applied sciences

Article

Graph-Based Multi-Label Classification for WiFi Network Traffic Analysis

Giuseppe Granato [1], Alessio Martino [2,*], Andrea Baiocchi [1] and Antonello Rizzi [1]

[1] Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", Via Eudossiana 32, 00184 Rome, Italy
[2] Department of Business and Management, LUISS University, Viale Romania 32, 00197 Rome, Italy
* Correspondence: amartino@luiss.it; Tel.: +39-06-8522-5957

Abstract: Network traffic analysis, and specifically anomaly and attack detection, call for sophisticated tools relying on a large number of features. Mathematical modeling is extremely difficult, given the ample variety of traffic patterns and the subtle and varied ways that malicious activity can be carried out in a network. We address this problem by exploiting data-driven modeling and computational intelligence techniques. Sequences of packets captured on the communication medium are considered, along with multi-label metadata. Graph-based modeling of the data are introduced, thus resorting to the powerful GRALG approach based on feature information granulation, identification of a representative alphabet, embedding and genetic optimization. The obtained classifier is evaluated both under accuracy and complexity for two different supervised problems and compared with state-of-the-art algorithms. We show that the proposed preprocessing strategy is able to describe higher level relations between data instances in the input domain, thus allowing the algorithms to suitably reconstruct the structure of the input domain itself. Furthermore, the considered Granular Computing approach is able to extract knowledge on multiple semantic levels, thus effectively describing anomalies as subgraphs-based symbols of the whole network graph, in a specific time interval. Interesting performances can thus be achieved in identifying network traffic patterns, in spite of the complexity of the considered traffic classes.

Keywords: machine learning; communication networks; granular computing; IEEE 802.11; graphs; sequences; graph neural networks; genetic algorithms

check for updates

Citation: Granato, G.; Martino, A.; Baiocchi, A.; Rizzi, A. Graph-Based Multi-Label Classification for WiFi Network Traffic Analysis. *Appl. Sci.* **2022**, *12*, 11303. https://doi.org/10.3390/app122111303

Academic Editors: Cristina Portalés Ricart, João M. F. Rodrigues and Pedro J. S. Cardoso

Received: 20 August 2022
Accepted: 4 November 2022
Published: 7 November 2022

1. Introduction

Computational Intelligence is well established and pervasive in technological and social contexts, providing the highest benefit to address those problems for which a useful mathematical modeling is not available or too complex to be practical. Network security is one such field. Specifically, traffic monitoring and analysis are key security functions of modern networks that lend themselves naturally to being implemented by exploiting computational intelligence tools.

Broadly speaking, traffic monitoring and analysis consist of observing network traffic flows and applying classification to identify different types of flows, e.g., with respect to management of differentiated quality of service or in relation with security functions. As for the latter application, identification of anomalous or malicious traffic is the primary goal. Such traffic may consist of isolated packets or complex flows, depending on the purpose of the attack and its complexity. The aim of automated tools to identify anomalous and malicious traffic can be:

- to assist network operator staff in network management and safeguarding of services;
- to filter/mark/block suspicious traffic;
- to identify different attacks for network diagnostics and reporting or to strengthen defenses in an optimized way, given the available resources;

- to provide robust self-management of networks.

Network traffic attacks are most prominent in wireless networks, e.g., WiFi networks. The pervasive nature of this kind of access, and in general, the increasing reliance on access networks for accomplishing more and more tasks both in professional and personal contexts, set a strong demand for robust networks, able to identify and block even sophisticated attacks in an automatic way. The widely available computing and processing power of networking elements inevitably shifts the highest interest in machine learning-based approaches. This kind of approach is further fostered by considering that identifying and classifying network traffic attacks does not lend itself to mathematical modeling as laws of Physics would do. On the other hand, anomalous and even malicious behavior can be revealed with respect to regular traffic, as a long series of works over the last two decades have shown. In other words, the information to identify attacks is essentially there, yet it is not easy at all to extract the relevant information and to exploit it.

We have already shown in [1,2] that many attacks on WiFi networks can be identified with very high success probability by means of classification algorithms looking at a single packet (more precisely, MAC frame in the WiFi context) at a time. When scanning the traffic over the wireless channel, each single packet can be labeled as being regular traffic or belonging to one of several different attack types. For many attack types, this kind of classification is very promising, exceeding success ratio levels of 90% or 95%.

There are, however, other attacks that cannot be easily identified. This depends mostly on the fact that those attacks are mounted by means of several steps, each one involving one or possibly more packet exchange(s) between the attacker node and its victims. What is lacking in single-packet analysis is a way to unveil the *structure* of the attack evolution over multiple packets, thus taking care of modeling relations between entities in the specific input domain.

