Growing Self-Organizing Maps for Firewall Log Classification in Data Streams

Wagner Rafael Giarini[1], **Herbert Gonçalves Dias**[1], **Ricardo Cerri**[2]

[1]Federal University of São Carlos (UFSCar), Department of Computer Science
São Carlos, SP – Brazil

[2]University of São Paulo, Institute of Mathematics and Computer Science
São Carlos, SP – Brazil

wagner@ufscar.br, herbert@ufscar.br, cerri@icmc.usp.br

Abstract. *The growing volume of data and dynamic nature of computational networks present significant challenges to information security, particularly in data stream classification. This work investigates multiclass classification of real firewall logs from an academic environment. To address class imbalance, binary classification (allowed and denied actions) was performed. We propose an adaptation of the Growing Self-Organizing Map (GSOM) for streaming classification. Experiments were conducted in offline and online phases, evaluating three update strategies. The results demonstrate GSOM's streaming adaptability and potential for classifying firewall logs in data streams.*

1. Introduction

Network Security Systems (NSS), such as intrusion detection and firewall log classification, traditionally rely on batch approaches and are not well adapted to data streams (DS) [Qu et al. 2021, Haripriya et al. 2024]. The exponential growth of internet traffic and new types of cyberattacks (e.g., phishing, ransomware, DDoS) have made real-time, adaptive security solutions critical. Furthermore, manually managing firewall rules based on actions such as "Allow", "Deny", "Reset Both" or "Drop" is often prone to errors, potentially resulting in critical security vulnerabilities [Ucar and Ozhan 2017].

Machine Learning (ML) techniques have shown promise in this context [Dua et al. 2019], particularly models like KNN, Random Forests, and Artificial Neural Networks [Liao and Vemuri 2002, Patgiri et al. 2018]. However, most ML models depend on static parameters, limiting their adaptability to changing traffic patterns. Self-Organizing Maps (SOMs) [Kohonen 2001] offer continuous learning capabilities but suffer from rigid topologies that are unsuitable for dynamic, evolving data [Alahakoon et al. 2000].

To address these limitations, this work investigates Growing Self-Organizing Maps (GSOMs) [Alahakoon et al. 2000] for the classification of firewall logs in data streams. We evaluate two approaches: multiclass classification (four firewall actions) and binary classification between allowed ("allow") and denied traffic (combining "Deny", "Reset Both" and "Drop" actions).

The experiments were divided into two phases: offline and online. In the offline phase, the GSOM was trained on a batch dataset to establish an initial model. In the online phase, three distinct experiments were conducted to evaluate different GSOM adaptation strategies: (i) identification of the closest winning neuron without modifying the map

structure or its weights; (ii) dynamically expanding the map by adding new neurons to better accommodate the input data; and (iii) adjusting the weights of existing neurons to refine the map without altering its structure. We provide a detailed analysis of the results obtained, evaluating the GSOM's behavior in each strategy and highlighting its applicability in classifying firewall logs within data streams.

2. Related Work

Research in firewall log analysis prioritizes multiclass classification of actions (*allow*, *deny*, *drop*, *reset-both*) for granular policy control. Ertam and Kaya 2018 achieved 98.5% recall for *reset-both* using SVM, while Sharma et al. 2021 reached 99.8% accuracy with ensemble stacking. Aljabri et al. 2022 addressed severe class imbalance (*reset-both* $<$ 0.5%) via supervised under-sampling, reporting 99.64% accuracy and 99.60% F1-Score after including application and category features.

Conversely, binary approaches prioritize anomaly detection over firewall policy actions. Allagi and Rachh 2019 applied K-means clustering and SOM to a large-scale dataset from the UCI repository, achieving an accuracy of 97.2% in distinguishing normal from abnormal events, while Ucar and Ozhan 2017 utilized algorithms such as Naive Bayes, kNN, and Decision Tables to detect policy anomalies in firewall logs, with kNN reaching up to 100% accuracy in anomaly detection. No studies were found that frame the problem as the direct classification of permitted versus blocked traffic. In contrast to previous studies, our work employs GSOM in a streaming context, offering adaptability to dynamic and imbalanced firewall log datasets.

