International Journal of Safety and Security Engineering
Vol. 15, No. 8, August, 2025, pp. 1565-1575
Journal homepage: http://iieta.org/journals/ijsse

Spark-Based Contextual Embeddings for Automated Feature Extraction in Intrusion Detection

R. Panneerselvi*, J. Visumathi

Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai 600062, India

Corresponding Author Email: panneerselvir@veltech.edu.in

https://doi.org/10.18280/ijsse.150803

Received: 15 April 2025
Revised: 25 May 2025
Accepted: 8 July 2025
Available online: 31 August 2025

Keywords:
Word2Vec, latent semantic indexing, Doc2Vec, BERT, natural language processing

ABSTRACT

The increasing requirement for the fast and accurate detection of abnormal network traffic has led to an increase in the popularity of automated intrusion detection systems (IDS). Despite advancements in machine learning (ML) and deep learning (DL) for anomaly detection, many IDS models rely on handcrafted features. This reliance results in the feature drift phenomenon, and the old handcrafted features will become outdated to accommodate newer attack patterns, diminishing detection performance. Furthermore, if traditional ML or DL methods frequently involve computing with large amounts of network traffic which is computationally expensive and does not allow real-time performance. To address these limitations, this paper presents an IDS pipeline that applies natural language processing (NLP) techniques to construct a new surface of automatic feature extraction from network data. This mitigates feature drift and maintains the ability to adapt to evolving behaviours in maliciousness. The pipeline combines an attention mechanism with NLP models, including Word2Vec, program vector, and latent semantic indexing (LSI) to produce powerful hybrid feature vectors. The BERT-based classifier is trained with these vectors, and its performance in terms of accuracy, precision, recall, and F1-score is evaluated. Experimental results demonstrate the superior performance of the proposed approach over previous methods, achieving an F1-score of 0.99 with automatically extracted features. Integrating Apache Spark's processing power makes our system fast enough to be good in case of real-time intrusion detection, and also scalable.

1. INTRODUCTION

Smart cities, smart houses, etc., have made our lives more intelligent. One of the main requirements for being intelligent is being connected, and most of the items we use daily are linked thanks to networks, which makes life easier. The shift in technology is a good thing, but increased connection also makes hackers more likely to target security features like availability, confidentiality, and integrity. Understanding the locations of incoming and outgoing traffic on any network is crucial for handling different kinds of security assaults, and this is often accomplished through the use of an intrusion detection system (IDS). Though the current IDS uses machine learning (ML)/deep learning (DL)-based algorithms for segregating normal and abnormal traffic, one major concern with them is their inability to understand the new forms of attack, resulting in a decrease in detection accuracy and an increase in the number of false alarms [1].

1.1 Challenges

Most of the ML models used for improving the detection accuracy of the classification model are dependent on the features that are used for classification. As a result, feature reduction techniques are used in ML techniques [2] to increase detection accuracy. Although it functions well for the current samples, they encounter problems in terms of generalization when the features. Furthermore, domain expertise plays a crucial role in determining which traits are optimal, necessitating manual dealing with adversarial samples [3], as the detection accuracy is largely dependent on the optimal selection of involvement in this technique [4]. Because of the feature drift issue [5], these manually extracted features are more prone to becoming out of date and becoming targets for assaults.

In contrast to ML, DL algorithms can automatically extract packets from raw packets without the need for feature engineering [6]. A few studies employed CNN and LSTM networks with natural language processing (NLP) techniques

to automate feature extraction using DL-based approaches [7]. While the use of NLP in IDS is not new, it is currently limited to hand-crafted feature datasets such as NSL-KDD, Trace, and others that are prone to feature drift.

1.2 Motivation

Understanding the viability of sophisticated NLP approaches for feature extraction from raw data packets rather than analyzing hand-crafted feature datasets is the driving force behind this project, which is motivated by the difficulties encountered with ML and DL methods. This concept is covered in the study [8], where notable results on the DARPA 2009 dataset are obtained by processing the raw packets to produce vectors used for categorization. The proposed strategy is expected to perform well even in the presence of newer types of traffic data that are not included in the dataset samples. These sample sets that are completely new are considered to be adversarial samples, and if the model performs well with the set of adversarial samples, then the problem of feature drift is addressed, and thus, it can perform well. In this study, we try to tackle the issue of feature drift and investigate the impact of including temporal information in the DL model, drawing inspiration from the Packet2Vec work. Since the Word2Vec approach used in the study [8] produces more duplicated samples and we also try to use a hybrid embedding approach and observed the variation in performance.

1.3 Contributions

Our proposed methodology emphasizes the analysis of complete packet data, moving beyond reliance on manually crafted features. This strategy employs automatic feature extraction techniques to enhance the accuracy of categorization.

The process consists of several distinct phases:

1. N-grams: In this initial phase, we convert the entire packet data into a sequence of words that includes IP address information, enabling us to capture the temporal dynamics present in the data. By representing packets as n-grams, we provide important context that aids in understanding the flow and interaction of network traffic, ultimately contributing to more accurate categorization.
2. Embeddings: Following the extraction of n-grams, we utilize lexical embeddings along with an attention mechanism from NLP. This step involves generating vector representations for each sequence of n-grams. The attention mechanism allows the model to concentrate on the most relevant portions of the data, effectively emphasizing key characteristics that enhance the representation of the packets.
3. Feature Vectors: After creating embeddings, we compute a comprehensive vectorized representation for each packet. This is done by averaging the word embeddings derived from a combination of Global Vectors for Word Representation (GloVe) and the attention mechanism. The result is a succinct representation of each packet that captures its critical features in a format suitable for further analysis.
4. Classification: In the final phase, we take the vectorized representations of the packets and input them into a transformer architecture, which is specifically designed to process sequential data. This allows the model to capture essential temporal information necessary for distinguishing between attack and non-attack patterns in network traffic. By leveraging this advanced classification technique, we aim to enhance our capability to effectively identify and categorize network threats. This refined methodology facilitates a deeper understanding of network traffic behavior, ultimately improving our ability to detect anomalies and potential security threats.

One of the most crucial fields of research is intrusion detection as it is essential to safeguarding people's and organisations' safety and privacy. But according to recent studies from Symantec Corporation, the number of IoT-based assaults has grown by about 20% [9], and similar findings have been made regarding malware for Macs and mobile devices. These statistics suggest that attackers have become more coordinated. We see our contribution in this circumstance as a strong defense against the attackers who are always changing. Today, it is necessary to identify assaults quickly and cheaply with the least amount of interaction. Since our method depends on the auto feature extraction method, it may improve classification accuracy and differentiate bot activity from human conduct.

This document offers an in-depth review of n-gram encoding techniques and their application for automatic feature selection across various fields, including image processing. The entire structure of the paper is structured as below:

In Section 2, we delve into these techniques, discussing their advantages and how they can significantly enhance the performance of IDS. A detailed exploration of the methodology behind n-gram encoding is provided, highlighting its effectiveness in extracting relevant features from complex datasets.