To overcome this limitation, in the present work, we consider sequences of packets captured from the network traffic. In addition, however, rather than working on sequences of packets (as already explored in [3]), we derive graphs from ordered sequences of packets being transmitted over the network and use elements of those graphs to build an alphabet of symbols to encode the information carried in the traffic stream useful to classify attacks.

The purpose of this work is to compare a multi-labelled Granular Computing-based approach with a conventional Graph Neural Network, in order to perform network traffic classification. Specifically, we aim to study network anomalies from a higher semantic level, described by network graphs directly built from buffered frames captured from a monitor host. For this purpose, the main contributions of this paper are described as follows:

- We introduce a structured domain representation by a preprocessing strategy able to extract weighted network graphs directly from buffered frames;
- We formally define the network anomaly detection problem as a non-exclusive supervised problem in graph domain;
- We introduce a Granular Computing approach in the graph domain aimed at network traffic anomaly detection and automatic analysis;
- Furthermore, we provide an extensive analysis of two consolidated approaches revised to work in a multi-label context, the aforementioned Granular Computing-based one and a genetically optimized Graph Neural Network;
- Next, we show experimental results and performance analysis of such two consolidated approaches for graph machine learning, applied to network traffic classification.
- Lastly, we describe the model interpretability and knowledge discovery capabilities of the considered approaches, as applied to the specific application domain of network traffic anomaly detection.

These contributions are particularly important in order to build prediction systems that are able not only to achieve state-of-the-art performances as other analyzed works, but also to keep focus on improving models' human interpretability, obtaining further semantic insights on the analyzed data.

The remainder of the paper is organized as follows: In Section 2, we summarize a review of recent works focusing on machine learning on structured domains and anomaly detection in telecommunications networks, particularly concerning IEEE 802.11 networks. Next, in Section 3, we describe the dataset used, introducing the dissimilarity measures adopted in the starting features space and in the graph structured domain, and explaining the sequences to graphs procedure along with the supervised problems we are going to test. In Section 4, we detail how the GRALG classification system works, by formally recalling each key component behavior and describing its application on the target dataset. In Section 5, we perform an overview of the GNN architecture used as baseline to compare results with GRALG. Furthermore, in Section 6, we outline results analysis, comparing performances obtained with other state-of-the-art papers and previous works. Lastly, in Section 7, we sum up details concerning proposed solutions, along with results analysis obtained so far, and provide further points for future works.

2. Related Works

Recent developments in communication networks allowed us to build faster and more reliable communication services which can be accessed more easily and safely thanks to several security protocols available.

The complexity of this new kind of technology requires new approaches to perform monitoring, anomaly detection and intrusion detection both to evaluate Quality of Service Key Performance Indicators [4,5] and to check activity from potential attackers [6,7].

Furthermore, since this aspect concerns both wired and wireless communications technologies, new paradigms are studied in both ways, such as Software Defined Networking (SDN). In fact, Segura et al. in [6] provide a study concerning the usage of the SDN paradigm to perform both centralized and distributed intrusion detection in resource-constrained wireless networks.

High data rate, low latency, and high volumes of data require a set of processing techniques that should be easily scalable both in terms of capacity and functionality. For this purpose, adaptive signal processing and machine learning provide exactly these kinds of capabilities, thanks to the possibility to build models from sets of examples and multiple ways to scale to previously unseen situations.

Dietz et al., for example, studied machine learning-based performance prediction algorithms for SDN environments [8], providing case studies both on real and synthetic networks. Results taken from a centralized and a distributed architecture show the feasibility to perform network status prediction based on the considered features.

In this context, wireless communications are a flexible technology able to connect different kinds of devices, from enterprise environments to industrial Internet of Things. Both 5G and IEEE 802.11 WiFi standards offer key capabilities for deployment in such scenarios, so they are subject to constant security reviews.

In fact, in [9], in order to perform a security analysis of the 5G Core Network components, the authors employ a machine learning approach to assist in the definition and processing of attack graphs. In this way, they have been able to find multiple vulnerabilities and novel attacks on the Authentication and Key Agreement protocol.

Concerning IEEE 802.11 WiFi networks, attack analysis and intrusion detection have been particularly studied.

Starting from the publication of the Aegean WiFi Intrusion Detection dataset in [10], multiple works have tackled the problem. Notably, in [11], Kolias et al. proposed a new approach based on Particle Swarm algorithms able to achieve an interesting level of accuracy and, at the same time, generate a set of *IF-THEN* human-readable rules.

Despite multiple works having achieved interesting results by processing individual network packets and frames (e.g., by means of several techniques not only related to Deep Learning [12,13]), processing network flows is still required to achieve state-of-the-art performance able to handle deployments in real scenarios [14].

