Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 14, No. 2, 2025

Causal Entropy–Driven Generative Adversarial Learning for Anomalous Event Detection in Social Networks

Tian Tian [1, 3], and Tianqi Chen [2, 3, *]

[1] Science and Technology Innovation Promotion and Intelligence Research Center, Luoyang Henan 471000, China
[2] School of Computer Engineering, Shangqiu University, Shangqiu Henan 476000, China
[3] College of Information Engineering and Artificial Intelligence, Henan University of Science and Technology, Luoyang Henan, 471000, China

*** Corresponding author:** Tianqi Chen

Abstract: In social networks, anomalous events not only disrupt normal network evolution but also pose potential threats to network security. Therefore, achieving efficient anomaly detection in dynamically and semantically complex environments is of significant importance. However, existing methods often struggle to effectively capture the dynamic evolutionary relationships between events and their underlying causal dependencies. To address this, this paper proposes a causal entropy–driven generative adversarial learning for anomalous event detection in social networks. First, a log parser converts unstructured social network logs into structured events, which contain timestamps, IP addresses, and event content. Subsequently, we standardize causal entropy to measure the strength of causal relationships between events and construct causal sequences. Building upon this foundation, we design a generative adversarial framework. The generator combines a gated recurrent unit, a self-attention mechanism, and a long short-term memory network to capture deep event semantics and generate realistic log samples. The discriminator achieves precise differentiation between normal and abnormal events through mean cross-entropy loss on fully connected layers. Experimental results demonstrate that the proposed method achieves approximately 6.5% higher detection accuracy than existing approaches, validating its effectiveness and robustness in identifying anomaly events.

Keywords: Social Networks; Anomalous Event Detection; Generative Adversarial Learning; Long Short-Term Memory Networks; Causality Mining.

1. Introduction

Social networks are formed through interactions among individuals, resulting in relatively stable relationship structures that encompass both continuous relationships and discrete events. Within these networks, anomalous events—triggered by abnormal behaviors of individuals or communities such as deviant actions, false information dissemination, and cyberattacks [1]—can persist for specific durations and disrupt part or even the entirety of the network. These anomalies are often reflected in the massive logs generated by social networks, which record user activities and network requests [2] and capture underlying social patterns and information flow. Deviations in log behaviors, such as unusually high posting frequency or the replication of identical content across multiple accounts, may indicate spam propagation or malicious activity. Consequently, analyzing the global semantic relationships within log data is critical for safeguarding network security and stability. Existing research has explored clustering-based [3], topic modeling-based [4], and deep learning-based methods [5] for anomaly detection. While clustering methods identify events through grouping documents or features, topic models such as PLSA [6] and LDA [7] uncover hidden themes to determine event boundaries, and graph convolutional networks (GCNs) [8, 9] enhance classification by modeling heterogeneous word–document graphs. However, these approaches struggle to capture the dynamic evolution of social networks and the causal dependencies among events, thereby necessitating more comprehensive and robust solutions.

To address these challenges, this paper proposes a causal entropy–driven generative adversarial learning for anomalous event detection in social networks. Specifically, unstructured logs are first parsed into structured events containing timestamps, IP addresses, and event content, after which standardized causal entropy is employed to quantify causal strengths and construct event sequences. Building on this foundation, a generative model integrating gated recurrent units (GRUs), self-attention, and long short-term memory (LSTM) networks is designed to capture semantic context and generate realistic event samples, while the discriminator distinguishes normal from anomalous events by optimizing the average cross-entropy loss. The main contributions of this paper are as follows:

1. We propose a causality-entropy-driven event modeling method. This paper introduces normalized causality entropy into social network log analysis to quantify causal strengths between events and construct causal sequences. Compared to traditional time-based or correlation-based modeling methods, it more effectively reveals the underlying causal dependencies of anomalous events.

2. We design a multi-mechanism generative adversarial framework. This paper introduces a novel generative adversarial learning architecture, where the generator integrates gated recurrent units (GRU), long short-term memory networks, and self-attention mechanisms to simultaneously capture long-range dependencies and key semantic features. The discriminator achieves high-precision differentiation of abnormal events based on average cross-entropy loss, thereby enhancing detection robustness.

3. We conducted systematic experimental evaluations on public datasets, demonstrating a 6.5% improvement in detection accuracy over baseline models. This validates the effectiveness of the proposed framework in dynamic and semantically complex environments, and showcases its potential applications in cybersecurity and anomaly detection.

The remainder of this paper is organized as follows: Section 2 introduces related work. Section 3 presents the design and composition of the proposed methods. Section 4 evaluates the performance of the model, and Section 5 concludes the paper.

2. Related work

Anomaly detection is a fundamental task in areas such as internet monitoring, cybersecurity, and social network analysis [10]. Existing approaches for event detection in social networks can be broadly categorized into clustering-based, topic modeling-based, and deep learning-based methods, all of which aim to ensure network security and stability. In practical applications, researchers typically exploit features such as keywords, timestamps, geographic information, and tags. For instance, Li and Jung [11] proposed MALED by combining statistical techniques with keyword analysis, while Zuo et al. [12] constructed weighted co-occurrence graphs for event tracking. Zhang et al. [13] further integrated geographic data for local event monitoring, and Stilo et al. [14] addressed the short-text limitation of tweets by applying temporal cooccurrence and Symbolic Aggregate approXimation (SAX) to time-series modeling. In addition, Bountrogiannis et al. [15] adopted a neural variational document model for analyzing log data. Nonetheless, methods relying on fixed event numbers or constant time intervals often struggle to characterize dynamic and evolving events. Topic modeling has also been extensively employed for event identification. Li et al. [16] utilized probabilistic models to identify log events by uncovering latent topic distributions, whereas Xie et al. [17] exploited hashtags and LDA to reveal thematic structures. Diao et al. [18] introduced TimeUserLDA, which incorporates temporal signals and Poisson processes for bursty topic detection, and Wang et al. [19] proposed a GAN-based neural topic model to explore semantic correlations among latent topics. More recently, Logan et al. [20] developed Warble, a spatiotemporal probabilistic framework linking online social networks with realworld events. However, the requirement of predefined event numbers in most topic models limits their ability to detect emerging or evolving patterns.

Beyond topic modeling, clustering- and feature-driven methods have also been investigated for anomaly detection. Pu et al. [21] clustered historical and predicted log events with the aid of knowledge bases, while Yan et al. [22] proposed an invariant anomaly detection approach for unstructured logs. Omuya et al. [23] applied PCA to separate normal and abnormal log event spaces, and Gnouma et al. [24] employed stacked sparse autoencoders to extract critical events. Nevertheless, unsupervised learning methods often face performance degradation in terms of accuracy and stability due to the lack of labeled data.

Overall, prior studies provide valuable foundations but remain limited by assumptions of fixed event structures and insufficient modeling of causal relationships. To overcome these challenges, we introduce a causal entropy–driven generative adversarial approach that unifies causal dependency modeling with adversarial sequence learning, enabling more precise and robust detection of anomalous events in social networks.