3. The Growing Self-Organizing Map

This section introduces the Growing Self-Organizing Map (GSOM), used in our proposal for classifying firewall logs in data streams. A challenge addressed in our work is the concept of infinite label latency, where labels are never available during classification [Das et al. 2020]. We present how GSOM addresses key challenges associated with dynamic and non-stationary data, focusing on its application in network security scenarios.

To understand the foundations of the GSOM algorithm, it is essential to first revisit the original Self-Organizing Map (SOM) principles. The SOM is a neural network model that projects high-dimensional data onto a lower-dimensional (typically two-dimensional) grid, preserving the topological relationships among input patterns [Kohonen 2001]. In this structure, each neuron in the grid represents a prototype vector, and the number of neurons is set according to the dataset characteristics. The input layer matches the number of attributes in the dataset, with all features presented to every output neuron [Haykin 2009, Rolemberg 2021]. Figure 1 illustrates the mapping process, which forms the foundation for the GSOM extension.

Although the traditional SOM performs well in many scenarios, it faces challenges when handling highly dynamic and continuously evolving data, such as network traffic analysis. In these cases, a more adaptive approach is required to effectively handle the complexities of changing data distributions.

One such approach is the use of the GSOM, an unsupervised neural network designed to address the limitations of the traditional SOM. Unlike the fixed topology of the

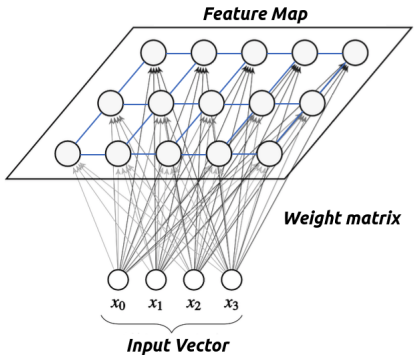


Figure 1. The SOM architecture. Adapted from [Barnawi et al. 2023].

SOM, the GSOM starts with a minimal structure of four neurons and dynamically adds new neurons as needed to capture the underlying patterns in the input data [Casarotto and Cerri 2024]. As new neurons are created, the weight values of the neurons are reorganized in a manner similar to the SOM process. The GSOM phases are described in the following subsections.

3.1. Initialization Phase

The initialization phase sets the foundation for the GSOM network using the *spread factor* (SF), a user-defined hyperparameter controlling map granularity [Zheng et al. 2008]. This SF directly determines the *growth threshold* (GT), which represents the maximum accumulated error a neuron can have and is computed based on the dimensionality D of the data vectors and the SF using Equation 1.

$$GT = -D \cdot \ln(\text{SF}) \tag{1}$$

Here, D refers to the dimensionality of the data vectors, and SF represents the growth control variable, which ranges between 0 and 1. A value of 0 corresponds to minimal growth, while 1 indicates maximal growth [Alahakoon et al. 2000].

To initialize the GSOM network, four neurons are arranged in a two-dimensional square lattice, as shown in Figure 2. This configuration provides an ideal starting point for constructing a flexible lattice structure, with each neuron serving as a boundary unit to enable dynamic growth in multiple directions based on the input data.

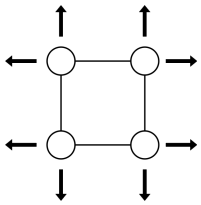


Figure 2. Initial GSOM configuration. Adapted from [Alahakoon et al. 2000].

The weight vectors of the initial neurons are randomly initialized within the range of the normalized input data (0 to 1). This setup ensures compatibility and allows the network to grow in any direction dictated by the input distribution.

3.2. Growing Phase

The growing phase is responsible for expanding the map by creating new neurons when necessary. This expansion is controlled by GT, which determines when the accumulated error of a neuron justifies the addition of a new one [Alahakoon et al. 2000]. During training, the neuron with the smallest Euclidean distance to the presented input vector $\mathbf{x}_j$ is identified as the winner. The weight vector $\mathbf{w}$ of the winner and its neighboring neurons are adapted by the original SOM learning scheme as follows:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t) \cdot h(t) \cdot (\mathbf{x}_j - \mathbf{w}_i(t)) \tag{2}$$

where $\mathbf{w_i}$ is the weight vector of neuron i, t is the current time step, η is the learning rate, and h is the neighborhood function, which can be Gaussian. Both the learning rate and the width of the neighborhood decrease over iterations during the weight adaptation process.

