國立臺北科技大學

創新資安碩士班

碩士學位論文

S2GE-NIDS: A hybrid architecture combining structured semantics and generation embedded network intrusion detection system in IoT

研究生：周玟萱

指導教授：陳香君博士

中華民國一百一十四年五月

國立臺北科技大學

創新資安碩士班

碩士學位論文

S2GE-NIDS: A hybrid architecture combining structured semantics and generation embedded network intrusion detection system in IoT

研究生：周玟萱

指導教授：陳香君博士

中華民國一百一十四年五月

「學位論文口試委員會審定書」掃描檔

審定書填寫方式以系所規定為準，但檢附在電子論文內的掃描檔須具備以下條件：

1. 含指導教授、口試委員及系所主管的完整簽名。
2. 口試委員人數正確，碩士口試委員至少 3 人、博士口試委員至少 5 人。
3. 若此頁有論文題目，題目應和書背、封面、書名頁、摘要頁的題目相符。
4. 此頁有無浮水印皆可。

審定書不用頁碼

Abstract

Title: S2GE-NIDS: A hybrid architecture combining structured semantics and generation embedded network intrusion detection system in IoT

Pages: (Fill it)

School: National Taipei University of Technology

Department: Science in Information Security

Time: May, 2025

Degree: Master of Science in Information Security

Researcher: Wen-Hsuan, Chou

Advisor: Shiang-Jiun Chen, Ph.D.

Keyword: Image Classification, Genetic Programming, Loss Function, Deep Learning

In the recent, with the rapid rise of deep learning, image classification models have quickly become one of the most popular and widely known models. When training such models, the steps are broadly divided into the following: preparing image datasets, dividing them into training, validation and testing sets as needed, training the model, evaluating the model, and repeating these steps until the ideal result is met or the computational resources are exhausted.

A crucial function during the process of training a model is called the loss function, which calculate the difference between the predicted values and the ground-truth values. The results of the loss function can significantly influence the effectiveness of the model's training because it simply decide the direction of the adjustment to the model. However, designing a loss function often requires the assistance of experts in the related field, leading to a resource-intensive design process. Recent research has proposed using Genetic Programming (GP) to generate loss functions to avoid the necessity of hiring numerous domain experts for assistance. Nevertheless, using GP typically results in decreased computational efficiency. This paper aims to improve the method of using GP to generate loss functions by modifying certain genetic operations and introducing the concept of tournament selection.

這邊要加結果

Acknowledgements

所有對於研究提供協助之人或機構，作者都可在誌謝中表達感謝之意。

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Driven by the rapid advancement of digital transformation and smart infrastructure, the **Internet of Things (IoT)** has emerged as a cornerstone of next-generation information technology. Through the integration of sensors, embedded devices, communication modules, and platform software, IoT enables physical objects to communicate in real time and generate massive volumes of data. These data streams support a broad range of applications—such as smart manufacturing, intelligent transportation, remote healthcare, and smart homes—yielding substantial economic and societal value [1].

However, as the number of connected devices increases and deployment scenarios become more complex, IoT systems face unprecedented cybersecurity challenges. Many IoT devices are resource-constrained, infrequently updated, and difficult for users to manage. With limited encryption and a lack of monitoring mechanisms, these devices become prime targets for cyber intrusions and attacks. Effectively identifying abnormal behaviors and hidden threats in IoT network traffic has therefore become a pressing research priority.

Furthermore, existing intrusion detection technologies often struggle to adapt to evolving threats. While deep learning approaches such as Word2Vec and Transformer-based models [2, 3] have demonstrated semantic learning capabilities, they also introduce critical drawbacks: large vocabulary requirements, high computational complexity, and limited flexibility in dynamic or resource-constrained environments.

To address these limitations, we propose **S2GE-NIDS** (Structured Semantics and Generation Embedded Network Intrusion Detection System)—a lightweight, interpretable anomaly detection framework designed for IoT environments. S2GE-NIDS combines hash-based token embedding with a multi-layer perceptron (MLP) model and introduces a linked-list mechanism to mitigate hash collisions inherent to non-cryptographic hash functions such as MurmurHash3 [4]. This design enables efficient feature encoding while avoiding the need to maintain a large vocabulary.

In our approach, network packets are first transformed into semantic tokens and encoded using hash-based indexing. The resulting embedding vectors are concatenated into a single, fixed-length semantic vector, which is processed by an MLP and projected near a learned semantic

center. Any significant deviation from this center—measured by Mahalanobis distance—is classified as a potential anomaly [5].

