LogEDL model shows improvements in all cases, especially when using the ECE loss function, resulting in significant enhancements.Therefore, for similar log anomaly detection tasks, it is recommended to use the LogEDL model optimized with the evidence cross-entropy loss function to achieve the best performance.

6. Conclusion

This paper proposes a log sequence anomaly detection framework based on evidential deep learning, named logEDL. This method goes beyond traditional log sequence anomaly detection approaches by introducing a new task that leverages uncertainty to identify anomalies. Specifically, logEDL not only prioritizes precise classification but also quantifies the uncertainty of prediction outcomes, thereby augmenting the model's robustness and accuracy in the context of anomalous log detection tasks. This functionality is similar to open-set recognition patterns. Experimental results show that compared to state-of-the-art anomaly detection methods, logEDL can maximally learn various contextual patterns of normal log sequences, thereby achieving advanced performance in anomaly detection tasks. In the future, we plan to further extend logEDL to large-scale and diverse log datasets to establish a foundational model for log sequence analysis.

7. Patents

This section is not mandatory, but may be added if there are patents resulting from the work reported in this manuscript.

Author Contributions: The conceptualization and design of the study were led by Y.D., K.X.,H.S. and Z.O. Experiments were carried out by Y.D.,K.X, H.B., X.J., and Y.W. Data were analyzed by Y.D., K.X., and Z.Y. The framework was developed by Y.D., Z.O., and K.X. The manuscript was primarily written by Y.D. and K.X., with substantial contributions from H.S., Y.Z., and Z.O. X.J., S.Y. and B.D. provided critical feedback and helped shape the research, analysis, and manuscript. Y.D.and K.X. made significant contributions to the preparation, execution, and analysis of this study, which justifies a shared first-authorship. As Y.D. initiated the study (together with B.D., J.C., and Z.O.), he is listed first among the shared first authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62076035.

Data Availability Statement: The research data of this paper is available by contracting the corresponding author upon reasonable requests.

Acknowledgments: The authors would like ti thank the anonymous reviewers for their invaluable comments.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
DL	Deep Learning
MLM	Masked Language Modeling
NSP	Next Sentence Prediction
EDL	Evidential Deep Learning
VHM	volume of hypersphere minimization
OSR	Open Set Recognition
DNNs	Deep Neural Networks
MLE	Maximum Likelihood Estimation
DST	Shafer Theory
SL	Subjective Logic
FC	Fully Connected
AC	Accurate and Certain
AU	Accurate and Uncertain
IC	Inaccurate and Certain
IU	Inaccurate and Uncertain
EMSE	Evidential Mean Squared Error Loss
ENL	Evidential Negative Log Likelihood Loss

ECE	Evidential cross-entropy Loss
LLNL	Lawrence Livermore National Laboratory
PCA	Principal Component Analysis
iForest	IsolationForest
OCSVM	One-Class SVM

References

1. He, S.; He, P.; Chen, Z.; Yang, T.; Su, Y.; Lyu, M.R. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* **2021**, *54*, 1–37.
2. Liu, Y.; Zhang, X.; He, S.; Zhang, H.; Li, L.; Kang, Y.; Xu, Y.; Ma, M.; Lin, Q.; Dang, Y.; others. Uniparser: A unified log parser for heterogeneous log data. Proceedings of the ACM Web Conference 2022, 2022, pp. 1893–1901.
3. Ma, L.; Yang, W.; Xu, B.; Jiang, S.; Fei, B.; Liang, J.; Zhou, M.; Xiao, Y. KnowLog: Knowledge Enhanced Pre-trained Language Model for Log Understanding. Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, 2024, pp. 1–13.
4. Lin, Q.; Zhang, H.; Lou, J.G.; Zhang, Y.; Chen, X. Log clustering based problem identification for online service systems. Proceedings of the 38th International Conference on Software Engineering Companion, 2016, pp. 102–111.
5. Xie, Y.; Zhang, H.; Zhang, B.; Babar, M.A.; Lu, S. Logdp: Combining dependency and proximity for log-based anomaly detection. Service-Oriented Computing: 19th International Conference, ICSOC 2021, Virtual Event, November 22–25, 2021, Proceedings 19. Springer, 2021, pp. 708–716.
6. Zhang, X.; Xu, Y.; Qin, S.; He, S.; Qiao, B.; Li, Z.; Zhang, H.; Li, X.; Dang, Y.; Lin, Q.; others. Onion: identifying incident-indicating logs for cloud systems. Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 1253–1263.
7. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, 2017, pp. 1285–1298.
8. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
9. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**.
10. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* **2019**.
11. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; others. Language models are few-shot learners. *Advances in neural information processing systems* **2020**, *33*, 1877–1901.
12. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* **2019**.
13. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* **2020**, *21*, 1–67.
14. Geng, C.; Huang, S.j.; Chen, S. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence* **2020**, *43*, 3614–3631.
15. Liu, Z.; Miao, Z.; Zhan, X.; Wang, J.; Gong, B.; Yu, S.X. Large-scale long-tailed recognition in an open world. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 2537–2546.
16. Jafarzadeh, M.; Dhamija, A.R.; Cruz, S.; Li, C.; Ahmad, T.; Boult, T.E. A Review of Open-World Learning and Steps Toward Open-World Learning Without Labels. *arXiv preprint arXiv:2011.12906* **2020**.
17. Amini, A.; Schwarting, W.; Soleimany, A.; Rus, D. Deep evidential regression. *Advances in neural information processing systems* **2020**, *33*, 14927–14937.