In the growth process, the accumulated error E of the winning neuron is computed and updated based on the Euclidean distance between the input vector and the weight vector of the winner, according to Equation 3.

$$E_{\text{winner}}(t+1) = E_{\text{winner}}(t) + \|\mathbf{x}_j - \mathbf{w}_{\text{winner}}\| \tag{3}$$

Therefore, new neurons can be added in available free positions around the winning neuron to distribute the accumulated error and better represent the data space. This occurs when the accumulated error of a neuron exceeds GT, and the weight vectors of the new neurons are allocated based on neighboring neurons through extrapolation to preserve the grid's smoothness [Vasighi and Amini 2017]. After all input vectors have been presented, completing one epoch, the growth phase is repeated for a predefined number of epochs, gradually increasing the map size.

3.3. New Neuron Generation

New neurons in the map are always generated from boundary neurons. A boundary neuron is defined as one that has at least one of its four immediate neighboring positions unoccupied, allowing for grid expansion [Alahakoon et al. 2000].

Figure 3 illustrates the process of generating a new neuron. Figure 3a shows the initial grid, Figure 3b shows the grid after the error of a neuron exceeds the GT, and Figure 3c highlights the two possible positions for inserting a new neuron into the map. The new neuron's weights are determined by the weights of its neighbors, making sure that areas with similar characteristics are preserved. Four different cases define how the new weight values are set. These cases are listed below and illustrated in Figure 4.

1. The new neuron has two consecutive old neurons in one of its sides (Figure 4a):

   $$\texttt{if } w_2 > w_1\texttt{, then } w_{\texttt{new}} = w_1 - (w_2 - w_1)$$
   $$\texttt{if } w_1 > w_2\texttt{, then } w_{\texttt{new}} = w_1 + (w_1 - w_2)$$

2. The new neuron is in-between two old neurons (Figure 4b):

   $$w_{\texttt{new}} = \tfrac{w_1 + w_2}{2}$$

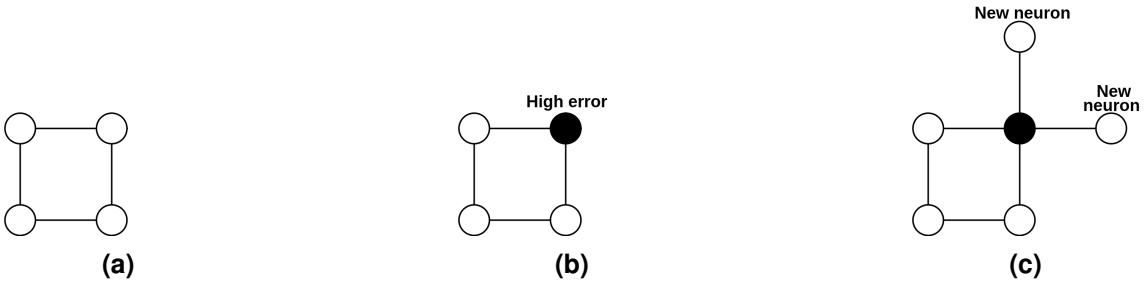


Figure 3. Steps in generating a new neuron. Adapted from [Alahakoon et al. 2000].

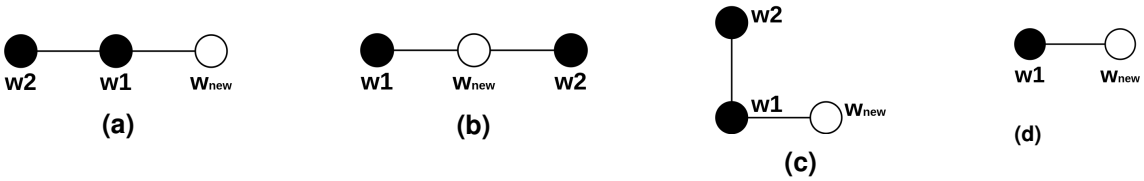


Figure 4. Weight initialization of new neurons. Adapted from [Alahakoon et al. 2000].

3. The new neuron has only one direct neighbor: an older neuron (Figure 4c). In this case, the following rules are applied:

$$\texttt{if } w_2 > w_1\texttt{, then } w_{\text{new}} = w_1 - (w_2 - w_1)$$
$$\texttt{if } w_1 > w_2\texttt{, then } w_{\text{new}} = w_1 + (w_1 - w_2)$$

4. The new neuron has only one neighboring old neuron (Figure 4d). The weight of the new neuron is $W_{\text{new}} = m$, where $m = \frac{r_1+r_2}{2}$, with r_1 and r_2 being the lower and upper values of the range of the weight vector distribution.