Section 3 presents a comprehensive overview of the entire pipeline designed to manage and process invasive raw data. This section outlines the steps from data collection and preprocessing to feature extraction and model training, emphasizing the challenges encountered at each stage and the strategies implemented to overcome them.

In Section 4, we conduct a thorough analysis of the results obtained from the implementation of the n-gram approach. This section evaluates its effectiveness by comparing the overall processing costs and time required to analyze intrusion data with those of other methodologies. We include statistical data and performance metrics to support our findings and provide insights into the operational efficiency of this technique.

Finally, Section 5 offers concluding thoughts on the proposed n-gram approach, summarizing its strengths and weaknesses. We also discuss potential future directions for research in this domain, suggesting improvements and new applications that could further enhance the effectiveness of n-gram encoding in feature selection and IDS.

2. RELATED WORKS

Text analysis and image processing have advanced significantly with automatic feature extraction. In text processing, word embeddings are frequently created without the requirement for manual feature extraction using several

models, Word2Vec, GloVe, fastText, and BERT [10]. The created embeddings have grown to be tailored for other domains, such as the one indicated in the study [11], since they can solve a variety of text processing-related issues. To give the embedding models a thorough understanding of medical knowledge that cannot be attained by the training of a small corpus of medical data, the embedding model with the medical knowledge base.

The Word2Vec model found its application in understanding the emotions of the text in various domains, like the one mentioned in the study [12]. Similar to text processing, automatic feature extraction is highly beneficial in image processing for the improvement of object detection and classification analysis. The Word2Vec is used in the studies [13, 14] for detecting objects in the sea and for food classification. This shows that the use of embedding models is greatly beneficial for solving various types of classification problems.

Since the problem of intrusion detection is a classification problem with the packet, including the textual form of traffic data, the usage of embedding models in the IDS can be beneficial not only for the improvement of detection accuracy but also for reducing the associated cost and response time. A similar concept of ours is proposed in the study [15] called Deep Packet, which uses raw traffic packet data as input and solves the problems of traffic type identification and application identification. In a deep packet, the entire packet is fed into the DL model for the auto-selection of features; this method is not ideal for the complicated multiple-channel input types.

The concept of auto-extracting features that give importance to the semantic relationship is presented in the study [16], where embedding models like Word2Vec and Glove are initially used and the challenges associated with the auto-extract are verified so that the more appropriate embedding model for packet analysis is identified. Though the automatic feature extraction is done, one of the significant problems with this approach lies in identifying its suitability in a multiclass environment. The word embedding approach is used in malware analysis [17]. In contrast to previous methods discussed above, the features recovered through the use of the Hidden Markov Model (HMM) for the embedding vectors in this study should lead to an improvement in classification accuracy when compared to the straight opcode sequence data. Thus, this research suggests that embedding vectors are a viable substitute for the feature engineering method when it comes to malware identification.

The transformer model BERT is another model used widely for the detection of anomalies in intrusive data. The problem of data imbalance is addressed using transfer learning, and the classification is carried out using the BERT model [18]. The BERT model is further enhanced to make the transformer model lightweight [19]. Though the BERT model is used in the identification of anomalies, it is not used here for the auto-feature extraction. Packet2Vec [8] presents the automated feature extraction procedure in which the raw packet is transformed into a sequence of words, upon which the Word2Vec model is used for the identification of frequent n-grams, and then the classification of attacks is carried out using a supervised ML algorithm. Similar to fast text processing, an approach called FastPacket [20] enhances the Packet2Vec by performing encoding on the raw data in packet format.

Though the idea of creating an auto-feature extraction model is not new, as observed in the literature survey we have done so far, certain research gaps are identified and they are as follows:

1. The current DL models that perform the process of auto feature extraction include the entire packet. Thus, when the complexity of the packet increases with multiple channel inputs, the methods like Deep Packet cannot perform the extraction effectively.
2. As the number of target classes increases, the auto feature extraction needs to consider the semantic importance of the packet. However, this creates a lot of complexity as there exists similar packet information for multiple classes.
3. The Packet2Vec [9] excludes the source of information, like the port and IP number from the packet; however, this is equally important as the context of the information, as a quick identification of the repeated traffic from the same IP or port could also sometimes be an attack. If this information is tracked promptly, a distinction between real and fake traffic can be identified, and this is what is addressed in our proposed work.

3. PROPOSED SYSTEM

In this section, the overall approach for the automatic feature extraction and classification of the IDS is presented. As shown in Figure 1, the overall approach is carried out as multiple phases, and under each phase, we have achieved parallelization. As this work attempts to identify the auto extraction of features and then perform the classification, it is important to identify the presence of an attack in lesser time and lesser computation cost irrespective of the volume and variety of the data. Thus, each of the phases mentioned in the architecture requires parallelization and that is performed in this work using Spark. The phases are as follows:

3.1 N-gram construction

The initial phase of the auto-feature extraction process for intrusion traffic data involves creating a comprehensive dictionary that maps n-grams to integers. This approach is consistent with traditional n-gram methods commonly used in text processing; however, its application in the realm of intrusion detection traffic remains largely unexplored. This lack of exploration is primarily due to the complexities associated with the vast volume and diverse nature of the data typically encountered in this field. Given the emergence of big data, distributed processing techniques are particularly effective solutions.

Making a dictionary, which is needed to create the integer vector, is the first stage. Following that, word embedding is done, and the resulting altered vectors are known as feature vectors.

Algorithm 1: N-gram Dictionary Creation

Input: User input traffic in the form of PCAP files that will serve as the training set (T_s)

Each T_s includes multiuser traffic with multiple packets

$Pack_s$* denotes the individual packet inside a traffic *$user_s$

Output: Dictionary mapping n-gram to numerical form

Process:

Step 1 Load the PCAP files into the spark dataframe
Step 2 Call ngram procedure ()
Step 3 for all $user_s \in T_s$ do
Step 4 for all $pack_s \in user_s$ do
Step 5 initialize n gram generator $ngram \leftarrow ngramgenerator(pack_s)$
Step 6 for all $ngram \in ngramdo$
Step 7 assign to dictionary D and increment the counter
Step 8 end for
Step 9 end for
Step 10 end for
Step 11 Build the pipeline
Step 12 Sort the dictionary D