Moreover, it is important to notice the changes, which are constantly evolving, that network technology and services are subject to, due to evolving quality of service and security requirements. Hence, computational intelligence models further require increasingly-deeper knowledge discovery capabilities, simplifying human interpretability and support in decision-making. As an attempt to overcome this limitation, in this work, we further study these kinds of problems, redefining the original supervised problem in the graph domain. This allows us to model the relationships linking raw WiFi frames, with a Granular Computing-based approach able to describe network status on multiple semantic levels. Graph signal processing and machine learning are a florid research field when it comes to dealing with structured data such as graphs. These kinds of approaches have been proven successful in multiple scenarios, from bioinformatics to social networks [15].

In our case, we aim to represent the specific status of the target WiFi network as a graph, defined in a precise time interval. This graph represents what is happening in the network in terms of multiple parameters (such as exchanged data and delay).

To do so, we employ two consolidated techniques: the first one based on a Granular Computing approach [16], and the second one based on Graph Neural Networks [17].

In this way, we can perform a deep analysis of the two different approaches in terms of:

- classification performances;
- model interpretability;
- knowledge discovery and extraction capabilities.

3. Dataset Description

The present work is the result of a set of research efforts that began in [1], aiming to improve automatic network traffic recognition in different situations. Specifically, in this case, we aim to recognize attacks on the network infrastructure by defining a supervised problem in a non-metric space. The elements of the input space are graphs representing how information flows through the network in a specific time frame.

Simulations have been performed on the Aegean WiFi Intrusion Detection Dataset (http://icsdweb.aegean.gr/awid/ accessed on 20 August 2022) [10] that is a rigorous attempt to capture the inner workings of a WiFi Small Office Home Office (SOHO) network environment. Basically, it includes heterogeneous legit devices (such as smartphones, workstations and other smart devices), along with the attacker WiFi station and monitoring device to capture frames exchanged by other stations.

The WiFi network setup used in the collection of AWID data employs the WEP security protocol to secure information traffic. This choice arises from the need to be able to deeply study multiple kinds of attacks while keeping frame capture and labeling procedures as simple as possible. This dataset gives valuable insights into the anomaly and attack detection problem in WiFi networks thanks to the large number of details included for each captured MAC frame (i.e., more than one hundred features). Moreover, fourteen different types of attacks have been implemented as well as realistic production of legitimate traffic flows. The extensiveness of considered scenarios and the abundant data made available with the AWID dataset make it a useful tool to assess algorithms validation both in terms of classes recognition capabilities and in terms of management of a huge amount of data.

Frames are divided into fifteen classes, hierarchically organized in the following four macro-classes:

- **Flooding**: Attacks of this kind exploit unauthenticated management and control mechanisms [18] by forging vast amounts of frames in order to disrupt communications by flooding the available medium. These kinds of attacks can be used to build more sophisticated strategies, such as in WiFi *Impersonation* attack scenarios.
- **Frame Injection**: This macro-class includes attacks that exploit vulnerabilities in the WEP security cryptographic protocols. More in depth, the attacker uses the Access Point (AP) as an oracle to check cryptographic correctness of frames. Repeating this process multiple times, the attacker can retrieve cryptographic key material information.

- **Impersonation**: This set of attacks is a key component to prepare more complex adversary scenarios, where the attacker tries to impersonate the access point by exploiting similar vulnerabilities as described before.
- **Normal**: Lastly, this macro-class collects network traffic of legitimate users connected through multiple smart devices.

Kolias et al. [10] forged and captured the aforementioned attacks using the well-known penetration testing toolchain provided by the Kali Linux project (e.g., *aircrack-ng* suite (https://www.aircrack-ng.org accessed on 20 August 2022) and MDK3 (https://tools.kali.org/wireless-attacks/mdk3 accessed on 20 August 2022)).

The present work has been developed by employing a subset of the AWID dataset, containing patterns of 15 classes (which distribution has been detailed in Table 1).

Table 1. Distribution of the 14 attack classes within the three attack macro-classes.

Flooding	Frame Injection	Impersonation
`beacon`	`arp`	`evil_twin`
`de-authentication`	`chop_chop`	`cafe_latte`
`disassociation`	`fragmentation`	`hirte`
`amok`		
`power_saving`		
`probe_request`		
`rts`		
`cts`		

3.1. Dissimilarity Measure

Captured MAC frames (generically referred to also as "packets" in the following) include a variety of features describing different fields with different numerical needs.

Designing an appropriate dissimilarity measure (non necessarily metric) for the problem and data at hand is a key facet for the success of any pattern recognition system. Furthermore, it is possible to define parametric dissimilarity measures thanks to a weighting vector $\mathbf{w}$ [19].