3.4. Smoothing Phase

The smoothing phase follows the growing phase and is dedicated to fine-tuning the map structure. During this stage, no new neurons are added. Instead, the goal is to reduce quantization errors and improve the map's ability to represent the input data accurately. This plays a crucial role in stabilizing regions of the map developed in the later stages, thereby enhancing the accuracy and coherence of the map [Alahakoon et al. 2000].

4. Methodology and Proposed Strategies

4.1. Data Collection

The dataset used in this study consists of real firewall logs collected from a large-scale academic network using a Palo Alto 3260 firewall. Over 1.5 million instances were captured, each log entry representing a network traffic session between two devices. The attribute "action" serves as the target class to be predicted, with possible values: allow, deny, reset-both, and drop. Table 1 describes each of these classes. We believe this dataset is a valuable contribution to the data stream and network security communities, as it enables benchmarking with real data. However, due to privacy considerations and institutional security policies, the firewall log dataset cannot be made publicly available, as it contains sensitive information regarding network activity.

Various tests were performed, and the outcomes were analyzed to assess how different features influenced the model's performance and adaptability. The dataset initially

Table 1. Description of firewall action classes

Action	Description
Allow	Authorizes internet traffic according to firewall rules
Deny	Blocks traffic silently (no host notification) and resets TCP connection
Drop	Discards packets without any response to sender
Reset-Both	Terminates connection with reset packets sent to both endpoints

included 113 attributes, the most relevant features of which were chosen based on their impact on enhancing the model's performance. The attributes that produced the best results during this initial evaluation are *Application*, *Subcategory of app*, *Source Country*, *Destination Zone*, *Category*, *IP Protocol*, and *Technology of app*.

4.2. Data Partitioning and Preprocessing

Data from both offline and online phases were preprocessed to ensure consistency and standardization. Categorical variables were converted to numerical values, and *Min-Max* normalization was applied to all features, restricting values to the range [0, 1].

The dataset was split into offline and online phases based on temporal order, simulating a realistic data stream scenario. Table 2 presents the class distributions for both phases, including absolute counts and percentages. Notably, class proportions differ between offline and online sets, reflecting natural variations in network activity.

In the offline phase, a static dataset of 545,365 instances was divided into 300,000 for training (approximately 55.0%) and 245,365 for validation (approximately 45.0%). During this phase, the GSOM is trained on a labeled dataset, allowing the model to learn the initial data structure with access to labels. The model was initialized with a 2x2 neuron grid, expanding as needed according to a growth criterion to capture the data's structure.

For the online phase, classification is performed without any label information: the GSOM processes incoming data in a fully unsupervised manner. This setup mirrors practical data stream scenarios, where labels are unavailable during classification. To simulate real-world conditions, a continuous stream of 1,048,575 instances was used.

Min-Max normalization was applied in blocks of 1,000 records, meaning that rather than processing each instance as it arrives, the system temporarily buffers a batch of 1,000 records, normalizes them based on the minimum and maximum values within the batch, and then proceeds with classification. This strategy ensures computational efficiency and adaptability to dynamic patterns while maintaining consistent normalization within each block, and avoids the need to continuously update global minimum and maximum values across the entire data stream.

4.3. Proposed GSOM Data Stream Strategies

With the normalized data, one of the three proposed strategies was applied to each block independently. These strategies allow the GSOM architecture to adapt flexibly to the evolving characteristics of streaming data.

1. *winner_neurons*: identifies the winner neurons for the incoming data without modifying the offline map structure or its weights, preserving model stability.

Table 2. Class distribution in offline and online datasets

Action	Offline	Offline (%)	Online	Online (%)
Allow	276,490	50.7%	696,162	66.4%
Deny	229,104	42.0%	311,488	29.7%
Drop	39,208	7.2%	39,379	3.8%
Reset-Both	563	0.1%	1,546	0.1%
Total	**545,365**	**100%**	**1,048,575**	**100%**

2. *growing_phase*: dynamically expands the GSOM by adding neurons to better represent emerging patterns in the streaming data.
3. *smoothing_phase*: refines the neuron weights based on new data, adapting the map smoothly without altering its topology.