The proposed S2GE-NIDS framework offers several key advantages over conventional intrusion detection systems. First, it eliminates the need for manual feature engineering and vocabulary maintenance by using a hash-based embedding approach, where field-value pairs are directly encoded into semantic vectors without relying on predefined lookup tables. This design greatly simplifies the preprocessing pipeline and enhances scalability. Second, the model provides a mathematically interpretable anomaly scoring mechanism by integrating Mahalanobis distance, which quantifies how far a sample deviates from the learned distribution of normal behavior. This not only improves detection accuracy but also enables explainable results. Third, the system is lightweight and highly efficient, relying on simple MLP-based encoding instead of complex deep architectures, making it well-suited for deployment in real-time or resource-constrained environments such as edge devices in IoT networks. Lastly, its generalized tokenization strategy allows for wide applicability across diverse packet structures, further improving its adaptability and robustness in various network scenarios.

The structure of this paper is as follows: Chapter 2 is the relevant background knowledge about S2GE-NIDS (Structured Semantics and Generation Embedded Network Intrusion Detection System). Chapter 3 introduces the architecture and methodology of the proposed S2GE-NIDS framework, presenting each module and its rationale in detail. Chapter 4 presents the experimental setup, evaluation metrics, and results on two benchmark datasets, as well as interpretability demonstrations. Chapter 5 summarizes the main findings, limitations, and directions for future research.

Chapter 2 Related Work

This section will introduce the relevant basic knowledge, including existing IoT network intrusion detection methods, Tokenization, Hash Embedding and language tags.

2.1 Network Intrusion Detection System in IoT

In recent years, the proliferation of Internet of Things (IoT) devices has led to an increased focus on developing effective network intrusion detection systems (NIDS) tailored to the specific characteristics of IoT environments. Various approaches have been proposed to address the challenges associated with high-volume, heterogeneous network traffic, constrained device capabilities, and evolving attack patterns. Kharoubi et al. [6] proposed NIDS-DL-CNN, a convolutional neural network (CNN)-based detection system designed for IoT security. By applying CNN layers to extract spatial features from traffic data, the model achieved high classification performance on datasets such as CICIoT2023 and CICIoMT2024. The authors demonstrated that their method achieved excellent precision and recall in both binary and multi-class scenarios. However, a notable limitation of the CNN-based approach lies in its inability to fully capture temporal dependencies across packet sequences, and its reliance on supervised learning requires extensive labeled datasets. Ashraf et al. [7] introduced a real-time intrusion detection system (INIDS) based on traditional machine learning classifiers applied to the BoT-IoT dataset. The study compared seven algorithms, including Random Forest, Artificial Neural Networks (ANN), and Support Vector Machines. Their results showed that Random Forest and ANN achieved the highest accuracy and robustness among all tested classifiers. Despite its efficiency, the INIDS system was highly dependent on manual feature engineering and lacked adaptability to novel threats, which are critical in fast-evolving IoT environments. Elrawy et al. [8] conducted a comprehensive survey of intrusion detection methodologies in IoT-based smart environments, categorizing techniques according to architectural design (centralized vs. distributed), detection strategy (signature-based, anomaly-based, or hybrid), and system layer (perception, network, application). While the survey provided valuable insights and synthesized a broad range of IDS approaches, it lacked im-

plementation evidence and empirical comparisons, limiting its utility for practical system design. Collectively, these studies highlight the trade-offs between detection performance, computational cost, and deployment feasibility. Deep learning models offer strong accuracy but demand computational resources, while traditional classifiers provide efficiency but often lack flexibility. In contrast, our proposed S2GE-NIDS framework leverages hash-based semantic embeddings and a lightweight MLP, offering a balanced approach that combines scalability, interpretability, and effectiveness in detecting network anomalies in resource-constrained IoT environments.

Table 2.1 Common Anomalous Features in IoT Network Traffic and Their Descriptions

Feature (with Reference)	**Description**
Destination Port [9]	Specific port targets (e.g., 22, 23, 80, 443) are often associated with attacks. Abnormal access to these ports may suggest behaviors such as scanning, DDoS, or brute-force intrusion.
Flow Duration [10]	Extremely short or long connection durations within brief timeframes may signal scanning activity or data exfiltration.
Total Forward Packets [11]	Unusually high or low packet counts in one direction may indicate abnormal sessions or flooding behavior.
Packet Length [9]	Anomalies in packet size—whether fixed, too long, or too short—often reflect malicious traffic like botnet propagation or worms.
Protocol Type [10]	Sudden increases in uncommon protocols (e.g., ICMP, UDP) may reveal attempts to exploit protocol vulnerabilities or bypass filters.
Source IP / Destination IP [12]	Repeated access from abnormal IP addresses, or sudden surges in novel IP sources, are indicative of scanning, spoofing, or DDoS activity.
Flow Bytes per Second [9]	Sharp fluctuations—surges or drops—in flow byte rate may suggest DoS attacks or unauthorized data transfer.
TCP Flags [11]	Unusual combinations (e.g., SYN, FIN, RST) can indicate stealth scans or TCP-based flooding.
Number of Connections [9]	A large number of new connections established by a single IP in a short time often reflects worm propagation or botnet coordination.