3. Methods

In this paper, proposes a causal entropy–driven generative adversarial learning (GANAED) for anomalous event detection in social networks, aimed at identifying anomalous behaviors within the content of social network events. To facilitate better understanding, Figure 1 illustrates the architecture of the GANAED method, which is primarily divided into the following three key components:

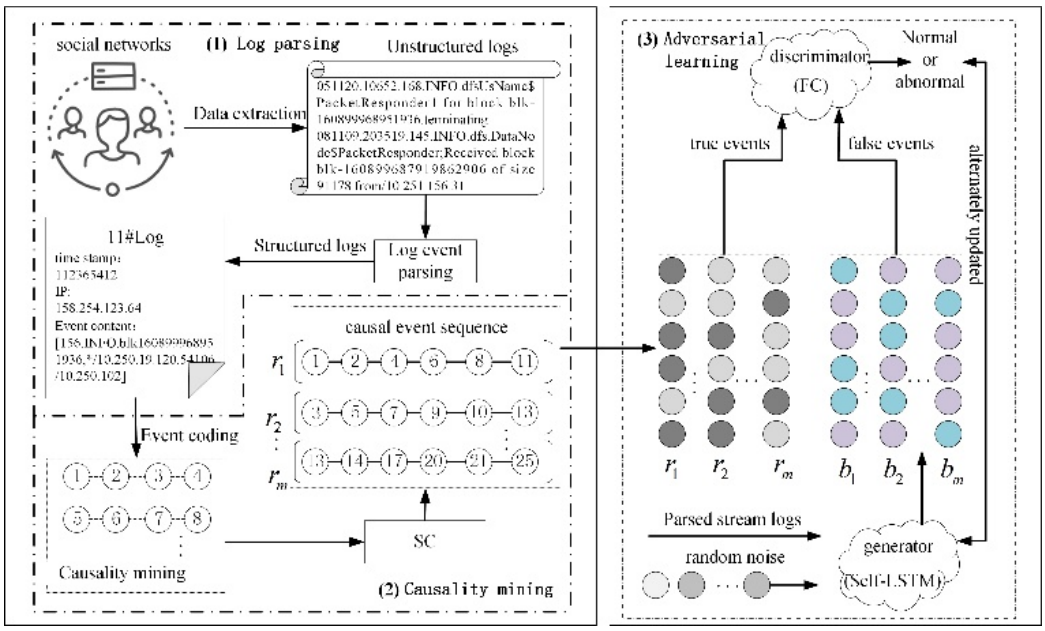

Fig 1. Method architecture.

1. Log parsing: This phase involves converting unstructured logs into a structured format for subsequent analysis. This process includes extracting crucial information such as timestamps, IP addresses, user behaviors, and event content.

2. Causality mining: Structured log events are encoded

chronologically, and standardized causal entropy is utilized to compute the causal strength between two log events. This step mitigates the misleading effects of indirect causal relationships resulting from multiple confounding factors. Understanding causality is crucial for building more precise predictive models and devising effective intervention strategies.

3. Generative adversarial learning: Parsed social flow logs are inputted into the generator, which is trained to generate events similar to real log events. The discriminator is then employed to distinguish between real log events and events generated by the generator. Through fully connected layers, it calculates their average cross-entropy total loss to discern normalcy or anomaly, thus enhancing the efficiency of anomaly event recognition.

3.1. Log parsing

The primary objective of log parsing is to convert unstructured log events into a structured format for subsequent data analysis and processing. This entails extracting key information such as timestamps, geographical locations, and user behaviors from raw logs and organizing them into a format conducive to machine learning algorithms. Through this transformation, subsequent models can effectively identify and analyze anomalous behaviors or activities in social networks, thereby enhancing social network security.

In social networks, logs typically consist of numerous log messages, each comprising three crucial components: timestamp, IP address, and event content. These are represented as triplets:

$$S = \langle T, IP, Event\_Con \rangle \tag{1}$$

In this section, the timestamps, which record the specific dates and times of events, are crucial for tracking event sequences, analyzing user behavior patterns, and identifying potential security threats. IP addresses provide the network location information where an event occurs, enabling the identification of users' geographical locations, tracking different login attempts, and monitoring potential abnormal activities or security vulnerabilities. The content of log events contains detailed information about the events, such as the specifics of user actions (e.g., posting text, commenting, liking), system status information, error codes, etc. Understanding the context and intent of events is key, and analyzing these thematic keywords and specific descriptions of events is vital for analyzing user behavior and abnormal patterns.

3.2. Causality Mining

There exists a certain causal relationship between events, and by mining the causal relationships between events, we can identify key events that have a significant impact on the dynamics of social networks, thereby enhancing the accuracy of anomaly detection. Therefore, we utilize a method based on information entropy proposed in literature [25] to measure the causal relationships between log events.

Firstly, we encode the parsed structured log events, aiming to convert each log event into a numerical form for subsequent data processing and analysis. Subsequently, we arrange these encoded log events in chronological order, which can reveal the temporal dependencies and continuity between events. To illustrate this process more intuitively, we provide an example of a log event sequence arranged in chronological order in Table 1. In the table, the encoding of each log event is listed in the last column. For instance, if the event content of a log is "Userbehavior*", its corresponding encoding is ; if another log's event content is "Info", then its encoding is *, and so forth. Such encoding not only reflects the uniqueness of each log event but also provides a simple and effective way for us to track and compare different log events. Through this encoding and time-series processing, we transform the originally chaotic and unordered logs into ordered and analyzable information, laying the foundation for us to explore the causal relationships between log events subsequently.

Table 1. Examples of log events

Code	T	IP	Event Con
e_1	0rz13654	168.201.50.26	Userbehavior*
e_2	0rz13e52	168.201.59.12	Info*
e_3	0rz14659	168.201.59.20	topics

To explore the causal relationship between two events, also known as a causal template, we define a binary (e_1, e_2). This binary represents a causal relationship between event e_1 and event e_2, where event e_1 is the cause of event e_2 and event e_2 is the result of event e_1. Among them, e_1 and e_2 are event identification codes, $i = 1,2,3,...n$, $j = 1,2,3,...n$, and n is the total number of events. Generally speaking, two events with a causal relationship will occur within a certain time range, and the closer the two events are in time, the more likely they are to have a causal relationship. This also means that when the time interval between two events is too far apart, it is unlikely that there is a strong causal relationship between the two. Therefore, we have chosen a sliding time window length of l, $l \in o$, $o \notin 0$. In order to more accurately explore the causal relationship between events, we set the sliding window length to 5, associate each current event in the 5-length window log event with subsequent events, calculate their transition entropy, and determine whether event e_1 and event e_2 have a causal relationship.