$$pipe \leftarrow sort_D$$

Step 13 while$i \leq ngramsize$ do

$$D[pipe[i]]$$

$$i++$$

Step 14 end while
Step 15 extract D

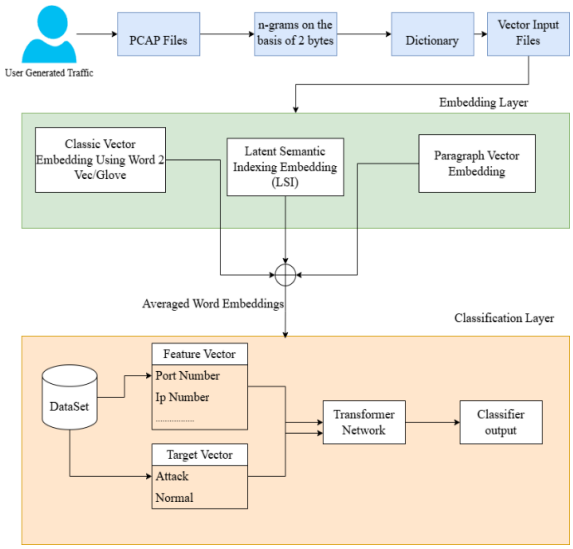

Figure 1. Architecture for identifying IDS attacks

The process begins by iterating through each packet contained in the packet capture (PCAP) files. For every packet, an n-gram analysis is performed, generating subsequences of n items from the data stream. After processing each packet, a loop counter is incremented, ensuring systematic traversal through the dataset. The size of each n-gram—determined by the parameter 'n'—is crucial, as it acts as a hyperparameter that can be adjusted to optimize the model's performance. This n-gram generation process needs to be applied to all PCAP files in the dataset. However, this repetitive task can be resource-intensive, consuming significant amounts of memory and processing time. To alleviate these constraints, the algorithm is designed to utilize distributed computing capabilities. By distributing the packets across multiple processing units, the task of generating n-grams can be executed in parallel, thereby accelerating the overall computation.

The output of the n-gram generation algorithm is a vector composed of various grams, which are then combined to form a single, cohesive vector. This consolidated vector is subsequently mapped to the previously constructed dictionary, enabling efficient data representation and retrieval for further analysis. The comprehensive methodology and steps involved in the n-gram generation process are outlined in the accompanying algorithm, which serves as a guide for implementing this structured approach.

A dictionary containing the necessary data is created using Algorithm 1 to transform PCAP files into the appropriate numeric format. The training set here is the user traffic data that is generated from multiple users with varied types. This variety of traffic and packet are denoted as $user_s$ and $pack_s$. Two loops are thus used for iterating the entire type of traffic from multiple users, within which the conversion of the integer files that are indexed at the packet level with the help of the dictionary denoted as $list_w$.

3.2 Embedding layer

The indexed integer file is then processed for the creation of embeddings. This process is the most crucial step, as this is the phase where the intrusion data is made to fit into the transformer architecture. The corpus of packets gets converted to vectors at this stage and here the embeddings are carried out using the conventional method and semantic work embedding. For the conventional method, Word2Vec [21] is used and the process aims to obtain the vector representation for each of the n-grams in the PCAP traffic file. Semantic embedding is obtained using the latent semantic indexing (LSI) and paragraph vector. Algorithm 2 explains the general procedure involved for the embedding.

Algorithm 2: Vector Generation Using Various Embedding Approaches

Input: set of n gram files
Output: Updated Model embeddings
Procedure

$$List_i \leftarrow TRUE$$

Step 1 Set the initial vale of the model to be true
Step 2 for all l in $List_i$ do
a. if ($List_i \leftarrow TRUE$
$Embed_i \leftarrow generatemodel(l)$
Set $List_i \leftarrow FALSE$)

b. Else
$Embed_i \leftarrow generatemodel(l)$
$endif$
c. End for
d. Output ($Embed_i$)

As mentioned in Algorithm 2, the vector representation in the form of an embedding is obtained by processing the dictionary-converted integers. Thus, after the embedding, a matrix is obtained with dimensions including the matrix size × embedding size, and each row determines the vector of n-grams. This matrix is used consecutively for the next iteration with different integer forms of a PCAP file. The embedding matrix is mathematically obtained using Eq. (1) for various models.

$$P(pcap_{context} = context | pcap_{center} = center) = \frac{\exp\left(vec1^T_{context} vec2_{center}\right)}{\sum_{window \in vocabexp} \left(vect1^T_{window} vect2_{center}\right)} \quad (1)$$

The Word2Vec model considers an n-gram and tries to identify its closeness with other n-grams in terms of the distance value. So basically, given a center word, we aim to maximize the probability of predicting the context word given the center word. As far as the PCAP files are concerned, the center word is not chosen randomly as we do in the regular Word2Vec; instead, the center word is chosen based on the layer. As far as the intrusion PCAP traffic file is concerned, certain attributes get generated from each layer. If we consider the datalink layer, the IP information and port information are generated. So, the center word for each of the layers is chosen, and the context concerning the center word is identified for the prediction of the next word.

Mathematically if we consider the center position of the PCAP as $pcap_{center}$ and the context word as $pcap_{context}$ then the window is defined based on the layer denoted as $pcap_{window}$ then the Word2Vec basically look for attributes from $pcap_{center} - pcap_{window}$ and $pcap_{center} \pm pcap_{window}$ for identifying their context. The relevance of the context word is predicted to that the center word is identified with the computation of the probability as mentioned in Eq. (1), where *vec1* and *vec2* represent the two vectors of each entry of the n-gram. As observed from Eq. (1), since the dot product of context and center is taken, we can find the similarity between them, and the higher probability denotes the higher similarity, and hence the features can be auto-extracted.

To strengthen the embeddings based on the contextual and semantic basis, the classification model has not only been trained with the embeddings from Word2Vec, but it is also supported by other embedding methodologies as well. Paragraph vector embeddings have also been done in this work because it is important that the model not ignore the information relative to the context of the word. In this method, the words are replaced by the document ID so that the same word with different meanings in different contexts is well identified.

How this incorporation benefits better feature extraction can be understood with a sample scenario. As far as this intrusion data is concerned, PCAP traffic includes the n-grams frame, time_relative and tcp. time relative, within which time relative is the same word used in both cases; however, the difference comes from the origin of this feature. As the name denotes, frame.time_relative originates from the data link, and tcp.time_relative comes from the transport layer. This understandability is also needed for the model, so the paragraph vector embeddings are also considered for the model training. Except for the inclusion of the document, the probability computation uses the same probability computation as mentioned in Eq. (1).

The final embedding design that is used in this work is the LSI to further strengthen the embeddings generated based on the semantics. Since this method aims to find out the relevance of the words and documents using frequency computation, this can assist us in giving a more crisp feature set. In this case, the document group defines the category, and the words denote the actual parameters in the PCAP capture. So, for instance, the presence of a frame. relative_time and its occurrence in the training samples are computed for the identification of its relevance to that of the category using Eq. (2).

$$Term\ frquency = freq_{i,j} \times log \frac{|D|}{d: d \ni t_i} \tag{2}$$

where, $freq_{i,j}$ represents the occurrence of the parameter in the category, |D| denotes the entire PCAP dataset, and the t_i denotes the total number of times the parameter is used in the entire PCAP dataset.