2.2 Tokenization

Following feature extraction, the next critical step is the tokenization process, which prepares network traffic data for semantic embedding. Each data record typically contains multiple fields—such as `Port`, `Protocol`, and `SrcIP`—representing structural and behavioral attributes of a network flow. To ensure consistency and distinguishability among features during embedding, we adopt a composite tokenization strategy that combines each field name with its corresponding value to form a unique token string. For instance, a sample token may take the form

`Protocol:TCP` or `DstPort:443`.

This strategy preserves the semantic association between field-value pairs without relying on predefined vocabularies, making it particularly suitable for dynamic and heterogeneous IoT environments. Each composite token is subsequently encoded using hash-based mapping techniques (as described in Section **??**), thereby eliminating the need for extensive memory allocation or manually constructed token dictionaries. By treating each token as a self-contained semantic unit, this method also enhances the model's ability to generalize to previously unseen feature combinations, ultimately improving both the adaptability and scalability of the proposed system [13].

Table 2.2 Examples of Field-Value Tokenization in IoT Network Traffic

Feature Field	**Tokenized Representation**
Protocol = TCP	Protocol:TCP
Destination Port = 443	DstPort:443
Source Port = 80	SrcPort:80
Source IP = 192.168.0.1	SrcIP:192.168.0.1
Flow Duration = 120000	FlowDuration:120000
Payload Bytes = 56	PayloadBytes:56
Packet Count = 10	PacketCount:10
Flag = ACK	Flag:ACK
Protocol = ICMP	Protocol:ICMP
Destination IP = 10.0.0.5	DstIP:10.0.0.5

Chapter 3 Methodology

In this session, we will introduce the S2GE-NIDS (structured semantics and generation embedded network intrusion detection system) architecture and details its operational workflow, clearly delineating each step from semantic tokenization through anomaly detection and decision-making processes.

3.1 Architecture

S2GE-NiDS is presented as Figure 3.1 including preprocess model, embedding model, and Mahalanobis model.

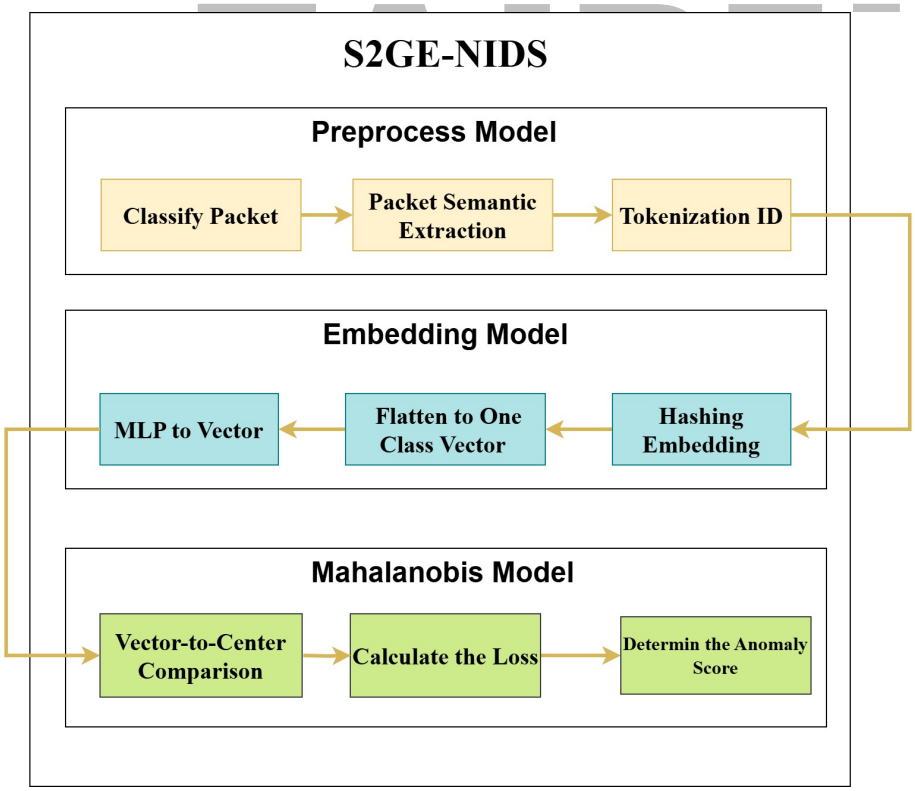

Figure 3.1 Architecture of S2GE-NIDS

During the preprocessing stage, relevant features are first extracted from network data packets and transformed into a combination of textual and numerical tokens. To prevent duplication and ensure a more uniform distribution within the embedding space, the model applies the non-encrypted MurmurHash3 function to encode each token.