To distinguish direct causal relationships between events and exclude indirect causal relationships that are difficult to track and analyze due to multiple confounding variables, we propose an improved method for measuring causal strength called "standardized causal entropy," defined as follows:

$$SC_{J \to i|N} = \frac{I\left(X_i^t; X_J^{t-1:t-l} \mid X_N^{t-1:t-l}\right)}{h\left(X_i^t\right)}. \tag{2}$$

Among them, under condition N, the standardized causal entropy of node set J for node i. Among them, X_i^t

represents the value of node i at time t, while $X_J^{t-1:t-l}$ and $X_N^{t-1:t-l}$ respectively represent the values of node set J and condition set N within the first 1 time t.

The causal strength is determined by the amount of information transmitted between nodes, and two types of information transmission are distinguished here: direct causal relationship $(e_1 \to e_2 \to e_3)$ and indirect causal relationship $(e_1 \to e_3)$. In the analysis of causal relationships, the true strength of causality is mainly determined by direct causal relationships. This also means that for any two events, the strength of their causal relationship should be determined based on the direct amount of information transmitted between them, rather than those indirectly transmitted through other nodes or events. This distinction helps to more accurately identify and understand the actual correlations between events, providing more precise insights for data analysis.

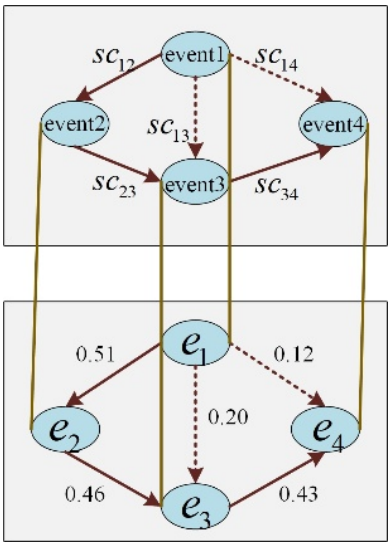

Fig 2. Standardized causal entropy between partial events.

Therefore, we measure the direct causal relationship between events by standardizing causal entropy. Compared to transfer entropy, it incorporates more nodes to form a condition set, effectively eliminating indirect effects. As shown in Figure 2, measuring the causal strength between e_1 and e_2, the transfer entropy is $I\left(X_{e_3}^t; X_{e_1}^{t-1:t-l} \mid X_{e_3}^{t-1:t-l}\right)$, which cannot exclude the indirect influence transmitted through e_2, leading to redundancy. However, the normalized standard entropy is $I\left(X_{e_3}^t; X_{e_1}^{t-1:t-l} \mid X_{e_2}^{t-1:t-l}\right)$, which excludes the indirect influence by using e_2 as the condition set. Meanwhile, according to the theory of Granger causality (the past influences the future), since $X_{e_1}^{t-1:t-l}$ occurred before $X_{e_3}^t$, $I\left(X_{e_3}^t; X_{e_1}^{t-1:t-l} \mid X_{e_2}^{t-1:t-l}\right)$ must be the information flow transmitted from $X_{e_1}^{t-1:t-l}$ to $X_{e_3}^t$. Specifically, when $e_2 = pa_i - \{e_1\}$, $I\left(X_{e_3}^t; X_{e_1}^{t-1:t-l} \mid X_{e_2}^{t-1:t-l}\right)$ represents the direct information flow from e_1 to e_3, excluding all indirect influences. he normalized standard entropy between $e_1 e_3$ takes the parent e_3 of e_2 as the condition set, excludes the indirect influence transmitted through e_2 between $e_1 e_3$, and measures the causal strength between them using normalized standard entropy to obtain $SC_{e_1 \to e_2} > SC_{e_1 \to e_3 | e_2}$. Therefore, the causal relationship between $e_1 \to e_3$ is excluded. It can be seen that normalized standard entropy can effectively eliminate indirect influence, accurately measure the direct causal relationship between nodes, and eliminate redundancy caused by indirect influence. Through this method, we can more accurately and effectively analyze and understand causal relationships in complex systems, providing conditions for subsequent identification of events.

3.3. Generative Adversarial Learning

Generative Adversarial Networks (GANs), as a dynamic game model, comprise two adversarial components: the generator G and the discriminator D. The generator network takes random noise as input and generates data samples. It receives input z from a probability distribution $p(z)$ and produces generated sample data for the discriminator D.The discriminator, in turn, takes real and generated data as input and predicts and discriminates between the two types of input data.

The optimization objective of Generative Adversarial Networks is:

$$\min_G \max_D V(D,G) = \mathrm{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathrm{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{3}$$

In this equation, $V(D,G)$ represents the value function of the Generative Adversarial Network, $D(x)$ signifies the probability that the discriminator perceives real data as genuine, $G(z)$ denotes the samples generated by the generator from the noise z, $D(G(z))$ signifies the probability that the discriminator views samples generated by the generator as real data, p_{data} represents the distribution of real data samples, and p_z represents the distribution of samples in the latent space. equation 3 can also be expressed as:

$$\min_G \max_D V(D,G) = \mathrm{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathrm{E}_{\tilde{x} \sim p_g}[\log(1 - D(\tilde{x}))] \tag{4}$$

Among them, $G(z) = \tilde{x}$, p_g represents the distribution of sample $\tilde{x}$ generated by the generator. On this basis, calculate the ideal discriminator D^* and the ideal generator G^*.

$$D^* = \arg\max_D V(D,G) \tag{5}$$

$$G^* = \arg\min_G \max_D V(D,G) = \arg\min_G V(D^*,G) \tag{6}$$

For the above D^*, assuming that generator G is fixed and has $G(z)=\tilde{x}, \tilde{x} \sim p_g$, then:

$$\begin{aligned} V(D,G) &= E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p_z}[\log(1-D(G(z)))] \\ &= \int \left(p_{data}(x)\log D(x) + p_g(x)\log(1-D(x)) \right) dx \end{aligned} \tag{7}$$

Now we need to find a D that maximizes V, that is, we hope to maximize the value of x for the integrand function $f(x) = p_{data}\log D(x) + p_g \log(1-D(x))$ regardless of its value. Among them, p_{data} is fixed, and assuming the generator is fixed, so p_g is also fixed. If the derivative of $f(x)$ over $D(x)$ is equal to zero, then D^* can be calculated to maximize $f(x)$. Namely:

$$\frac{df(x)}{D(x)} = \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0 \tag{8}$$

Obtain:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{9}$$

From equation 9, it can be seen that its value is between 0 and 1, which conforms to the pattern of the discriminator. The ideal discriminator judges the real data as 1 and the generated data as 0. When the distribution of the generated data by the generator is very close to the distribution of the real data, the output probability of the discriminator is 0.5.

According to the definitions of KL divergence and JS divergence:

$$JSD(P\|Q) = \frac{1}{2}KL(P\|M) + \frac{1}{2}KL(Q\|M) \tag{10}$$

Where $M = \frac{1}{2}(P+Q)$, so equation 10 can be expressed as:

$$\max_D V(G,D) = -\log 4 + KL\left(p_{data} \| \frac{p_{data}+p_g}{2} \right) + KL\left(p_g \| \frac{p_{data}+p_g}{2} \right) \tag{11}$$

Due to the non negative divergence of JS, equation 11 can obtain the global minimum value $-\log 4$ only when $p_{data} = p_g$. Therefore, it can be inferred that the generator is optimal when the distribution of the generated data is equal to the actual distribution of the data.