After the extraction of embeddings using all three methods, the final word embeddings are needed in a fixed-size vector representation for all the considered packets. This is created by the simple averaging approach, as shown in Eq. (3).

$$feature_{avg} = \frac{\sum_{t \in P} emb(t)}{|p|} \tag{3}$$

where, P denotes the packet, *emb*(*t*) denotes the individual embeddings of each approach and $t \in P$ represents the n-grams of the packet. The procedure for obtaining the feature vectors is mentioned in Algorithm 3.

Algorithm 3: Aggregation Process of the Word Embeddings

Input: 3 set of files as 2D vector $\vec{1} \times \vec{2}$ and the word embeddings D obtained from Algorithm 1
Output: Single vector (FV)

$fori \leftarrow 1 | \vec{1} \times \vec{2} | do$
$vv \leftarrow \vec{1} \times \vec{2}[i]$
$forj \leftarrow 1 | \vec{1} \times \vec{2} | do$
$v \leftarrow vv[j]$
$forallinteger$
$D \leftarrow D + v$
$endfor$
$D \leftarrow \frac{D}{v \vee}$
$FV[j] \leftarrow D$
$endfor$
$write(FV)$
$endfor$

Thus, with the simple averaging of the embeddings, a single representation for the entire packet is obtained, making it suitable for further processing. The format obtained as a feature vector using the embedding method is suitable to be used in any machine or DL model. Thus, the output of this embedding layer results in a set of files that are the auto-extracted feature vectors of each entry in the PCAP file and another set of files that are multiclass label files denoting the various DoS attacks and the normal class of data.

3.3 Attention blocks

In our study, we focus on automating feature extraction while also prioritizing cost minimization to improve computational efficiency. To achieve this, we have incorporated an attention block into our proposed system. This attention block is essential for constructing a context vector, which is generated from a streamlined set of feature sets obtained from the embedding layer.

As illustrated in Figure 2, each feature produced during the embedding process is fed into the attention block. The purpose of this block is to create the context vector based on the relevance and significance of each embedding in relation to the targeted class we are analyzing. This approach ensures that only the most important features contribute to the classification process, resulting in more efficient

computations.

Within the attention mechanism framework, the generated embeddings are considered as keys. The mechanism utilizes two input sequences and a query to identify the most relevant embedding for the task at hand. In our implementation, the two input sequences consist of "feature frame.time_relative" and "tcp.time_relative." These sequences provide crucial temporal information about the frames and TCP connections, respectively. Meanwhile, the query is designed to determine the class to which a specific observation belongs.

To facilitate this classification, we employ a compatibility function, which aids in assessing how well each embedding aligns with the given class. For the calculation of compatibility, we define a straightforward similarity function, as detailed in Eq. (4). This function allows us to quantify the degree of similarity between the embeddings and the query, guiding the attention mechanism in selecting the most relevant features for accurate classification. Through this method, we aim to enhance both the effectiveness and efficiency of our proposed system.

$$f(q,k) = (K,q) \tag{4}$$

where, K and q are the key and query. The process of the attention block is also illustrated in Figure 2, which helps in identifying the most relevant feature vectors.

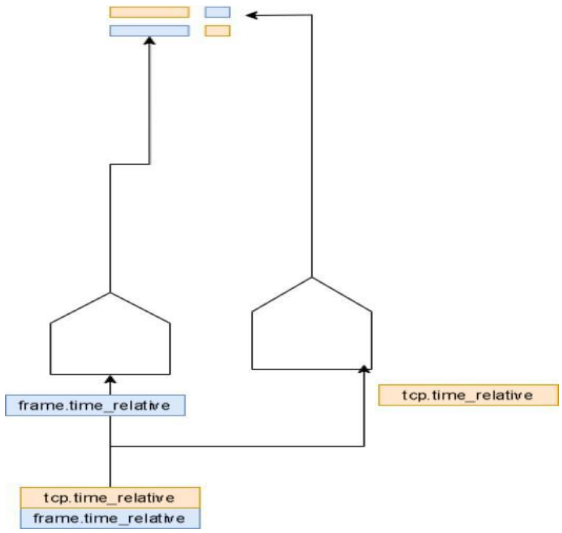

Figure 2. The overall mechanism used in the attention block for the identification of the most relevant feature vectors

3.4 Classification network

With the help of the embedding layer, the lexical features of the PCAP file are constructed using the various language models and the attention block. These features and label vectors are used for the training of the classification network for the detection of normal and abnormal traffic. Algorithm 4 details the overall training and testing of the classification network.

Algorithm 4: Attack Traffic Identification

/* String Extraction */
Step 1 for all packets of normal traffic m in the PCAP file do
a. nm ← extract strings for normal traffic
b. call Algorithm1()
Step 2 end for
Step 3 for all packets in attack traffic a in PCAP file do
a. dos ← extract strings for attack traffic
Step 4 end for
/* Perform steps for the embedding's generation*/
Step 5 enm ← selecting frequent words from m
Step 6 edos ← selecting frequent words from a
/* Construct a Word2Vec model*/
Step 7 Construct a Word2Vec model for enm,edos using Algorithm 2
/* Construct a Program vector model*/
Step 8 Construct a program vector model for enm,edos using Algorithm 2
/* Construct a LSI model*/
Step 9 Construct a LSI model for enm,edos using Algorithm 2
/* Apply Aggregation and Attention*/
Step 10 for all normal traffic nm
a. call Algorithm 3()
Step 11 end for
Step 12 for all attack traffic dos
a. call Algorithm 3()
Step 13 end for
/* Classification */
Step 14 Train transformer(normal,attack)
Step 15 Test transformer
Step 16 for all unknown vectors do
Step 17 label(normal,attack)
Step 18 end for
Step 19 return

The final stage of this effort is the detection of assaults, as method 4 indicates. A transformer network is used for this purpose for the following reasons:

1. Transformer networks can gather a large number of contextual and structural bits of data in the pre-training stage, which improves their ability to identify attacks efficiently and raises the model's generalization capacity.
2. As far as the dataset is concerned, there exists an imbalance where the number of non-attack samples is slightly higher than the number of attack samples, and hence, the non-attack samples are learned effectively. Since the transformer network can effectively learn these differences and can effectively balance the imbalance, this is another reason for choosing this as the base network for performing the classification of PCAP intrusive data.
3. The transformer network is ideal to be used in the detection of anomalies under prompt-based learning, as the task downstream is not dependent on the layers but rather on the keys.

BERT is the transformer model that is used in this work, and the overall process is carried out using three steps in the form of preprocessing, training, and score computation.