As shown in Table 2, we consider patterns composed by a mixture of numerical (integer- or real-valued), discrete nominal and Boolean features. Furthermore, for the purpose of this work, numerical features are assumed to be normalized in $[0, 1]$. To perform processing on patterns (i.e., packets) defined in the starting features space, we adopt the following dissimilarity measure (already proposed in [1,2,20]) ad-hoc tailored to this heterogeneous structure. Let $\mathbf{x}$ and $\mathbf{y}$ be two generic patterns, then:

- if the ith feature is numerical, the dissimilarity between homologous features reads as their absolute difference:
$$d_i(\mathbf{x}_i, \mathbf{y}_i) = |\mathbf{x}_i - \mathbf{y}_i| \tag{1}$$
- if the ith feature is nominal or Boolean, the dissimilarity between homologous features reads as the discrete distance:
$$d_i(\mathbf{x}_i, \mathbf{y}_i) = \begin{cases} 1 & \mathbf{x}_i \neq \mathbf{y}_i \\ 0 & \mathbf{x}_i = \mathbf{y}_i \end{cases} \tag{2}$$

Finally, if we allow each feature to be weighted independently by means of a weighting vector $\mathbf{w}$, the overall dissimilarity measure reads as:

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{w}_i \cdot d_i(\mathbf{x}_i, \mathbf{y}_i) \tag{3}$$

where n is the number of considered features. Since each dissimilarity measure between homologous features (see Equations (1) and (2)) assumes values in $[0, 1]$, it is straightforward to prove that Equation (3) also assumes values in range $[0, 1]$.

3.2. Dynamic Time Warping

The preprocessing phase requires dealing with structured data in the form of sequences of packets. In order to be able to compare these kinds of objects and evaluate distances, we introduce the Dynamic Time Warping (DTW) algorithm [21]. The DTW algorithm is a function $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$ which maps a pair of sequences to a non-negative real-valued dissimilarity value (that can be further normalized). Formally, it is defined as the solution of the following optimization problem:

$$d^{(\mathrm{DTW})}(s,t) = \min_{\pi \in \mathcal{P}(s,t)} \left(\sum_{(i,j)\in\pi} d(s_i, t_j) \right) \tag{4}$$

where $d(\cdot,\cdot)$ is a suitable cost function (i.e., a dissimilarity measure between sequence's elements), π is the alignment path, and $\mathcal{P}(s,t)$ is the set of all admissible paths between sequences s and t. Such optimization problem can be solved in multiple ways, among which we choose the dynamic programming one (synthesized in Algorithm 1). Recalling that sequences are made of structured data that represent WiFi frames, the dissimilarity measure $d(\cdot,\cdot)$ in Equation (4) is the one described in Section 3.1 (namely, Equation (3)):

Algorithm 1: DTW Routine

```
double DTWDistance(s: array [1..n], t: array [1..m]) {
DTW := array [0..n, 0..m];
for i = 0,...,n do
    for j = 0,...,m do
        DTW[i, j] := infinity;
    end
end
for i = 1,...,n do
    for j = 1,...,m do
        cost := d(s[i], t[j]);                    /* d() in Equation (3) */
        DTW[i, j] := cost + min(DTW[i-1, j],
                                DTW[i, j-1],
                                DTW[i-1, j-1]);
    end
end
return DTW[n, m];
}
```

3.3. Graph Edit Distance

Graph Edit Distances (GEDs) [15,22,23] are dissimilarity measures belonging to the wide family of edit distances, extending a set of consolidated and well-known dissimilarity measures defined on sequences (e.g., the Levenshtein distance [24]) towards graphs. If in a discrete finite domain, such as the string domain, the edit distance can be simply evaluated as the minimum number of insertions, substitutions and deletion of letters needed to transform a given string into a target one, the same does not hold where annotations of atomic objects can be equipped with arbitrary data structures. In other words, the "exactness" of a match is always guaranteed in the string domain (i.e., two letters are the same or not), but it is rarely guaranteed in the graph domain (i.e., only in cases when nodes and/or edges are equipped with categorical attributes).

Amongst the several GEDs available in the literature, our choice fell on the node Best Match First (nBMF) GED. In short, the nBMF similarity measure starts by matching (in a greedy manner) most similar nodes first, where the similarity between nodes is evaluated thanks to a suitable dissimilarity measure $d^{\mathrm{node}}(\cdot,\cdot)$. Then, possible edges between each pair of previously-found similar nodes are searched for in both graphs to be compared: if found, this counts as an edge substitution whose cost is given by a suitable dissimilarity measure $d^{\text{edge}}(\cdot, \cdot)$ defined on the edge labels domain. The difference between the order of the two graphs accounts for the nodes' deletion and insertion, whereas the existence of induced edges in only one of the two graphs yields an edge deletion or insertion. Full mathematical details on nBMF can be found in [25].