3.4. Generator Model

In this section of the research, a generative model was constructed using a network architecture that integrates GRU and LSTM with a self-attention mechanism, as depicted in Figure 3. This generative network is designed to produce fake samples that are highly similar to real samples. Broadly speaking, the generative model encompasses three major components: an embedding layer, a self-attention layer, and a feed-forward network layer.

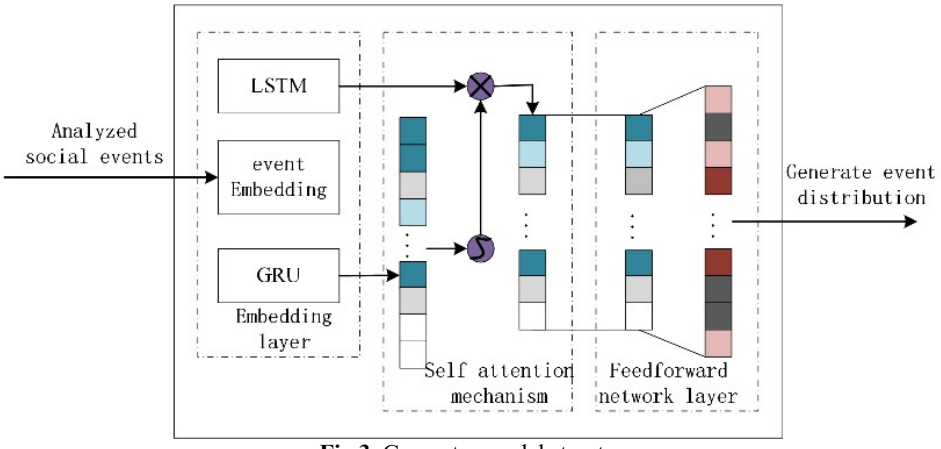

Fig 3. Generator model structure.

The embedding layer includes an event embedding module, GRU units, and LSTM units. Assuming a sequence $A = \{a_1, a_2, ..., a_m\}$ is obtained after parsing social stream logs, where m represents the total number of log events, and each event is assigned a unique identifier upon log parsing, implying that all events within a can be labeled. This study employs the widely used Word2Vec technology in NLP to convert each event into a continuous vector form. Events are presented in a serialized pattern and, following embedding

transformation, are sequentially fed into the LSTM and GRU units. Upon inputting the serialized event pattern into the LSTM and GRU, we obtain two continuous one-dimensional vectors, h_{LSTM} and h_{GRU}, which represent the output of the embedding layer and are further transmitted to the self-attention mechanism layer.

The key component of LSTM is its storage unit, which consists of a series of parameters and gate control systems that collectively determine the storage or disposal of information. As shown in Figure 4, each LSTM storage unit contains three types of gates: input gates, forget gates, and output gates. The LSTM layer takes the word embeddings of each log event obtained from the word embedding layer as input. In an LSTM unit, the output contains both unit states and hidden states, which are passed from one LSTM unit to the next, and the hidden states are also fed into the LSTM unit of the previous layer as input. At each time step, LSTM processes the current input x_t and the previously stored unit state h_{t-1} to calculate the new unit state. x_t represents event word embedding, C_t represents the state of a certain LSTM unit in the current sequence, represents the hidden state of the t-th LSTM in the same sequence, σ represents the nonlinear activation function, and the derivation of the output of the LSTM unit is shown in equation 12-16.

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \tag{12}$$

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \tag{13}$$

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right) \tag{14}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{15}$$

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{16}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{17}$$

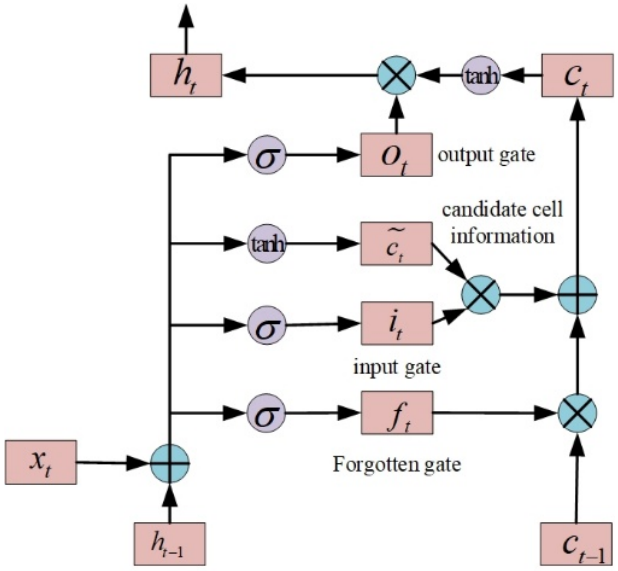

Fig 4. LSTM structure

In equation 12, f_t represents the forget gate, and the amount of information removed from the unit state by this gating mechanism is influenced by the word embeddings in the current time series at time t and the output of the previous unit h_{t-1}. equation 12 and 14 involve the operation of input gates, and the new candidate value vector $\tilde{c}_t$ is generated and added through the operation of the σ function and $\tanh$ layer. Multiply the old cell state c_{t-1} by f_t to determine the information to be discarded. equation 15 and 16 explain the mechanism of the output gate, starting with the σ function to determine which parts of information will be output, and then $\tanh$ processes the updated units to generate the final output. This process is repeated in multiple time steps until the final hidden state output is generated.

The main purpose of the self-attention mechanism layer is to further process the output of the embedding module. By learning the correlation and importance between different parts of the text data, the model's attention to key information is improved, which helps to generate models that more accurately capture abnormal patterns and features in log events. represented by equation 18 to 20:

$$out_i = \tanh\left(W_w h_i + b_a\right) \tag{18}$$

$$\alpha_i = \frac{\exp\left(out_i^T u_w\right)}{\sum \exp\left(out_i^T u_w\right)} \tag{19}$$

$$s = \sum\nolimits_n out_i \alpha_i \tag{20}$$

Among them, W_w and b_a represent weights and bias terms, respectively. Each log h_i is encoded as a key value vector out_i, and a random vector out_i is initialized to represent the features of the entire sequence. The importance of each log in the sequence to the entire sequence is calculated α_i, and finally, weighted sum is performed to obtain the vector representation s of the entire sequence. This article uses LSTM and GRU networks to fit the serialized event patterns. The output h_{LSTM} of LSTM serves as a feature transmitted to the feedforward network layer, while the output h_{GRU} of GRU acts as a weight of the attention mechanism on h_{LSTM} to fine tune it in various dimensions. The two are cascaded through equation 21:

$$o_a = soft\max\left(h_{GRU} w_a + b_a\right) \times h_{LSTM} \tag{21}$$

Among them, o_a is the output of the attention mechanism, and w_a and b_a are the weights and bias terms.