To further mitigate the risk of hash collisions, a modulo operation is performed on the resulting hash values, which are then used to index into the embedding table. This strategy reduces the likelihood of different tokens being mapped to the same location, thereby enhancing both the accuracy and efficiency of the embedding process.

Next, the individual embedding vectors corresponding to each feature are concatenated into a single one-dimensional vector. This flattened vector is fed into a multi-layer perceptron (MLP) model, which transforms it into a compact semantic representation.

Finally, in the Mahalanobis distance evaluation phase, the semantic vector is compared to a predefined center point in the learned semantic space. If the computed distance exceeds a specified threshold, the sample is flagged as an anomaly. Evaluation metrics such as the F1 score are then used to quantify detection performance.

3.1.1 Preprocess Model

In the preprocessing phase, we will do the following process as data file selection and filtering, feature extraction, and tokenization. These steps are designed to transform raw network traffic into structured representations suitable for semantic embedding and anomaly detection.

A、Classify Data Packet

The first step in the preprocessing pipeline involves selecting and filtering the data files to ensure suitability for subsequent analysis. In this study, network traffic is collected and stored in the Comma-Separated Values (CSV) format—a widely adopted and flexible tabular data structure. CSV files are particularly well-suited for structured data representation due to their ease of parsing, compact storage, and seamless integration with mainstream data analysis libraries such as pandas and NumPy in Python.During this stage, only those CSV files containing the required packet-level features are retained, while incomplete, irrelevant, or malformed files are systematically excluded.

B、Packet Semantic Extraction

After the data is cleaned and organized, the first step is to extract meaningful features from the network packet data to better capture the characteristics of each packet. For example, we focus on key fields such as Destination Port, Protocol, and SrcIP, which are widely used in previous studies to detect abnormal network behavior [42]. In practice, we use Python tools to read each CSV file and select these important features as the main input for the S2GE-NIDS model. By focusing only on these key values, we can make the data cleaner and easier for the anomaly detection system to use.

C、Tokenization ID

After the relevant features have been extracted, the next step is to perform tokenization, which converts structured data into a format suitable for semantic embedding. Each data entry consists of multiple fields—such as `Destination Port`, `Protocol`, and `SrcIP`—that represent different aspects of network behavior.

Tokenization is achieved by concatenating each field name with its corresponding value to form a unique string representation. This composite token serves as the semantic unit used in downstream embedding processes. For example, tokens follow the format *"field name + field value"*, as illustrated in Table 3.1.

Table 3.1 Example of Tokenized Input Fields

Field Name	**Field Value**
Destination Port	80
Flow Duration	192.168.1.2
Protocol Type	TCP

3.1.2 Embedding Model

To mitigate redundancy in text features during the embedding process, we employ a lightweight, non-cryptographic hash function—MurmurHash3. This function ensures that input tokens are

more uniformly distributed across the embedding space, thus reducing overrepresentation in specific regions. To further minimize the probability of hash collisions—i.e., multiple tokens being mapped to the same position—we apply a modulo operation to the resulting hash values. This yields a deterministic index used to locate or store each feature vector within a fixed-size embedding table, enhancing both the accuracy and efficiency of the overall embedding process.

Once all relevant token embeddings are retrieved, their vectors are concatenated into a single flattened, one-dimensional feature vector. This unified representation is then fed into a multilayer perceptron (MLP), which learns high-level semantic abstractions and generates a compact semantic feature vector. The subsequent subsections provide detailed descriptions of the hash embedding mechanism, the flattening procedure, and the structure of the MLP used for semantic encoding.

A、Hash Embedding

Hash embedding is a lightweight vectorization technique that utilizes non-cryptographic hashing to encode tokenized field-value pairs into fixed-size, trainable embeddings [13]. In this study, we adopt the MurmurHash3 algorithm—an efficient and widely used hash function—to map each token to a specific position in the embedding table. Its advantages include fast computation, uniform distribution, and language-independent implementation, which make it well-suited for scalable anomaly detection in IoT environments [4].