Initial experiments with multiclass classification showed low accuracy for minority classes, negatively impacting overall performance. To address this, a binary classification approach was adopted, merging the deny, reset-both, and drop actions into a single denied category. This simplification improved model robustness by focusing on distinguishing between allowed and denied traffic.

5. Experiments and Results

This section describes the experiments carried out to assess the performance of the GSOM algorithm in both multiclass and binary classification tasks. The analysis begins by highlighting the difficulties posed by imbalanced datasets in the multiclass setting. To address these challenges, the problem was later reframed as a binary classification task, resulting in significant gains in predictive accuracy.

5.1. Offline Phase Setup

The offline phase is a critical step in initializing the GSOM for subsequent evaluation. This phase involves configuring the GSOM with hyperparameters that define the structure and behavior of the map, ensuring it is well-suited for the classification tasks. Two distinct configurations were employed: for multiclass classification, we used a Spread Factor (SF) of 0.6 with 15 growth epochs and 7 smoothing epochs; for binary classification, we employed $SF = 0.5$ with 20 growth epochs and 10 smoothing epochs. These settings, carefully selected through preliminary experiments, achieve an optimal balance between accuracy and computational efficiency.

5.2. Multiclass Classification Performance

The GSOM evaluation in the multiclass task revealed performance differences across traffic classes, as shown in Table 3. While the model performed well on the frequent *allow* and *deny* categories, it failed to classify the rare *reset-both* and *drop* instances, highlighting challenges with class imbalance, especially in detecting rare network events.

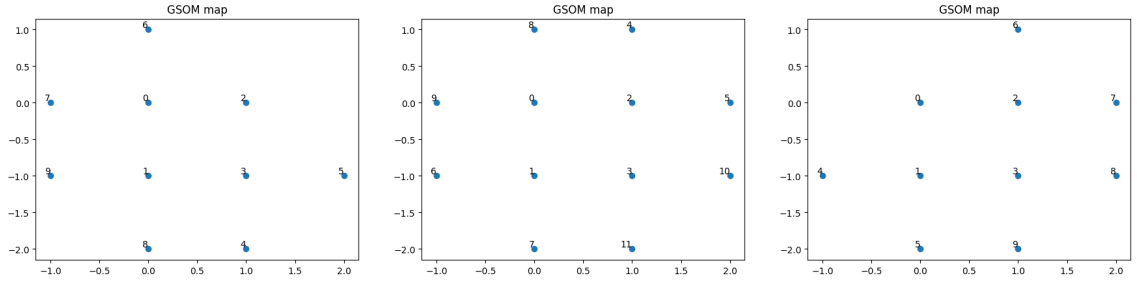
During the online phase, three GSOM adaptation strategies were evaluated with streaming data presented in blocks of 1,000 instances. As shown in Figure 5a, the *winner_neurons* strategy preserved the original map structure. The *growing_phase* strategy (Figure 5b) added two neurons to accommodate new patterns, while the *smoothing_phase* (Figure 5c) refined the neuron positions without expanding the map.

Table 3. Performance of GSOM Strategies in Multiclass Classification

Strategy	Metric	Allow	Deny	Reset-Both	Drop	Accuracy
winner_neurons	F1-Score Macro	23.49%	20.09%	0.00%	0.00%	88.38%
	F1-Score Weighted	62.39%	23.87%	0.00%	0.00%	
growing_phase	F1-Score Macro	21.46%	13.99%	0.00%	0.00%	76.96%
	F1-Score Weighted	57.00%	16.62%	0.00%	0.00%	
smoothing_phase	F1-Score Macro	20.88%	17.53%	0.00%	0.00%	76.76%
	F1-Score Weighted	55.45%	20.82%	0.00%	0.00%	

The *winner_neurons* strategy obtained superior overall performance with 88.38% accuracy. To assess class-specific performance in our imbalanced dataset, we utilized F1-Score metrics. The F1-Score Macro (23.49% for allow) treats all classes equally, exposing severe deficiencies in minority class recognition. In contrast, the F1-Score Weighted (62.39% for allow) reflects overall performance, as it is based on the number of instances per class. Both the reset-both and drop classes achieved 0% across all metrics, revealing significant model limitations in handling class imbalance despite high overall accuracy.