Subsequently, the output of the self-attention mechanism layer is fed to the feedforward network layer. In this article, the output of the feedforward network layer is an m-dimensional one-dimensional continuous vector o_g, which means the distribution of each event in the event set E as a generated event in the current event mode. The formula is:

$$o_g = \tanh\left(\mathrm{Re}\,LU\left(o_a w_f + b_f\right) \times w_o + b_a\right) \tag{22}$$

Among them, w_f, w_o, b_f, and b_a are the weights and bias terms in the feedforward network layer, respectively.

3.5. Discriminator Model

In this paper, the primary task of the discriminative model is to calculate the probability values of input events using its

fully connected layers. The determination of normality or anomaly is made by jointly utilizing the reconstruction loss values obtained from the generator and the generative loss values acquired from the discriminator, as illustrated in Figure 5.

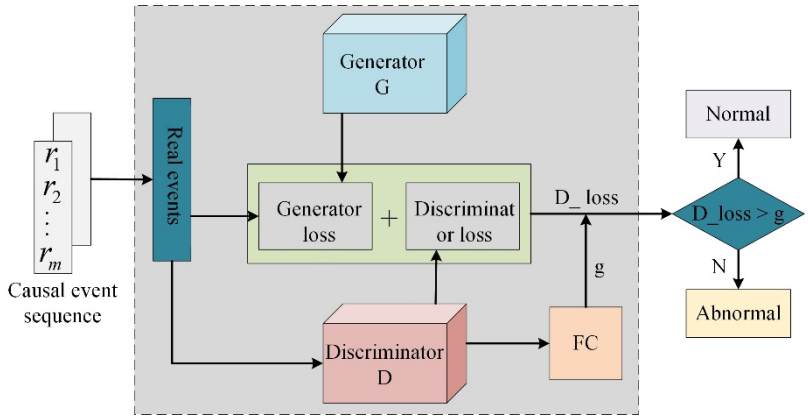

Fig 5. Discriminator model architecture.

As shown in equation 23, when the loss values of both exceed a predetermined threshold, they are considered anomalous; otherwise, they are deemed normal. The identification process follows the steps below:

$$\mathbf{o}_g = \tanh\left(\mathrm{ReLU}\left(\mathbf{o}_a \mathbf{W}_f + \mathbf{b}_f\right)\cdot \mathbf{w}_o + \mathbf{b}_a\right) \tag{23}$$

The initial phase involves setting parameters for both the generator G and discriminator D, where the generator combines events produced by the feed-forward network layer with actual events, which are then collectively input into the discriminator. Upon receiving feedback from the discriminator's fully connected layer, parameters for both the generator and discriminator are updated. The generator is trained to blur the distinction between real and generated events as much as possible, while keeping the discriminator's parameters unchanged. After a single update to the discriminator, multiple rounds of updates are applied to the generator. Through several iterations of optimization, the ideal goal is that the discriminator becomes unable to distinguish between normal events generated by the machine or those occurring naturally, thus concluding the training process.

Based on the description of the detection process, the objective for the discriminator's detection is to sensitively judge normal or anomalous events. The ultimate goal is for all normal events to be output as 1; for all anomalous events, the output should be 0. This criterion is measured using the fully connected layer and its average cross-entropy. Real event labels are denoted as 1, while events generated by the generator are denoted as 0. For any real event sequence R_{real} ,the average cross-entropy between the discriminator's judgment results for the real event sequence and 1 is calculated, as shown in equation 24. For the false event sequences o_g generated by the generator, the average cross-entropy between the discriminator's judgment results for the false event sequence and 0 is calculated, as indicated in equation 25. Averaging these two types of loss is due to the discriminator outputting a loss value for each event. The total loss of the discriminator is the sum of these two losses, as presented in equation 26.

$$D_{loss\_normal} = Ave\left(sigmoid\_cross\_entropy\left(D\left(R_{real}\right),1\right)\right) \tag{24}$$

$$D_{loss\_abnormal} = Ave\left(sigmoid\_cross\_entropy\left(D\left(o_g\right),0\right)\right) \tag{25}$$

$$D_{loss} = D_{loss\_normal} + D_{loss\_abnormal} \tag{26}$$

For equation 24, the $sigmoid\_cross\_entropy(\ )$ unction indicates that the output $D\left(R_{real}\right)$ of the discriminator is first activated by the $sigmoid$ function to ensure its value is between 0 and 1. Then, the cross entropy between the activated value and 1 is calculated, and D_{loss_normal} represents the loss of discriminating the true event as 1; For equation 25, the $sigmoid\_cross\_entropy(\ )$ function indicates that the output $D\left(o_g\right)$ of the discriminator is first activated to ensure its value is between 0 and 1, and then the cross entropy between the activated value and 0 is calculated. Among them, $Ave(\ )$ represents the average function, $D_{loss\_abnormal}$ represents the loss of discriminator's output discrimination as 0, and D_{loss} represents the overall loss of discriminator.

In each round of detection, a true subsequence is input into the discriminator, and the generator simultaneously generates a false event subsequence and inputs it into the discriminator for judgment. Therefore, equation 26 can be expressed as:

$$D_{loss} = -\frac{1}{n}\left(\sum \log\left(sigmoid\left(D\left(R_{real}\right)\right)\right) + \sum \log\left(1 - sigmoid\left(D\left(o_g\right)\right)\right)\right) \tag{27}$$

This is consistent with equation 3, with the only difference being that it is equal to taking a negative value for the loss function of the original GAN, corresponding to minimizing the loss function of the discriminator.

The training of the generator aims to successfully deceive the discriminator into classifying the generated output as real, that is, the generator aims to minimize the cross entropy between the discriminator's prediction of generated events and the labels of real events.

$$G_{loss} = Ave\left(sigmoid\_cross\_entropy\left(D\left(o_g\right),1\right)\right) \tag{28}$$

In each training round, i.e

$$G_{loss} = -\frac{1}{batch\_size}\sum \log\left(sigmoid\left(D\left(o_g\right)\right)\right) \tag{29}$$

Using the random gradient descent algorithm to update parameters for the discriminator and the Adam adaptive learning rate optimization algorithm for the generator can overcome the problem of gradient sparsity during the model update process, improve convergence speed, and to some extent shorten detection time.

4. Results and Discussion

In this section, we will first outline the evaluation metrics employed in the experiments and the datasets utilized. Subsequently, a detailed comparison of the performance of our proposed method on social network datasets with that of existing methods will be presented. The experimental platform was based on Python 3.9, and all experiments were conducted on a single computer equipped with an Intel(R) Core(TM) i7 processor (2.9GHz), 16GB of RAM, and the Windows 10 operating system. In addition, various parameters within the model were examined. The default values of the parameters used in this study are presented in Table 2.