As will be clear in Section 3.4, in this work, we deal with graphs whose nodes are equipped with MAC addresses and whose edges are equipped with a 6-elements real-valued vector. Therefore, we let $d^{\text{node}}(n_1, n_2)$ be the discrete distance (i.e., the dissimilarity is 0 or 1 depending on whether the two nodes n_1 and n_2 share the same MAC address or not) and we let $d^{\text{edge}}(e_1, e_2)$ be the ℓ_1-norm between the feature vectors of two given edges e_1 and e_2.

3.4. Buffering and Graphs Building Procedure

As anticipated, many research works focused on single-frame analysis [1,2,10,14,20], mainly highlighting that a subset of attacks can be effectively identified on a per-frame basis. As instead, in this work, we perform a study of the anomaly detection problem in WiFi networks expressed in the graph domain.

To this end, we define a preprocessing strategy that builds a new dataset, starting from a simple First-In-First-Out (FIFO) queue that collects B consecutive packets captured on the channel. We use this buffer as a B-length window that we slide over the time series of captured packets in order to build a new dataset made of sequences of packets. Meta-data are attached to each captured packet in the original dataset, assessing whether that packet belongs to legitimate traffic or, if it is an attack packet, what kind of attack it belongs to. Since each sequence could contain packets belonging to different classes, the new sequence-based dataset will be labeled in a non-exclusive manner, thus associating multiple labels to each sequence.

At this point, we introduce the graph building procedure, which aims to extract the information brought by sequences built from the FIFO buffer and to build a corresponding graph for each sequence. Let us consider a sequence s_i of length B. Formally, we define the mapping $\Gamma : \mathcal{S} \rightarrow \mathcal{G}$, which associates the set of sequences $\mathcal{S}$ with the set of graphs $\mathcal{G}$. Each graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ includes nodes corresponding to all MAC addresses found in frames that belong to s_i, and edges defined by the source–destination MAC addresses pairs found in the frames of the sequence s_i. Each edge, then, is weighted with a vector of elements described by the following equation:

$$w^{(e)}(i,j) = \begin{bmatrix} p_{\text{frame}}(i,j) & p_{\text{payload}}(i,j) & a_{\text{RX}}(i,j) & n_{\text{seq}}(i,j) & n_{\text{frag}}(i,j) & \Delta t(i,j) \end{bmatrix} \tag{5}$$

where

- i and j refer to MAC addresses found in frames belonging to s_i;
- $p_{\text{frame}}(i,j)$ and $p_{\text{payload}}(i,j)$ are respectively the normalized mean length and payload size of frames sent on that link;
- $a_{\text{RX}}(i,j)$ is the normalized mean of the received signal strength for that frame (taken from the field *radiotap.dbm_antsignal* listed in Table 2);
- $n_{\text{seq}}(i,j)$ and $n_{\text{frag}}(i,j)$ are respectively the normalized mean sequence number and fragment number for data frames on that link;
- $\Delta t(i,j)$ is the average time of arrival of frames on that link.

The whole sequence-to-graphs transformation is sketched in Figure 1. Lastly, by using a multi-label iterative stratification procedure [26], we perform training ($\mathcal{S}_{\text{tr}}$), validation ($\mathcal{S}_{\text{vd}}$), and test ($\mathcal{S}_{\text{ts}}$) set splitting for the proposed experiments. Brief dataset statistics are resumed in Table 3.

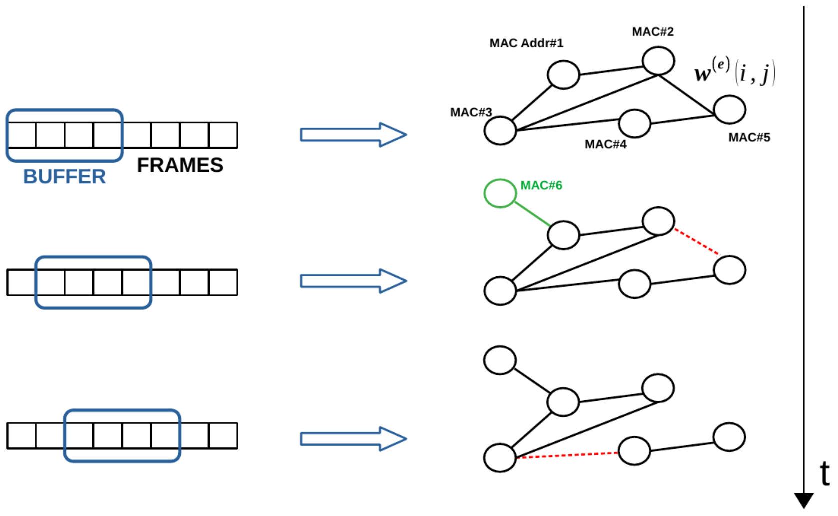

Figure 1. Single instances to graph preprocessing.