To address class imbalance, we conducted offline experiments using both synthetic oversampling (ADASYN and SMOTE) [Brandt and Lanzén 2021] and a fully balanced training set (125,000 instances per class). Despite these comprehensive balancing approaches, neither method yielded significant improvements for the reset-both and drop classes, with all metrics remaining at 0% across experimental configurations. These results highlight the challenges of handling imbalanced datasets in multiclass settings and underscore the need for alternative strategies to improve classification performance.



(a) *winner_neurons*. **(b)** *growing_phase*. **(c)** *smoothing_phase*.

Figure 5. Neuron mappings for Multiclass GSOM using different strategies.

5.3. Binary Classification Performance

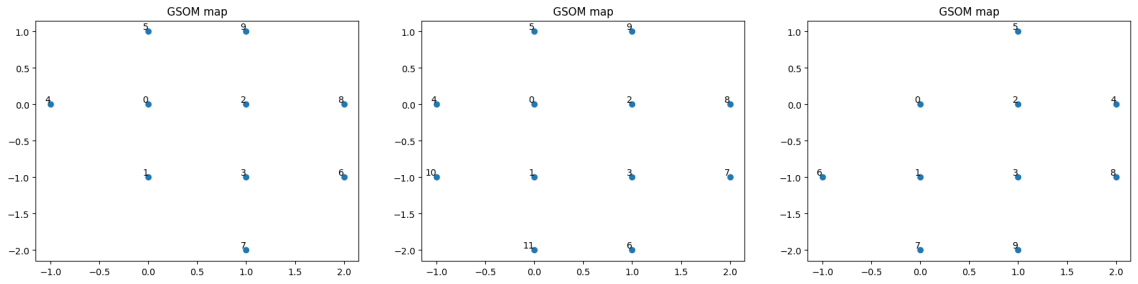
To analyze permitted and denied traffic, the dataset was grouped into classes Allowed and Denied. This reclassification revealed differences in the GSOM evaluation (Table 4), where the model performed well on the *allow* class but showed inconsistent results on the *deny* class across strategies, demonstrating persistent sensitivity to class distribution.

The *winner_neurons* strategy proved most effective in stable data streams, achieving the highest accuracy (91.50%). However, it exhibited clear bias toward the *allow* class, with F1-Score Weighted of 62.35% for Allow versus 28.88% for Deny, confirming significant sensitivity to class imbalance. In contrast, the *growing_phase* strategy expanded the network to accommodate new patterns but showed lower overall performance (79.10%

Table 4. Performance of GSOM Strategies in Binary Classification

Strategy	Metric	Allow	Deny	Accuracy
winner_neurons	F1-Score Macro	46.98%	42.81%	91.50%
	F1-Score Weighted	62.35%	28.88%	
growing_phase	F1-Score Macro	42.93%	30.00%	79.10%
	F1-Score Weighted	57.00%	20.16%	
smoothing_phase	F1-Score Macro	41.75%	37.63%	80.28%
	F1-Score Weighted	55.41%	25.30%	

Accuracy) and weak deny classification (20.16% F1-Score Weighted). This may be attributed to neuron addition in response to sparse data patterns, potentially diluting the network's representational power. Finally, the *smoothing_phase* strategy achieved intermediate results (80.28% Accuracy) through structural refinement, offering better adaptability to variable data streams where new patterns frequently emerge.



(a) *winner_neurons* **(b)** *growing_phase* **(c)** *smoothing_phase*.

Figure 6. Neuron mappings for Binary GSOM using different strategies.

In binary classification, the deny, reset-both, and drop actions are grouped together as a single denied class, since all represent forms of traffic blocking, albeit through different mechanisms. This approach helps address class imbalance and aligns with the primary objective in network operations: distinguishing between permitted and blocked traffic.

5.4. Comparison with Incremental SVM and Adaptive Random Forest (ARF)

To contextualize GSOM's performance, we compared it with two widely used methods: Incremental Support Vector Machine (SVM) and Adaptive Random Forest (ARF). Incremental SVM extends the SVM algorithm for streaming scenarios by incrementally updating its model as new data arrives. It maintains a fixed-size working set of support vectors and dynamically adjusts the decision boundary, making it particularly suitable for continuous data and limited memory environments [Cauwenberghs and Poggio 2000]. ARF, in turn, is an ensemble specifically designed for data streams; it adapts to concept drift by updating its decision trees and replacing underperforming ones using the ADWIN mechanism (Adaptive Windowing). This adaptability makes ARF effective for handling non-stationary environments, such as network traffic analysis [Gomes et al. 2017].