Table 2. Experimental parameter values

Hyperparameter	Value
Learning rate	0.001
Iterations	300
Number of LSTM network layers	2
Sliding window size	5
Set threshold	0.60

4.1. Evaluation Metrics

Precision, recall, and F1 score were selected as the evaluation metrics. These three metrics are derived from the confusion matrix, which categorizes results into four types: True Positives (TP) are instances correctly identified as positive. False Positives (FP) refer to negative instances incorrectly identified as positive. True Negatives (TN) are instances correctly identified as negative. Finally, False Negatives (FN) refer to positive instances incorrectly identified as negative.

Precision is calculated as the percentage of correctly predicted positive samples out of all predicted positive samples. Recall is calculated as the percentage of correctly predicted positive samples out of all actual positive samples. The F1 score is a metric that calculates the harmonic mean of precision and recall.

$$\Pr ecision = \frac{TP}{TP + FP} \tag{30}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{31}$$

$$F1 - Score = \frac{2 * \Pr ecision * \mathrm{Re}call}{\Pr ecision + \mathrm{Re}call} \tag{32}$$

4.2. Description of Datasets

Experiments were conducted on two datasets: the HDFS dataset and the UNSW-NB15 dataset. Below are the detailed descriptions of these datasets:

1. HDFS Dataset [26]: Comprising 11,175,629 log entries collected from over 200 Amazon EC2 platforms, the HDFS dataset was manually annotated through rules rafted to identify anomalies. It contains 575,061 log blocks, out of which 16,838 blocks are marked as anomalous.

2. UNSW-NB15 Dataset [27]: This dataset captures millions of records from digital social media and other sources for predicting, tracking, and responding to global events. It is primarily used to generate mixed data of modern normal activities and synthetic contemporary behavior attacks, totaling 2,540,044 data logs.

4.3. Baseline Method

Our proposed method was compared with four baseline approaches, summarized as follows:

MABED [28]: A statistical method for log event detection based on mention anomaly analysis, MABED relies solely on tweets and utilizes the frequency of dynamically inserted links (mentions) in tweets to detect significant events and estimate their impact on the population. Its uniqueness lies in dynamically estimating the time frame for discussing each event, instead of assuming a predefined fixed duration for all events.

RL-LDA [29]: This method effectively captures user intelligence behaviors and supports user decision-making. It employs programs that can perceive the environment, understand the relevance of complex events, and learn how to act in crucial situations. RL-LDA uses forwarding behavior for event evolution detection, capturing social media information through tags, locations, texts, and forwarding actions.

DeepSVDD [30]: A semi-supervised anomaly detection approach that uses neural networks to learn features that map normal sequences onto a hypersphere and map anomalous sequences far from the hypersphere.

LogRobust [31]: LogRobust extracts semantic information from log events and represents it as semantic vectors. It leverages an attention-based dual LSTM model to detect anomalous log sequences.

4.4. Comparative Experimental Analysis

In this section, we conducted a detailed comparative analysis, evaluating our proposed method, GANAED, against four baseline methods MABED, RL-LDA, DeepSVDD, and LogRobust across two distinct datasets.

As depicted in Figure 6, comparison results on the HDFS dataset indicate that both our method, GANAED, and LogRobust, outperform the other methods in terms of Precision, Recall, and F1-measure, with the most notable improvements observed in Precision. Taking LogRobust as an example, it exhibited a 26.3% and 17% increase in Precision over RL-LDA and DeepSVDD, respectively. This significant improvement is primarily attributed to the LogRobust model's ability not just to focus on the log events themselves but to effectively capture the contextual semantic information of events, which is often overlooked in traditional anomaly detection methods. Therefore, compared to other methods, GANAED and LogRobust demonstrate a stronger capacity for anomaly detection. Furthermore, it is evident from the graph that our GANAED achieves a 6.5% and 1.1% increase in Precision and F1 score, respectively, over LogRobust. This progress is due to the innovative strategies employed in GANAED: we utilize a generative model based on self-attention mechanism integrated GRU and LSTM networks, with a particular emphasis on introducing GRU to ensure its output effectively complements the self-attention mechanism. This approach allows log events input at different time points to be assigned varying weights, more accurately revealing causal relationships between events, thereby enhancing the generalization capability of the generative model and effectively improving the detection of anomalous events.

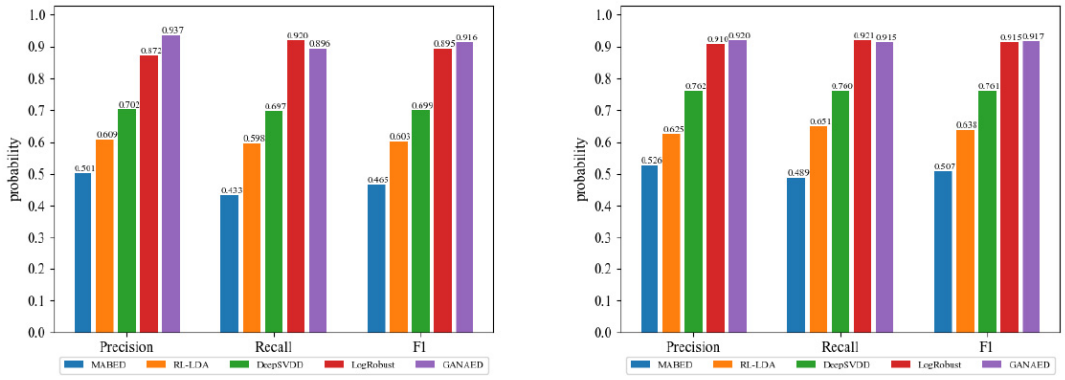

Fig 6. (a) Evaluation on the HDFS dataset. (b) Evaluation on the UNSW-NB15 dataset

Figure 6 presents a detailed comparison of our method, GANAED, with the other four methods DeepSVDD, LogRobust, MABED, and RL-LDA on the UNSW-NB15 dataset in terms of Precision, Recall, and F1-Score. It is clear from the charts that GANAED exhibits significant improvements across all evaluation metrics compared to DeepSVDD and LogRobust, particularly in Precision, where it outperforms DeepSVDD and LogRobust by 15.8% and 3.0%, respectively. This notable performance boost is credited to GANAED consideration of the potential causal relationships between social network events. It not only accurately captures the temporal dependencies present in social network logs but also delves into the complex contextual information. In contrast, both MABED and RL-LDA show performances below 65% in F1-Score, primarily because these methods focus on preserving the main features of original events, thereby overlooking essential information, leading to the omission of many actual anomalies and affecting the accuracy and reliability of the results.

4.5. Analysis of Model Performance by Parameters

To further optimize and tune the performance of the model, this subsection will analyze the impact of certain parameters on GANAED regarding four metrics: Precision, Recall, F1-Score, and D_loss. The default values of all parameters used in the experiments are provided in Table 2. The datasets selected for this experiment are the HDFS and UNSW-NB15 datasets, with the results presented in Figures 7, respectively.