Table 2. List of packets' features.

Attribute Name	Data Type	Description
frame.len	Numerical	Frame length
radiotap.dbm_antsignal	Numerical	Received Signal Strength
wlan.fc.type	Nominal	Frame Type
wlan.fc.subtype	Nominal	Frame Subtype
wlan.fc.ds	Nominal	Distribution System status
wlan.fc.frag	Boolean	More Fragments
wlan.fc.pwrmgt	Boolean	Power management
wlan.fc.order	Boolean	Order flag
wlan.duration	Numerical	Duration
wlan.ra	Nominal	Receiver address
wlan.da	Nominal	Destination address
wlan.ta	Nominal	Transmitter address
wlan.sa	Nominal	Source address
wlan.bssid	Nominal	BSS ID
wlan.frag	Numerical	Fragment number
wlan.seq	Numerical	Sequence number
wlan.fcs_good	Boolean	FCS correctness
wlan_mgt.fixed.listen_ival	Numerical	Listen Interval
wlan_mgt.fixed.timestamp	Numerical	Timestamp
wlan_mgt.fixed.beacon	Numerical	Beacon Interval
wlan_mgt.fixed.reason_code	Nominal	Reason code
wlan_mgt.fixed.sequence	Numerical	Starting Sequence Number
wlan.wep.iv	Nominal	Initialization Vector
wlan.wep.icv	Nominal	Integrity Check Value
data.len	Numerical	Data Length

3.5. Training Set Compression in Sequences Domain

The amount of (structured) data contained in the training set boosts the complexity of the whole training procedure. In order to improve processing speed and overall accuracy, we aim to reduce this complexity by performing a compression stage on the training set.

The compression strategy consists of the execution of the Basic Sequential Algorithmic Scheme (BSAS) [27]. BSAS is a clustering algorithm that aims to isolate per-label cluster representative patterns [28] to build a new compressed training set.

This step is performed in the sequence domain since the dissimilarity measure in Section 3.1 and the DTW in Section 3.2 for this kind of structured data do not require specific critical parameter tuning, especially if compared to the GED in Section 3.3.

Table 3. Patterns and graphs' distribution. Per label single instances' distribution and multi-labeled network graph splits.

Class Name		$\mathcal{S}$	
`normal`		530,785	
`amok`		477	
`arp`		13,644	
`beacon`		599	
`cafe_latte`		379	
`chop_chop`		2871	
`cts`		1759	
`deauthentication`		4445	
`disassociation`		84	
`evil_twin`		611	
`fragmentation`		167	
`hirte`		19,089	
`power_saving`		165	
`probe_request`		369	
`rts`		199	
Multi-Label Graphs	$\mathcal{S}_{\text{tr}}^{(g)}$	$\mathcal{S}_{\text{vd}}^{(g)}$	$\mathcal{S}_{\text{ts}}^{(g)}$
$B = 10$	414,456	46,051	115,127
$B = 20$	414,454	46,050	115,125
$B = 50$	414,444	46,047	115,119

The BSAS hyperparameters have been chosen as follows:

- maximum cluster radius $\theta = 0.05$;
- maximum number of allowed clusters $Q_{\max} = \{800, 10{,}000\}$.

Two values of $Q_{\max}$ have been used in order to study two kinds of scenarios that is a strongly compressed training set and a larger one.

3.6. *Supervised Exclusive and Non-Exclusive Problems Definitions*

In this section, we formally introduce the supervised problems that we are going to explore and analyze in this paper.

Due to the complexity of non-exclusive (multi-label) supervised problems, we need to define a baseline problem in order to be able to compare testing results. To this end, we aim to verify classification accuracy of the proposed approach in a basic situation where the main objective consists of distinguishing normal traffic from anomalies.

The problem is defined as the search of the best set of hyperparameters so that we can effectively approximate the target process $p : X \to \mathcal{L}$, which maps to each instance of the input space X, a specific label of the set $\mathcal{L} = \{normal, anomaly\}$.

Hence, all traffic flows belonging to any kind of anomaly/attack form a single class in this problem statement. In this way, we can perform a starting analysis of the Granular Computing capabilities of the GRALG classification system, trying to isolate key substructures of complex, heterogeneous network traffic.

To further improve the above definition, we consider that the analysis on different semantic levels of telecommunication networks implies the association of a set of labels to each kind of structured data that represents multiple network packets.