18. Sensoy, M.; Kaplan, L.; Kandemir, M. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems* **2018**, *31*.
19. Guo, H.; Yuan, S.; Wu, X. Logbert: Log anomaly detection via bert. 2021 international joint conference on neural networks (IJCNN). IEEE, 2021, pp. 1–8.
20. Lee, Y.; Kim, J.; Kang, P. Lanobert: System log anomaly detection based on bert masked language model. *Applied Soft Computing* **2023**, *146*, 110689.
21. Almodovar, C.; Sabrina, F.; Karimi, S.; Azad, S. LogFiT: Log anomaly detection using fine-tuned language models. *IEEE Transactions on Network and Service Management* **2024**.
22. Shafer, G. Dempster-shafer theory. *Encyclopedia of artificial intelligence* **1992**, *1*, 330–331.
23. Du, M.; Li, F. Spell: Streaming parsing of system event logs. 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016, pp. 859–864.
24. Makanju, A.A.; Zincir-Heywood, A.N.; Milios, E.E. Clustering event logs using iterative partitioning. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 1255–1264.
25. He, P.; Zhu, J.; Zheng, Z.; Lyu, M.R. Drain: An online log parsing approach with fixed depth tree. 2017 IEEE international conference on web services (ICWS). IEEE, 2017, pp. 33–40.
26. Huang, S.; Liu, Y.; Fung, C.; He, R.; Zhao, Y.; Yang, H.; Luan, Z. Paddy: An event log parsing approach using dynamic dictionary. NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2020, pp. 1–8.
27. Zhu, J.; He, S.; Liu, J.; He, P.; Xie, Q.; Zheng, Z.; Lyu, M.R. Tools and benchmarks for automated log parsing. 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2019, pp. 121–130.
28. Qi, J.; Luan, Z.; Huang, S.; Fung, C.; Yang, H.; Li, H.; Zhu, D.; Qian, D. Logencoder: Log-based contrastive representation learning for anomaly detection. *IEEE Transactions on Network and Service Management* **2023**.
29. Yager, R.R.; Liu, L. *Classic works of the Dempster-Shafer theory of belief functions*; Vol. 219, Springer, 2008.
30. Xu, W.; Huang, L.; Fox, A.; Patterson, D.; Jordan, M.I. Detecting large-scale system problems by mining console logs. Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, 2009, pp. 117–132.
31. Oliner, A.; Stearley, J. What supercomputers say: A study of five system logs. 37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07). IEEE, 2007, pp. 575–584.
32. Zhu, J.; He, S.; He, P.; Liu, J.; Lyu, M.R. Loghub: A large collection of system log datasets for ai-driven log analytics. 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2023, pp. 355–366.
33. Liu, F.T.; Ting, K.M.; Zhou, Z.H. 2008 eighth ieee international conference on data mining. Eighth IEEE International Conference on Data Mining, 2008, pp. 413–422.
34. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the support of a high-dimensional distribution. *Neural computation* **2001**, *13*, 1443–1471.
35. Meng, W.; Liu, Y.; Zhu, Y.; Zhang, S.; Pei, D.; Liu, Y.; Chen, Y.; Zhang, R.; Tao, S.; Sun, P.; others. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. IJCAI, 2019, Vol. 19, pp. 4739–4745.