First, the impact of the sliding window size on the model's evaluation metrics is considered. Observation of the trends in graphs (a) and (b) reveals that, as the window size gradually increases, the model's anomaly detection precision improves, while the recall rate decreases to some extent. However, the overall performance of the model, as indicated by the F1 score, shows gradual improvement, peaking at a window size of 5. Continuing to increase the window size leads to a noticeable decline in recall rate and a gradual downward trend in overall model performance. Given the longer detection time required by larger window sizes, the optimal window size is determined to be 5, balancing the model's performance across different window sizes. Next, the impact of the LSTM network layer parameter on the detection model is explored through graphs (c) and (d), which show the influence of varying network depth on the model's evaluation metrics. It is observed that the LSTM network layers have minimal impact on the three metrics, with optimal performance occurring at two layers. Subsequent increases in layers lead to a gradual performance decline, attributed to the complication of having too many parameters affecting the model's performance, resulting in diminishing returns. Graphs (e) and (f) examine the effect of different thresholds on the discriminator network's fully connected layer for correctly identifying real events as 1, misidentifying fake events as 0, and the overall loss. The results show that when the threshold is less than 0.6, the loss for correctly and incorrectly identified events, as well as the total loss, gradually increases. When the threshold exceeds 0.6, the model's loss begins to decrease. A too-low threshold results in a high false-positive rate, while a too-high threshold lowers the detection rate. The optimal anomaly

detection performance is achieved at a threshold of 0.6.

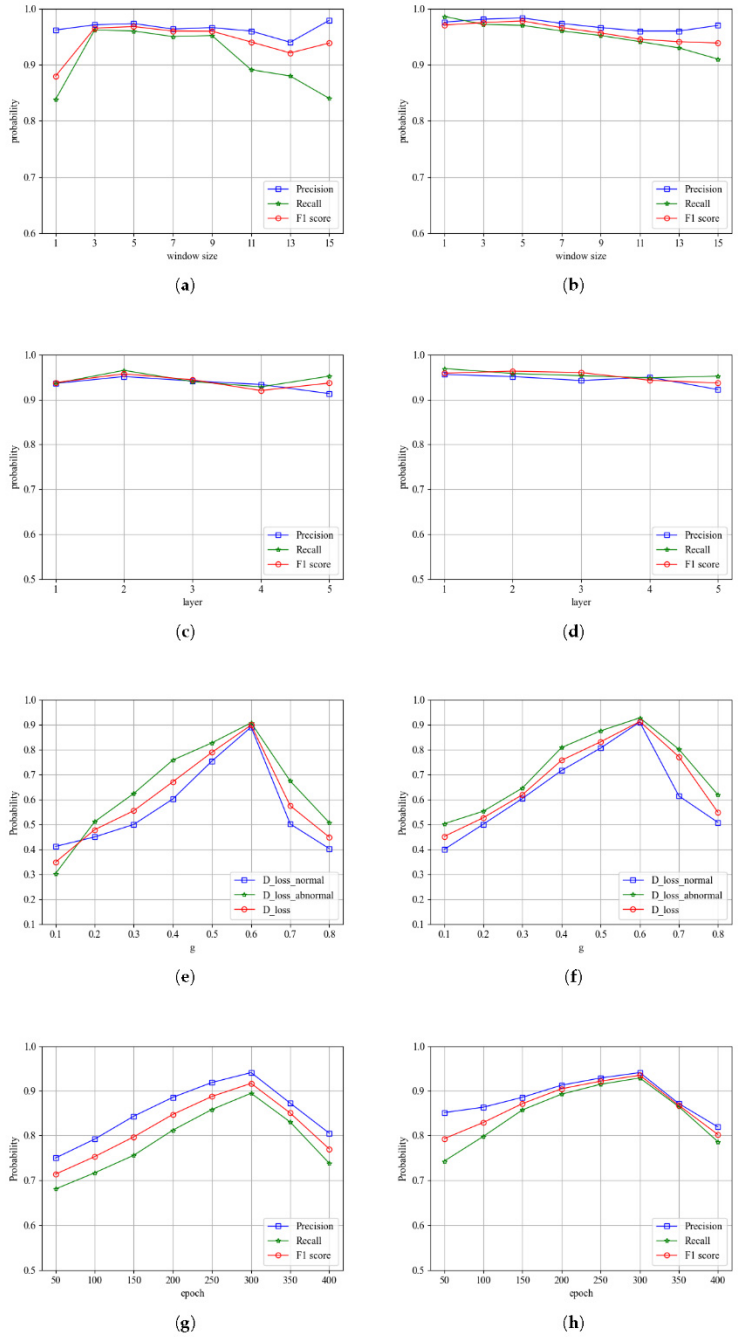

Fig 7. Standardized causal entropy between partial events.

Finally, graphs (g) and (h) depict changes in model performance with varying training epochs. Within a certain range of training iterations, performance improves with an increase in the number of iterations. However, beyond a certain point, performance reaches a saturation point. Therefore, it is necessary to balance the choice of iteration numbers to avoid performance degradation.

4.6. Ablation Study

Table 3. HDFS ablation results

Model	Precision	Recall	F1-Score
w/o SC	43.7%	75.2%	55.3%
w/o Sel-AttGRU	86.8%	86.8%	81.4%
w/o LSTM	87.3%	81.2%	84.1%
GANAED	93.7%	89.6%	91.6%

To further explore the specific impact of key technologies in GANAED on model performance, we assessed the performance of the model without normalized causal entropy, GRU lacking the self-attention mechanism, and excluding the LSTM network. According to the data in Tables 3 and 4, models lacking normalized causal entropy exhibited lower

accuracy and F1-Score on both HDFS and UNSW-NB15 datasets compared to the complete GANAED model. This phenomenon indicates that the inclusion of normalized causal entropy plays a crucial role in uncovering causal connections between log events and tracking event propagation trends.

Table 4. UNSW-NB15 ablation results

Model	Precision	Recall	F1-Score
w/o SC	50.2%	74.1%	59.9%
w/o Sel-AttGRU	84.7%	79.0%	81.8%
w/o LSTM	89.7%	79.2%	84.1%
GANAED	92.0%	91.5%	91.7%

Additionally, we found that models without the self-attention mechanism GRU, especially on the UNSW-NB15 dataset, performed worse in terms of accuracy and recall compared to the complete GANAED. On the HDFS dataset, while the GRU model without self-attention mechanism achieved higher accuracy, its recall rate was significantly lower than that of the complete model, leading to a decrease in overall F1-Score. These results suggest that the introduction of the self-attention mechanism, by focusing more on key topics or words and reducing semantic differences between log texts, is crucial for enhancing model performance. Lastly, when testing models without the LSTM network on both datasets, we observed a decline in accuracy of 6.4% and 4.9% respectively compared to the complete GANAED. This outcome highlights the importance of the LSTM network in capturing temporal dependencies and event context, enabling GANAED to more comprehensively judge the normality or anomaly of events, thereby effectively improving the efficiency and accuracy of anomaly detection.