To determine the target index for each token, we apply a modulo operation to the hash value using the smallest three-digit prime number, 233. This approach distributes tokens more evenly within the embedding space and reduces collision rates. For example, the token generated from the field name `PORT` may yield a MurmurHash3 value of 4283257230. Applying `4283257230 mod 233` results in 56. If the associated port number (e.g., 405) is similarly hashed and gives a value with mod 233 result of 7, these indices (row 7, column 56) are used to locate the corresponding vector in the embedding table.

Each embedding vector is initially randomized and refined during training. For instance, an example 8-dimensional vector might be:

$$[-0.982,\ -0.301,\ -0.555,\ 2.061,\ 0.045,\ -0.618,\ -0.786,\ 0.573]$$

These vectors are later concatenated and passed to the MLP model for further semantic encoding.

B、Flatten

Flatten will string the tokenized data into a single vector through the vectors after the embedding column. For example, Destinazation port 405 is [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573] andXXX 經過 tokenization 變成 [-0.024, 0.494, 0.754, -0.78, -1.002, 0.069, -0.52, -1.336] ,XXX [-1.042, -0.116, 0.542, -0.987, 1.001, 0.086, 0.699, -0.903] [-0.982, -0.301, -0.555, 2.061, 0.045, -0.618, -0.786, 0.573, -0.024, 0.494, 0.754, -0.78, -1.002, 0.069, -0.52, -1.336, -1.042, -0.116, 0.542, -0.987, 1.001, 0.086, 0.699, -0.903]

C、MLP

After generating semantic embeddings from each tokenized field, the resulting vectors are flattened into a single one-dimensional input vector. This unified semantic representation is then fed into a lightweight **Multi-Layer Perceptron (MLP)** to learn deeper semantic relationships and perform nonlinear transformation for anomaly detection.

The MLP adopted in this work is composed of an input layer, one or more hidden layers, and an output feature vector. Each layer consists of a fully connected network of neurons activated by the ReLU (Rectified Linear Unit) function, which enables the model to capture non-linear dependencies among input features while maintaining computational efficiency.

To prevent overfitting and enhance generalization, dropout layers are incorporated after each dense layer, and batch normalization is applied to stabilize the training process. The final output of the MLP is a low-dimensional semantic vector projected in a latent space. This vector is subsequently used for statistical anomaly detection based on its distance from a learned semantic center.

Compared to more complex architectures such as transformers or recurrent models, the MLP achieves an optimal balance between *expressiveness, interpretability, and computational efficiency*, making it particularly suitable for deployment in IoT environments with limited resources

Chapter 4 Result & Analysis

Chapter 5 Conclusion & Future Work

5.1 Conclusion

5.2 Future Work

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. “The Internet of Things: A survey”. In: *Computer Networks* 54.15 (2010), pp. 2787–2805.

[2] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. Vol. 30. 2017.

[4] Austin Appleby. *MurmurHash3*. `https://github.com/aappleby/smhasher`. 2011.

[5] Guoyin Liu, Ying Zhang, and Ming Sun. “Anomaly detection using Mahalanobis distance for high-dimensional data”. In: *IEEE Access* 8 (2020), pp. 211731–211741.

[6] Kamir Kharoubi et al. “Network Intrusion Detection System Using Convolutional Neural Networks: NIDS-DL-CNN for IoT Security”. In: *Cluster Computing* 28.219 (2025).

[7] Jawad Ashraf et al. “Making a Real-Time IoT Network Intrusion-Detection System (INIDS) Using a Realistic BoT−IoT Dataset with Multiple Machine-Learning Classifiers”. In: *Applied Sciences* 15.4 (2025), p. 2043.

[8] Mohamed Faisal Elrawy, Ali Ismail Awad, and Hesham FA Hamed. “Intrusion detection systems for IoT-based smart environments: a survey”. In: *Journal of Cloud Computing* 7.1 (2018), pp. 1–20.

[9] Tao Tang et al. “Deep learning approach for network intrusion detection in software-defined networking”. In: *IEEE Access* 7 (2019), pp. 110740–110749.

[10] Zhen Liu and Xudong Jiang. “A Lightweight Intrusion Detection Method for IoT Based on Reinforcement Learning and Feature Selection”. In: *Sensors* 21.22 (2021), p. 7601.

[11] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. 2018, pp. 108–116.

[12] Mahbod Tavallaee et al. “A Detailed Analysis of the KDD CUP 99 Data Set”. In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 2009, pp. 1–6.

[13] Kilian Weinberger et al. “Feature hashing for large scale multitask learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*. 2009, pp. 1113–1120.