Both approaches were tested using the same dataset in the online phase, with blocks of 1,000 instances. In the case of SVM, an offline phase preceded the online phase. This offline phase used 10% of the data from the online phase and was normalized. All methods were tested with the same attributes to ensure a fair comparison.

In the online phase, our proposed GSOM method operates in a fully unsupervised manner, without access to labels. This means that during classification, no label information is used to update the model. However, for benchmarking purposes, we compare GSOM's performance against supervised methods such as ARF and Incremental SVM, which utilize label information during the online phase to update their models continuously. These supervised methods represent an upper bound on achievable performance when labels are available. It is important to clarify that the comparison with ARF and Incremental SVM is intended to provide context on the potential performance gap between unsupervised and supervised approaches.

Table 5 presents a comparative analysis of GSOM, Incremental SVM, and ARF using F1-Score Macro, F1-Score Weighted and Accuracy for both binary and multiclass classification. In the multiclass scenario, ARF demonstrated superior performance with an overall accuracy of 96.90% and strong results across all classes, including minority ones such as *reset-both* (23.15% F1-Score Macro) and *drop* (13.23% F1-Score Macro). In contrast, GSOM and Incremental SVM showed high accuracy for majority classes but failed to correctly classify minority classes. For the binary scenario, ARF achieved 99.77% accuracy and delivered balanced results across all metrics, maintaining its advantage over Incremental SVM and GSOM, which showed slightly lower performance at 96.36% and 91.50% accuracy, respectively.

This comparison emphasizes the benefits and limitations between supervised and unsupervised approaches. While ARF and Incremental SVM achieved higher accuracy by making use of labeled data, GSOM provides a competitive and interpretable alternative in situations where labeled data is limited or unavailable. The results suggest that GSOM is a promising tool for classifying firewall logs in data streams, especially when adaptability and interpretability are key requirements.

Table 5. Binary and multiclass classification: model comparison

Type	Metric	Allow	Deny	Reset-Both	Drop	Accuracy
winner_neurons (Multiclass)	F1-Score Macro	23.49%	20.09%	0.00%	0.00%	88.38%
	F1-Score Weighted	65.37%	24.57%	0.00%	0.00%	
Incremental SVM (Multiclass)	F1-Score Macro	23.47%	21.12%	0.00%	0.07%	88.76%
	F1-Score Weighted	62.32%	25.10%	0.00%	0.01%	
ARF (Multiclass)	F1-Score Macro	24.94%	23.74%	23.15%	13.23%	96.90%
	F1-Score Weighted	66.22%	28.21%	0.14%	1.99%	
winner_neurons (Binary)	F1-Score Macro	46.98%	42.81%	–	–	91.50%
	F1-Score Weighted	62.35%	28.88%	–	–	
Incremental SVM (Binary)	F1-Score Macro	48.62%	47.34%	–	–	96.36%
	F1-Score Weighted	64.55%	31.82%	–	–	
ARF (Binary)	F1-Score Macro	49.91%	49.82%	–	–	99.77%
	F1-Score Weighted	65.45%	34.32%	–	–	

6. Conclusion

This study presented an innovative approach leveraging the GSOM for classifying firewall logs in data streams, highlighting the potential of this technique in network security scenarios. The application of GSOM demonstrated its capability for continuous adaptation to dynamic traffic patterns, an essential characteristic in environments where traffic behavior can change rapidly due to emerging threats or changes in network usage.

Although the dataset covers an extended period and a large volume of network activity, we did not observe significant concept drift. The class distributions remained relatively stable over time, with no abrupt or gradual shifts that would indicate changes in the data-generating process. Consistent with this stationarity, the evolution of the GSOM during the online phase also reflected a stable growth pattern, without abrupt expansions or major topological changes. This suggests that, in the absence of concept drift, the network adapts smoothly to the incoming data stream, which both simplifies the interpretation of results and highlights the robustness of the evaluated models under stationary conditions. Moreover, the use of the GSOM provided valuable insight into the scalability and flexibility of algorithms designed for firewall log classification. The ability of the GSOM to dynamically expand its neuron grid to accommodate new patterns without requiring manual reconfiguration offers significant advantages in real-world scenarios, where rapid and effective responses are critical.