5. Conclusion

This paper presents causal entropy–driven generative adversarial learning for anomalous event detection in social networks, a framework that integrates log parsing, causal relationship mining, and adversarial learning to address the challenges of dynamic environments and complex inter-event dependencies. By converting unstructured logs into structured events, employing normalized causal entropy to capture causal strengths, and designing a generative adversarial architecture that combines self-attention with GRU and LSTM for semantic modeling, the proposed method achieves approximately 6.5% higher detection accuracy than baseline approaches, demonstrating both effectiveness and robustness. Looking ahead, since anomalous events in social networks often extend beyond textual logs to multimodal content such as images, videos, and audio, future research could focus on extending this framework to heterogeneous data sources for more comprehensive and reliable anomaly detection.

References

[1] K. Z. Khanam, G. Srivastava, and V. Mago, ‘‘The homophily principle in social network analysis: A survey,’’ Multimedia Tools and Applications, vol. 82, no. 6, pp. 8811–8854, 2023.

[2] S. R. Sahoo and B. B. Gupta, ‘‘Multiple features based approach for automatic fake news detection on social networks using deep learning,’’ Applied Soft Computing, vol. 100, p. 106983, 2021.

[3] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, ‘‘A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects,’’ Engineering Applications of Artificial Intelligence, vol. 110, p. 104743, 2022.

[4] A. Abdelrazek, Y. Eid, E. Gawish, W. Medhat, and A. Hassan, ‘‘Topic modeling algorithms and applications: A survey,’’ Information Systems, vol. 112, p. 102131, 2023.

[5] M. R. Islam, S. Liu, X. Wang, and G. Xu, ‘‘Deep learning for misinformation detection on online social networks: a survey and new perspectives,’’ Social Network Analysis and Mining, vol. 10, no. 1, p. 82, 2020.

[6] G. Nagarajan, R. Minu, and A. Jayanthila Devi, ‘‘Optimal nonparametric bayesian model-based multimodal bovw creation using multilayer plsa,’’ Circuits, Systems, and Signal Processing, vol. 39, no. 2, pp. 1123–1132, 2020.

[7] H. Jelodar, Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, and L. Zhao, “Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey,” Multimedia tools and applications, vol. 78, pp. 15169–15211, 2019.

[8] H. Zhou, H. Yin, H. Zheng, and Y. Li, ‘‘A survey on multi-modal social event detection,’’ Knowledge-Based Systems, vol. 195, p. 105695, 2020.

[9] L. Zhang, B. Wu, and P. Dong, ‘‘Attribute graph anomaly detection utilizing memory networks enhanced by multi-embedding comparison,’’ Neurocomputing, vol. 633, p. 129762, 2025.

[10] S. Tabassum, F. S. Pereira, S. Fernandes, and J. Gama, ‘‘Social network analysis: An overview,’’ Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 8, no. 5, p. e1256, 2018.

[11] G. Li and J. J. Jung, ‘‘Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges,’’ Information Fusion, vol. 91, pp. 93–102, 2023.

[12] Y. Zuo, J. Zhao, and K. Xu, ‘‘Word network topic model: a simple but general solution for short and imbalanced texts,’’ Knowledge and Information Systems, vol. 48, pp. 379–398, 2016.

[13] J. Zhang, M. Liu, and Y. Zhang, ‘‘Topic-informed neural approach for biomedical event extraction,’’ Artificial intelligence in medicine, vol. 103, p. 101783, 2020.

[14] G. Stilo and P. Velardi, ‘‘Efficient temporal mining of micro-blog texts and its application to event discovery,’’ Data Mining and Knowledge Discovery, vol. 30, no. 2, pp. 372–402, 2016.

[15] K. Bountrogiannis, G. Tzagkarakis, and P. Tsakalides, ‘‘Distribution agnostic symbolic representations for time series dimensionality reduction and online anomaly detection,’’ IEEE Transactions on Knowledge and Data Engineering, 2022.

[16] S. Li, H. Ji, and J. Han, ‘‘Document-level event argument extraction by conditional generation,’’ arXiv preprint arXiv:2104.05919, 2021.

[17] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, ‘‘Topicsketch: Real-time bursty topic detection from twitter,’’ IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 8, pp. 2216–2229, 2019.

[18] Q. Diao, J. Jiang, F. Zhu, and E. P. LIM, ‘‘Finding bursty topics from microblogs.’’ Acl, 2012.

[19] R. Wang, D. Zhou, and Y. He, ‘‘Atm: Adversarial-neural topic model,’’ Information Processing & Management, vol. 56, no. 6, p. 102098, 2019.

[20] A. P. Logan, P. M. LaCasse, and B. J. Lunday, ‘‘Social network analysis of twitter interactions: a directed multilayer network approach,’’ Social Network Analysis and Mining, vol. 13, no. 1, p. 65, 2023.

[21] G. Pu, L. Wang, J. Shen, and F. Dong, ‘‘A hybrid unsupervised clustering-based anomaly detection method,’’ Tsinghua Science and Technology, vol. 26, no. 2, pp. 146–153, 2020.

[22] L. Yan, C. Luo, and R. Shao, ‘‘Discrete log anomaly detection: a novel time-aware graph-based link prediction approach,’’ Information Sciences, vol. 647, p. 119576, 2023.

[23] E. O. Omuya, G. O. Okeyo, and M. W. Kimwele, ‘‘Feature selection for classification using principal component analysis and information gain,’’ Expert Systems with Applications, vol. 174, p. 114765, 2021.

[24] M. Gnouma, R. Ejbali, and M. Zaied, ‘‘Deep hashing and sparse representation of abnormal events detection,’’ The Computer Journal, vol. 67, no. 1, pp. 3–17, 2024.

[25] J. Sun, D. Taylor, and E. M. Bollt, ‘‘Causal network inference by optimal causation entropy,’’ SIAM Journal on Applied Dynamical Systems, vol. 14, no. 1, pp. 73–106, 2015.

[26] A. El Abdouli, A. Chaffai, L. Hassouni, H. Anoun, and K. Rifi, ‘‘Current morroccan trends in social networks,’’ International Journal of Computer Science and Information Security, vol. 14, no. 3, pp. 86–98, 2016.

[27] C. Wang, Y.-M. Zhang, and C.-L. Liu, ‘‘Anomaly detection via minimum likelihood generative adversarial networks,’’ in 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, 2018, pp. 1121–1126.

[28] A. Guille and C. Favre, ‘‘Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach,’’ Social Network Analysis and Mining, vol. 5, no. 1, p. 18, 2015.

[29] P. Y. Erfanian, B. R. Cami, and H. Hassanpour, ‘‘An evolutionary event detection model using the matrix decomposition oriented dirichlet process,’’ Expert Systems with Applications, vol. 189, p. 116086, 2022.

[30] A. Borg and M. Boldt, ‘‘Using vader sentiment and svm for predicting customer response sentiment,’’ Expert Systems with Applications, vol. 162, p. 113746, 2020.

[31] X. Chen, C. Cao, and J. Mai, ‘‘Network anomaly detection based on deep support vector data description,’’ in 2020 5th IEEE International Conference on Big Data Analytics (ICBDA). IEEE, 2020, pp. 251–255.