Many IDS rely on existing datasets for training and evaluation; however, this approach often proves inadequate when applied to real-world scenarios. This limitation arises because these systems have typically not been tested against new or diverse sets of cyberattacks. To ensure robustness and reliability, it is essential to evaluate how an IDS performs when faced with adversarial samples—data inputs that have been intentionally altered to deceive the ML algorithms into

making incorrect classifications. Adversarial examples are created by making small perturbations to the original data, which can lead to significant misclassification by the system. In our approach, we employ the Jacobian Saliency Map Attack (JSMA) to generate these adversarial samples. JSMA is a sophisticated technique that emphasizes feature selection by identifying the parts of the input data that most significantly affect the model's decision-making process. The process begins with calculating the saliency values of the features in the input data, which indicate their importance in the model's predictions. The algorithm then iteratively applies perturbations to the features, starting with those that have the highest saliency scores, while keeping the number of altered features to a minimum. This targeted strategy not only helps generate effective adversarial examples but also enhances our understanding of the system's vulnerabilities, allowing for improvements in its overall resistance to malicious attacks. By rigorously testing our system using these adversarial samples, we aim to strengthen its ability to detect and respond effectively to new and evolving threats in dynamic environments.

3.4.1 Dataset and pre-processing

This section offers a detailed overview of the dataset generated through the MQTT sensors simulation. The dataset consists of five distinct recorded scenarios: one that represents normal operational conditions and four that depict various types of cyberattack attempts. Each of the four attack scenarios is documented independently, facilitating a thorough examination of the characteristics and impacts of each attack. This dataset is particularly advantageous when compared to older IDS datasets, such as NSL-KDD and CICIDS 2017, as it more accurately reflects patterns of normal network traffic. The authenticity of this dataset is crucial because it enables more effective training and testing of intrusion detection models in realistic environments.

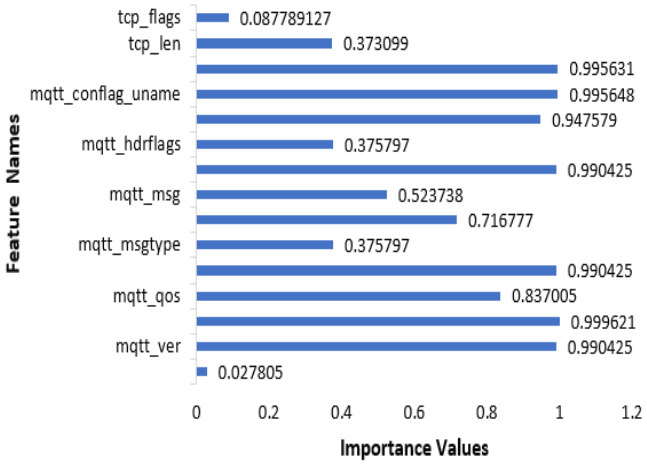

Figure 3. Visualization of the features extracted

The original raw dataset comprised unstructured packet characteristics along with both unidirectional and bidirectional flow information. Due to the complexity inherent in this data, it was essential to restructure and format the layers clearly, designating a transport label for both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) traffic. In preparing the data for analysis, the preprocessed sequences underwent training using the embedding models discussed in the section dedicated to the embedding layer. Notably, this training was conducted without the use of the BERT tokenizer, allowing for a focused examination of the specific data structures within the context of this simulation. The embedding process was carefully designed to ensure that each layer-level feature is effectively learned from the outset, thereby optimizing the performance of the models built upon this dataset. The features extracted and their importance is shown in Figure 3.

4. EXPERIMENTAL RESULTS

4.1 Dataset

For carrying out this study, the MQTT-IOT-IDS2020 dataset is used as the benchmark so that a fair comparison can be made with the other existing ML and DL approaches that were used in the literature and to understand the contribution of the embedding approaches ineffective handling of the unknown traffic. The sample distribution percentage and the classes are shown in Figure 4. As shown in the figure, there is a data imbalance as the normal traffic samples are more than 70%. However, this is not a concern in this work because we are taking the raw PCAP and performing the feature extraction based on layers. The data imbalance and the model generalization are well-balanced in this work.

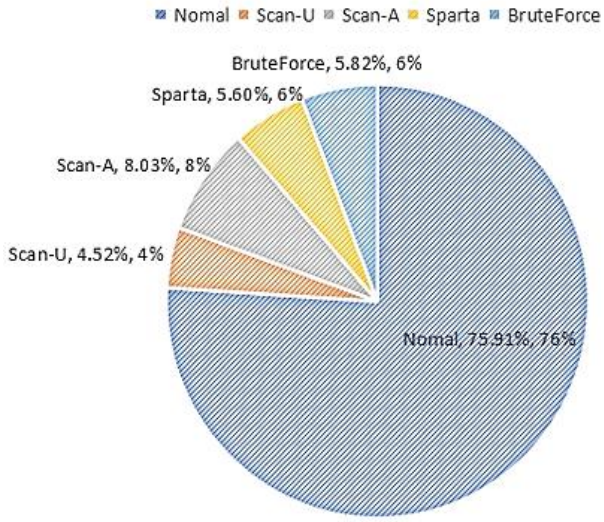

Figure 4. Sample distribution percentage of the dataset

The training and testing ratio is set to 70:30 and there existed redundant data as the feature extraction was automatically done using the PCAP raw files. Here, the various categories of the attacks are considered as a single class named as attack class and hence this becomes a binary classification problem.

5. RESULT ANALYSIS

Since this work concentrated on building a pipeline to solve the problem of feature drift, raw PCAP with auto feature extraction and classification was performed. Thus, the results are analyzed in various aspects, and this section lets us understand the contribution of this pipeline in dealing with the feature drift problem of the IDS so that the adversarial samples are well handled.

5.1 Performance analysis

The performance analysis of this approach is verified with the classification results obtained in the form of the evaluation metrics like Accuracy, Precision, Recall, and F1-score, and they are defined using the equations from Eqs. (5) to (8).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

$$Precision = \frac{TP}{TP + FN} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

$$F1\text{-}score = \frac{2 \times recall \times precision}{recall + precision} \quad (8)$$

Two methods are used to do the experimentation in this case. The MMQT dataset with hand-crafted features was first used, and the assessment criteria listed above were used to assess the dataset's performance. The transformer model produced superior classification results with its handmade characteristics; Table 1 displays the findings.