For this reason, we tackle a new supervised problem $p_{ML} : X_g \to \mathcal{M}$ where each network graph of the starting structured space X_g is mapped to a multi-label (a specific set of labels) that belongs to $\mathcal{M}$. In this way, we build a system that automatically learns this association and tries to give insight into label relationships.

Lastly, by considering the different nature of the two supervised problems, we take into account the mean accuracy and the Jaccard similarity score as performance metrics. The rationale behind these choices follows. Since the number of instances per class are unbalanced, a mean accuracy measure is required in order to avoid implicit weighting phenomena. On the other hand, the non-exclusive supervised problem requires a specific performance metric, namely the Jaccard similarity score [29], which is defined as follows:

$$J_{i,j} = \frac{\omega_i^{(m)} \bigcap \omega_j^{(m)}}{\omega_i^{(m)} \bigcup \omega_j^{(m)}} \tag{6}$$

The definition holds for a specific pair of multi-labels $\omega_i^{(m)}$. This measure can be extended to the entire dataset, by taking the mean value of all the measures obtained.

4. The GRALG Classification System

Granular Computing Approach for Labelled Graphs (GRALG) is a classification system suitable for dealing with (fully) labelled graphs grounded on the Granular Computing paradigm [30,31]. GRALG was originally proposed in [32] and later improved in [33–38], addressing some computational drawbacks in the original implementation. As anticipated, GRALG follows the Granular Computing paradigm, hence it aims at automatically extracting pivotal mathematical entities known in the technical literature as *information granules*, able to characterize the data at hand as much as possible. GRALG exploits such information granules in order to cast the graph classification problem towards a Euclidean space, where standard statistical pattern recognition tools can be used without alterations. To accomplish these tasks, GRALG relies on four macro-blocks, described in the following Sections 4.1–4.4 and highlighted in Figure 2, whereas, in Section 4.5, we describe how the four blocks cooperate in order to synthesize the classification model. In the following, we give an overview of the GRALG classification system, highlighting its main characteristics, and we refer the interested reader to our previous works for a thorough description. Let us recall the graph dataset $\mathcal{S}$ to be partitioned into three non-overlapping sets, namely the training ($\mathcal{S}_{\text{tr}}$), validation ($\mathcal{S}_{\text{vl}}$) and test set ($\mathcal{S}_{\text{ts}}$).

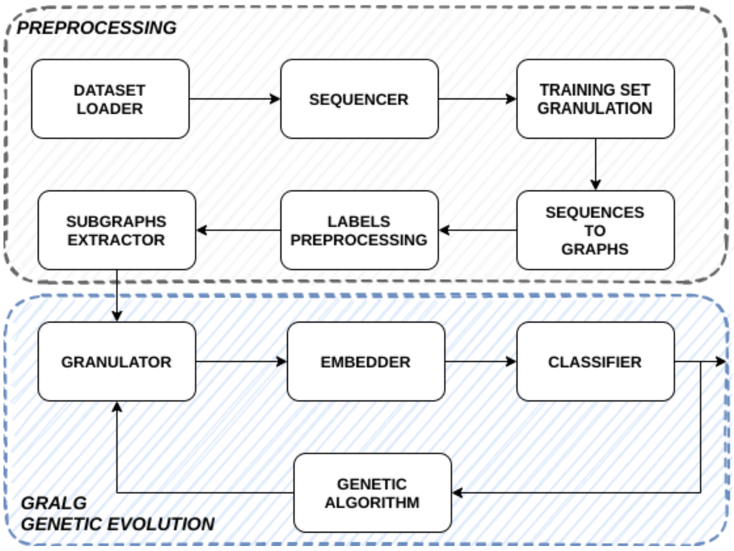

Figure 2. GRALG optimization architecture.

4.1. Extractor

The Extractor block is in charge of extracting from graphs in $\mathcal{S}_{\text{tr}}$ a suitable set of candidate information granules. With graph classification problems, suitable information granules can be intended as pivotal, meaningful and/or recurrent subgraphs drawn from the training data.

Due to the relatively small size of the training graphs, GRALG populates the set of candidate information granules $\mathcal{B}$ thanks to an exhaustive extraction of the subgraphs belonging to the training graphs, up to a maximum order of five nodes per subgraph.

4.2. Granulator and Alphabet Synthesis

The set of candidate information granules $\mathcal{B}$ is forwarded to the Granulator block, which is in charge of extracting a suitable subset $\mathcal{A} \subset \mathcal{B}$ of meaningful granules of information to be collected in an alphabet of symbols $\mathcal{A}$. In order to accomplish this task, the Granulator runs a clustering procedure [16] and the representative elements of well-formed clusters are considered as meaningful granules of information.