Although supervised methods such as ARF and Incremental SVM have achieved greater precision when using labeled data, GSOM operates effectively in a completely unsupervised environment with infinite label latency, offering a competitive alternative where adaptability and interpretability are prioritized. This study opens paths for future research, such as the exploration of approaches that combine unsupervised methods, like GSOM, with supervised techniques to improve classification performance for minority classes in the multiclass approach.

In summary, this study highlights the potential of GSOM as a robust and adaptable tool for network security applications. While supervised methods like ARF and Incremental SVM offer higher accuracy, GSOM provides a viable and interpretable alternative for scenarios where labeled data is limited or unavailable.

Acknowledgment

This study was financed by the São Paulo Research Foundation (FAPESP) grant 2022/02981-8 and the National Council for Scientific and Technological Development (CNPq).

References

Alahakoon, D., Halgamuge, S. K., and Srinivasan, B. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on neural networks*, 11(3):601–614.

Aljabri, M., Alahmadi, A. A., Mohammad, R. M. A., Aboulnour, M., Alomari, D. M., and Almotiri, S. H. (2022). Classification of firewall log data using multiclass machine learning models. *Electronics*, 11(12):1851.

Allagi, S. and Rachh, R. (2019). Analysis of network log data using machine learning. In *IEEE 5th International Conference for Convergence in Technology (I2CT)*, pages 1–3.

Barnawi, A., Gaba, S., Alphy, A., Jabbari, A., Budhiraja, I., Kumar, V., and Kumar, N. (2023). A systematic analysis of deep learning methods and potential attacks in internet-of-things surfaces. *Neural Comput. Appl.*, 35(25):18293–18308.

Brandt, J. and Lanzén, E. (2021). A comparative review of smote and adasyn in imbalanced data classification.

Casarotto, P. and Cerri, R. (2024). Growing self-organizing maps for multi-label classification. In *Brazilian Conference on Intelligent Systems*, pages 33–48. Springer.

Cauwenberghs, G. and Poggio, T. (2000). Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13.

Das, M., Pratama, M., Zhang, J., and Ong, Y. S. (2020). A skip-connected evolving recurrent neural network for data stream classification under label latency scenario. In *AAAI Conference on artificial intelligence*, volume 34, pages 3717–3724.

Dua, M. et al. (2019). Machine learning approach to ids: A comprehensive review. In *3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 117–121.

Ertam, F. and Kaya, M. (2018). Classification of firewall log files with multiclass support vector machine. In *6th International symposium on digital forensic and security (ISDFS)*, pages 1–4.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., and Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106:1469–1495.

Haripriya, D., Abou Ghaly, M., Deepak, A., Sharma, K., Chandre, S., Bajaj, K., and Shrivastava, A. (2024). A comparative study on online machine learning techniques for network traffic streams analysis. *Int. J. Intell. Syst. Appl. Eng.*, 12(13s):09–19.

Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.

Kohonen, T. (2001). Self-organizing maps, ser. *Information Sciences. Berlin: Springer*, 30.

Liao, Y. and Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, 21(5):439–448.

Patgiri, R., Varshney, U., Akutota, T., and Kunde, R. (2018). An investigation on intrusion detection system using machine learning. In *IEEE SSCI*, pages 1684–1691.

Qu, X., Yang, L., Guo, K., Ma, L., Sun, M., Ke, M., and Li, M. (2021). A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile networks and applications*, 26:808–829.

Rolemberg, T. M. (2021). Aplicação de conceitos de redes complexas para a descoberta de formação de grupos em mapas auto-organizáveis.

Sharma, D., Wason, V., and Johri, P. (2021). Optimized classification of firewall log data using heterogeneous ensemble techniques. In *ICACITE*, pages 368–372.

Ucar, E. and Ozhan, E. (2017). The analysis of firewall policy through machine learning and data mining. *Wireless Personal Communications*, 96:2891–2909.

Vasighi, M. and Amini, H. (2017). A directed batch growing approach to enhance the topology preservation of self-organizing map. *Applied Soft Computing*, 55:424–435.

Zheng, H., Wang, H., and Black, N. (2008). Human activity detection in smart home environment with self-adaptive neural networks. In *IEEE International conference on networking, sensing and control*, pages 1505–1510.