Table 1. Performance result analysis for the MQTT dataset with varied ML and DL models

Dataset	Classifier	Accuracy	Precision	Recall	F1-Score
MQTT	Transformer	99.91	99.90	99.90	99.90
MQTT	SVM	94.35	93.21	94.21	94.35
MQTT	Ensembled	99.88	99.89	99.88	99.88

Table 2. Meta-analysis of the various detection techniques used on various datasets

Method	Feature Extraction	Dataset	F1-Score
Dugat-LSTM [22]	Principal Component Analysis (PCA)	NSL-KDD	0.99
Two-layer [23]	Common Correlated Feature Selection (CCFS)	NSL-KDD	0.92
HAD-IDS [24]	CL-GAN	NSL-KDD, CICIDS2018, HAD-IDS	0.96
TS-IDS [25]	Graphical Neural Network (GNN)	CICIDS2018, Bot-IoT, Ton-IoT, UNSW-NB15	0.95
ADESSA [26]	Democratic colearning	NSL-KDD, SWAT	0.98
Ensembled [27]	Hybrid	AWID	0.99
Packet2Vec [28]	NLP	Darpa	0.65
NLP [22]	Word2Vec	MAWI	0.82
ML [29]	PCA	MMTQ	0.90
DL [30]	Neural Network	MMTQ	0.92
Proposed	Hybrid	MMTQ	0.99

The problem here is when the newer packet arrives with a whole set of entries that do not match those of the handmade features. As Table 1 illustrates, the detection rate utilizing the handcrafted features in various models is exhibiting superior results. As a result, the antagonistic samples are frequently not handled well. Therefore, each kind of data that is received ought to automatically extract its characteristics before classifying it. In the second experiment, we used the suggested pipeline to test, and the loss value was used to assess the testing and training efficiency. The findings are displayed visually in Figure 5. As shown in Figure 5, training loss and accuracy were observed in varied cases. In the first case, only the unidirectional features were considered, and their training loss and accuracy were measured. As far as the first case is concerned, since there is an absence of acknowledgment, it opens doors for many forms of attack. However, using the embedding approach, due to the correct identification of features that contribute mainly to attack detection, the accuracy and loss were very good. Similar results were observed for Case 2, considering only the biflow features. The amount of accuracy declined a little bit when the uniflow and biflow were combined, which might have resulted from the existence of duplicate features. The confusion matrix after the inclusion of adversarial samples is shown in Figures 6 and 7 for the training and testing sets.

Table 2 shows the F1-score of the proposed pipeline and eight additional models that are a combination of the supervised, semisupervised, and unsupervised learning approaches. It is important to note that the performance of these models is obtained from their respective original studies. The results indicate that the handcrafted features set on varied data exhibited good accuracy. However, there is no notable evidence in this work regarding how the adversarial samples are dealt with. Except for the ADESSA, other works did not make any notable points about the data imbalance issues. And in the case of the two-layer network, which is formed by the combination of KNN and SVM, there was the highest number of false alarms.

Since this work concentrated on the auto feature extraction using NLP techniques, the Packet2Vec and NLP, though using a different dataset, were verified for their F1-scores, and the Packet2Vec showed very low accuracy among all three approaches. Though the comparison is made with different datasets since our concentration is focused on the consideration of the feature drift problem, the PCAP was sampled using the Euclidean Jacobian Saliency Map [1] and those newer samples were evaluated, and their results were also verified through the performance analysis, and the F1-score was observed to be 0.94. So, the problems that are generally faced with training the model using handcrafted features are substantially reduced with the auto-feature extraction approach. Thus, this approach can handle the problem of feature drift effectively. Figure 8 presents the training and testing accuracy with the adversarial samples.

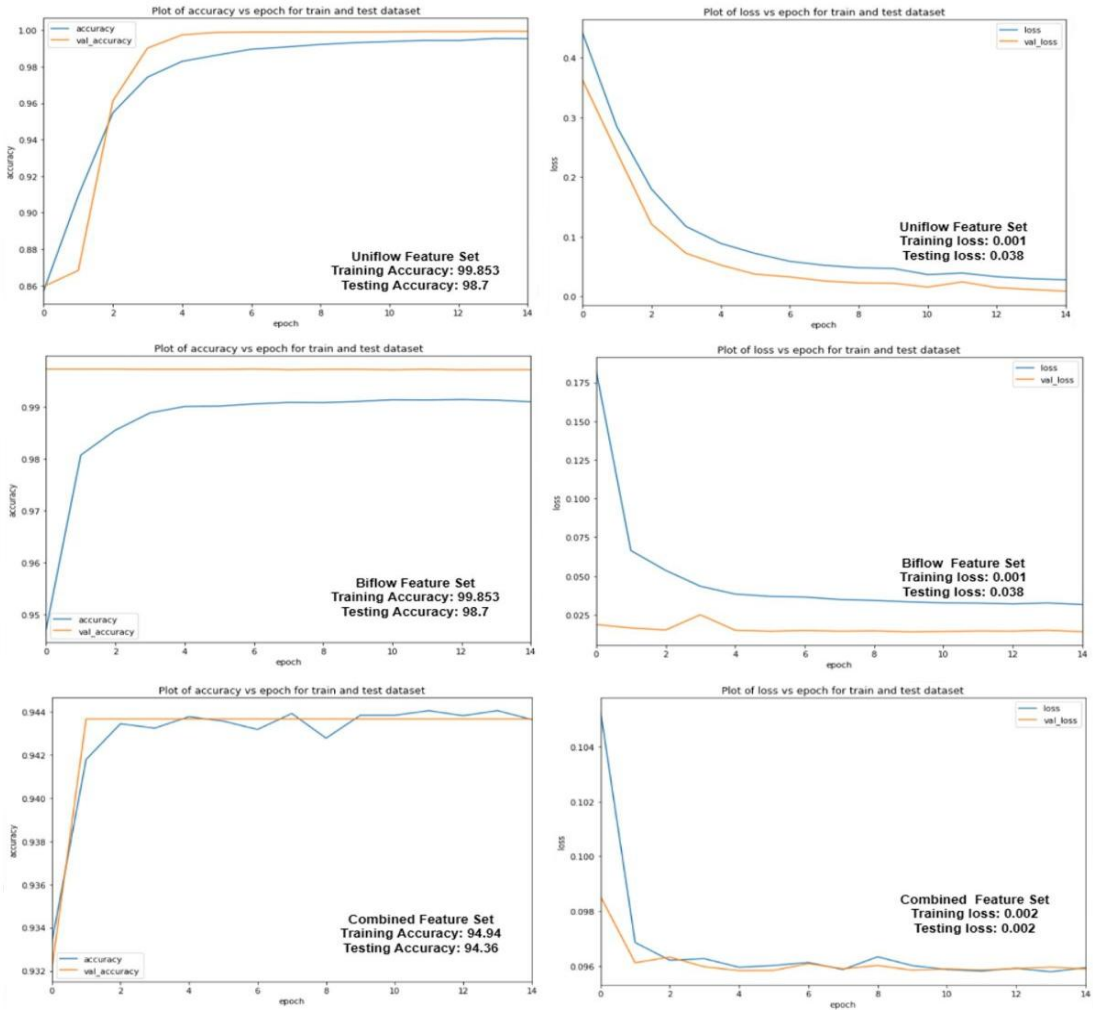

Figure 5. Performance analysis of the proposed pipeline in terms of the accuracy and loss, considering varied features

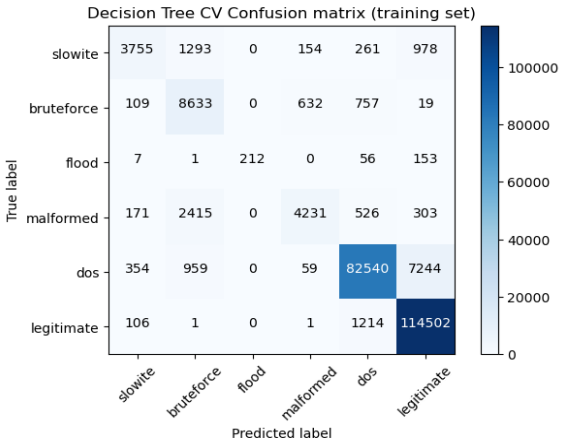

Figure 6. Confusion matrix for the adversarial sample inclusion in the training set

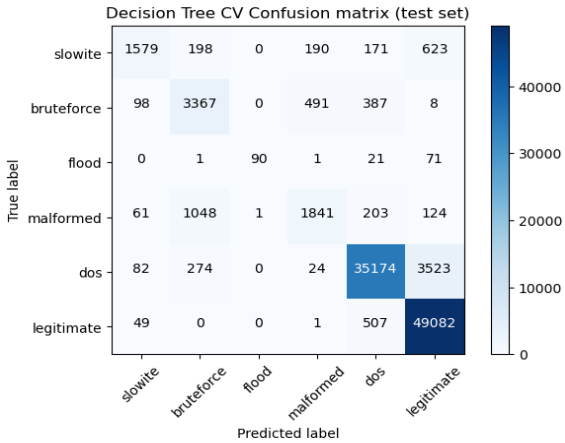

Figure 7. Confusion matrix for the adversarial sample inclusion in the test set

5.2 Processing time

Another important aspect of carrying out this work is to reduce the processing time of the intrusion data, as the intrusion data is generally huge in volume and variety. So, in this approach, distributed processing is achieved using Spark. Though the testing time is less than that of the training time, since this approach comprises several steps, the way this is handled in Spark and how that contributed to the lesser processing time is reviewed in this section. Reading the PCAP file is a serial operation; however, the creation of embeddings needs to be performed in parallel, and thus, all the other subsequent operations that are performed using the embedding and classification layers are distributed using the Spark environment. The processing times for the spark-based operation and serial operation are presented in Figure 9, and the results show that there is a considerable amount of time reduction when the processes are distributed, resulting in the overall processing time.

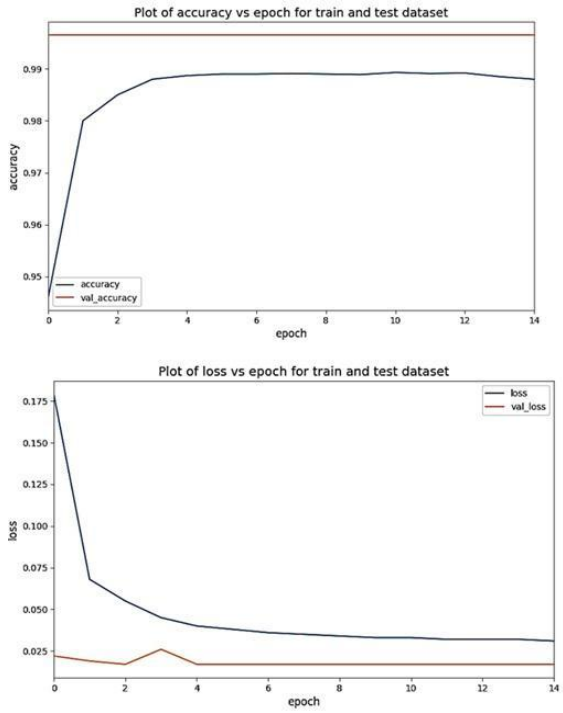

Figure 8. Training and testing loss and accuracy results with the adversarial samples

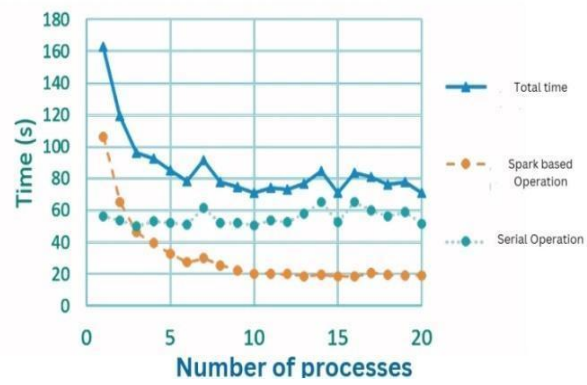

Figure 9. The time taken for the testing of one PCAP file and to obtain the classification results as the processes are distributed the time taken is also considerably reduced

6. CONCLUSIONS

This system, utilizing Spark-based contextual embeddings for automated feature extraction in intrusion detection, presents an innovative pipeline designed for the effective management of PCAP intrusion files, a vital aspect of cybersecurity monitoring and analysis. The proposed pipeline utilizes a robust methodology to transform network packets into vector representations, eliminating the need for predefined handcrafted features. This flexible and adaptive analysis of network traffic allows for the effective detection of intrusions based on the patterns identified within the generated vectors. At the heart of this pipeline is a BERT-based contextual learning model, which has been skillfully integrated for traffic classification.

This model is proficient in understanding the context of both normal and abnormal traffic data, capturing complex patterns and behaviors that may indicate potential threats. Rigorous testing was conducted using samples generated through the Euclidean Jacobian Saliency Attack, and the model demonstrated remarkable performance metrics, including a low prediction error and a high predictive probability value, thus affirming its reliability for real-world applications. Additionally, the proposed method underwent systematic evaluation alongside various alternative supervised and unsupervised learning approaches. The results indicated that our pipeline consistently outperformed these methods, even when relying on auto-extracted feature sets with an overall accuracy of more than 99% with minimal false alarms. The introduction of an attention layer further enhanced efficiency by optimizing the model's focus on relevant features while reducing computational costs.

A significant advancement of this work is the implementation of distributed processing using Apache Spark. This technology has streamlined processing capabilities, significantly decreasing the time required for analysis without compromising the quality of results. Despite the promising outcomes related to feature auto-extraction, several areas for future enhancement have been identified. Notably, the current approach has framed the classification problem as binary, which restricts its applicability in more complex scenarios. Therefore, it is essential to extend the methodology to support multiclass classification. Additionally, while the method was validated using a single dataset, it is crucial to evaluate its effectiveness across a wider range of common intrusion datasets to establish its generalizability and robustness. These considerations will not only inform the ongoing development of our approach but also shape future research directions in the field of intrusion detection and cybersecurity.

REFERENCES

[1] Vijayakumar, D.S., Ganapathy, S. (2023). Adversarial sample generation using the Euclidean Jacobian-based Saliency Map Attack (EJSMA) and classification for IEEE 802.11 using the Deep Deterministic Policy Gradient (DDPG). International Journal on Recent and Innovation Trends in Computing and Communication, 11(8): 204-216. https://doi.org/10.17762/ijritcc.v11i8.7946

[2] Azimjonov, J., Kim, T. (2024). Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. Expert Systems with Applications, 237: 121493. https://doi.org/10.1016/j.eswa.2023.121493

[3] Sharma, B., Pokhrel, S.R., Murali, S. (2024). Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach. Expert Systems with Applications, 238: 121751. https://doi.org/10.1016/j.eswa.2023.121751

[4] Xu, Y., Cao, J., Song, K., Xiang, Q., et al. (2023). FastTraffic: A lightweight method for encrypted traffic fast classification. Computer Networks, 235: 109965. https://doi.org/10.1016/j.comnet.2023.109965

[5] Barddal, J.P., Gomes, H.M., Enembreck, F., Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. Journal of Systems and Software, 127: 278-294. https://doi.org/10.1016/j.jss.2016.07.005

[6] Khan, M.A., Khan, M.A., Khan, K.M., Arif, S., et al. (2023). An optimized ensemble prediction model using AutoML based on soft voting classifier for network intrusion detection. Journal of Network and Computer Applications, 212: 103560. https://doi.org/10.1016/j.jnca.2022.103560

[7] Mananayaka, A.K., Chung, S.S. (2023). Network intrusion detection with two-phased hybrid ensemble learning and automatic feature selection. IEEE Access, 11: 45154-45167. https://doi.org/10.1109/ACCESS.2023.3274474

[8] Goodman, E.L., Zimmerman, C., Hudson, C. (2020). Packet2Vec: Utilizing Word2Vec for feature extraction in packet data. arXiv preprint arXiv:2004.14477. https://doi.org/10.48550/arXiv.2004.14477

[9] Hajiheidari, S., Wakil, K., Badri, M., Navimipour, N.J. (2019). Intrusion detection systems in the Internet of Things: A comprehensive investigation. Computer Networks, 160: 165-191.

[10] Mvula, P.K., Branco, P., Jourdan, G.V., Dobre, C. (2023). Evaluating word embedding feature extraction techniques for host-based intrusion detection systems. Discover Data, 1(1): 2. https://doi.org/10.1007/s44248-023-00002-y

[11] Khine, A.H., Wettayaprasit, W., Duangsuwan, J. (2024). A new word embedding model integrated with medical knowledge for deep learning-based sentiment classification. Artificial Intelligence in Medicine, 148: 102758. https://doi.org/10.1016/j.artmed.2023.102758

[12] Ghosal, S., Jain, A. (2023). Weighted aspect based sentiment analysis using extended OWA operators and Word2Vec for tourism. Multimedia Tools and Applications, 82(12): 18353-18380. https://doi.org/10.1007/s11042-022-13800-4

[13] Liu, K., Wang, W., Chen, J., Zhang, X. (2024). YOLOv5s maritime distress target detection method based on swin transformer. IET Image Processing, 18(5): 1258-1267. https://doi.org/10.1049/ipr2.13024

[14] Saklani, A., Tiwari, S., Pannu, H.S. (2024). Ameliorating multimodal food classification using state of the art deep learning techniques. Multimedia Tools and Applications, 83: 60189-60212. https://doi.org/10.1007/s11042-023-17850-0

[15] Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., Saberian, M. (2020). Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing, 24(3): 1999-2012. https://doi.org/10.1007/s00500-019-04030-2

[16] Kumar, Y., Subba, B. (2023). Stacking ensemble-based HIDS framework for detecting anomalous system processes in windows based operating systems using multiple word embedding. Computers & Security, 125: 102961. https://doi.org/10.1016/j.cose.2022.102961

[17] Kale, A.S., Bhandari, A., Almeida, R., Yang, Y. (2023). Malware classification with Word2Vec, HMM2Vec, BERT, and ELMo. Journal of Computer Virology and Hacking Techniques, 19(1): 1-16. https://doi.org/10.1007/s11416-022-00424-3

[18] Wang, Z., Liu, J., Zhang, Y., Li, H. (2024). A lightweight IoT intrusion detection model based on improved BERT-of-Theseus. Expert Systems with Applications, 238: 122045. https://doi.org/10.1016/j.eswa.2023.122045

[19] Ullah, F., Ullah, S., Srivastava, G., Lin, J.C.W. (2024). IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic. Digital Communications and Networks, 10(1): 190-204. https://doi.org/10.1016/j.dcan.2023.03.008

[20] Jallad, K.A. (2022). FastPacket: Towards pre-trained packets embedding based on FastText for next-generation NIDS. arXiv preprint arXiv:2209.14727. https://doi.org/10.48550/arXiv.2209.14727

[21] Word2Vec implementation. (2019). GitHub repository. https://github.com/dav/word2vec.

[22] Mimura, M., Ito, R. (2022). Applying NLP techniques to malware detection in a practical environment. International Journal of Information Security, 21: 279-291. https://doi.org/10.1007/s10207-021-00553-8

[23] Devendiran, R., Turukmane, A.V. (2024). Dugat-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy. Expert Systems with Applications, 245: 123027. https://doi.org/10.1016/j.eswa.2023.123027

[24] Patthi, S., Singh, S., P, I.C.K. (2024). 2-layer classification model with correlated common feature selection for intrusion detection system in networks. Multimedia Tools and Applications, 83(22): 61213-61238. https://doi.org/10.1007/s11042-023-17781-w

[25] Li, S., Zhao, S., Yang, Y., Cheng, X. (2024). HDA-IDS: A hybrid DoS attacks intrusion detection system for IoT by using semi-supervised CL-GAN. Expert Systems with Applications, 238: 122198. https://doi.org/10.1016/j.eswa.2023.122198

[26] Nguyen, H., Kashef, R. (2023). TS-IDS: Traffic-aware self-supervised learning for IoT network intrusion detection. Knowledge-Based Systems, 279: 110966. https://doi.org/10.1016/j.knosys.2023.110966

[27] Niu, Z., Xu, H., Sun, Y., Sun, Z., et al. (2023). A novel anomaly detection approach based on ensemble semi-supervised active learning (ADESSA). Computers & Security, 129: 103190. https://doi.org/10.1016/j.cose.2023.103190

[28] Vijayakumar, D.S., Ganapathy, S. (2022). Multistage ensembled classifier for wireless intrusion detection system. Wireless Personal Communications, 122: 645-668. https://doi.org/10.1007/s11277-021-08917-y

[29] Hindy, H., Bayne, E., Bures, M., Atkinson, R., et al. (2021). Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset). In Selected Papers from the 12th International Networking Conference, pp. 134-153. https://doi.org/10.1007/978-3-030-64758-2_6

[30] Khan, M.A., Khan, M.A., Jan, S.U., Ahmad, J., et al. (2021). A deep learning-based intrusion detection system for MQTT enabled IoT. Sensors, 21(21): 7016. https://doi.org/10.3390/s21217016