Specifically, the Granulator receives $\mathcal{B}$ from the Extractor block and runs a Basic Sequential Algorithmic Scheme (BSAS) free-clustering procedure on it. BSAS is driven by the following inputs:

- a dissimilarity measure between graphs;
- a value θ which determines the maximum radius of the resulting cluster;
- a value Q which determines the maximum number of clusters that can be spawned;
- a suitable way to determine the representative element of a cluster: since BSAS will work in the graph domain, the choice fell on using the medoid of each cluster as its representative element [39].

In order to analyze the data at different levels of granularity (i.e., resolution), BSAS runs several times for different values of θ. Without loss of generality, the considered values of θ are collected in a vector $\boldsymbol{\theta}$. Regardless of the value of θ, each cluster in each partition is then evaluated by means of a cluster quality index that jointly takes into account its cardinality (i.e., number of patterns) and compactness (i.e., average pattern-to-center distance) and only the centers of well-formed (i.e., compact and populated) clusters are admitted to be part of the alphabet. Mathematically speaking, the cluster quality index is defined as:

$$F(\mathcal{C}) = \eta \cdot \Phi(\mathcal{C}) + (1-\eta) \cdot \Psi(\mathcal{C}) \tag{7}$$

namely, as a linear convex combination between the compactness of the cluster $\Phi(\mathcal{C})$ and its cardinality $\Psi(\mathcal{C})$ weighted by a trade-off parameter $\eta \in [0,1]$. The compactness of the cluster is defined as the average pattern-to-center distance:

$$\Phi(\mathcal{C}) = \frac{1}{|\mathcal{C}|-1} \sum_i d(g^\star, g_i) \tag{8}$$

where $g_i \in \mathcal{C}$ is the i^{th} pattern of cluster $\mathcal{C}$, and $g^\star$ is the representative element of the cluster. Instead, the cardinality of the cluster is defined as the relative size of the cluster $\mathcal{C}$ with respect to the overall number of candidate information granules:

$$\Psi(\mathcal{C}) = 1 - \frac{|\mathcal{C}|}{|\mathcal{B}|} \tag{9}$$

Since both $\Phi(\mathcal{C})$ and $\Psi(\mathcal{C})$ are negative-oriented, a cluster is considered for being part of the alphabet $\mathcal{A}$ if and only if its quality index $F(\mathcal{C})$ is below a threshold $\tau_F \in [0,1]$. The medoids whose clusters satisfy such condition compose the alphabet of symbols $\mathcal{A}$.

4.3. Embedder

The set of granules of information collected in the alphabet $\mathcal{A}$ is exploited by the Embedder block in order to build an embedding space in which classification can be performed by means of any statistical pattern recognition system available in the current literature.

To do so, the Embedder exploits the so-called symbolic histograms [36], defined as an integer-valued vector containing the number of occurrences of each symbol in $\mathcal{A}$ within the graph $\mathcal{G}$ to be embedded, i.e.,:

$$\mathbf{h}(\mathcal{A}, \mathcal{G}) = [\text{occ}(s_1, \mathcal{G}), \ldots, \text{occ}(s_n, \mathcal{G})] \tag{10}$$

where $\mathcal{A} = \{s_1, \ldots, s_n\}$ is the alphabet of symbols and where $\text{occ} : \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{N}_0^+$ is the enumeration function that counts the number of times each symbol $s \in \mathcal{A}$ appears in $\mathcal{G}$. The counting procedure, in turn, exploits the same dissimilarity measure $d(\cdot, \cdot)$ already used in the Granulator block in order to seek for an (inexact) match between the symbols under analysis s_i and the constituent parts of the graph $\mathcal{G}$ to be embedded. If the dissimilarity measure is below a given value ζ_i, a match is considered as a 'hit' and a new occurrence is found. In order to scale the matching threshold in a symbol-aware fashion, we set:

$$\zeta_i = 1.1 \cdot \Phi(\mathcal{C}_i) \tag{11}$$

where $\mathcal{C}_i$ is the cluster whose representative element is s_i, and $\Phi(\cdot)$ is the compactness of the cluster (see Equation (8)).

The symbolic histogram in Equation (10) is evaluated for all graphs in $\mathcal{S}_{\text{tr}}$ yielding the instance matrix $\mathbf{H}_{\text{tr}}$, suitable to train the classifier.

4.4. Classifier

The Classifier receives $\mathbf{H}_{\text{tr}}$ from the Embedder block and trains a classification system on $\mathbf{H}_{\text{tr}}$. Eventually, the classifier also receives the embedded version of $\mathcal{S}_{\text{vl}}$ (i.e., $\mathbf{H}_{\text{vl}}$) to address the performance of the classifier. In this work, the classifier is set as a simple K-Nearest Neighbours (K-NN) decision rule [40] with $K = 5$.

4.5. Model Training and Testing

The model training starts with the Extractor block, which returns the set of candidate information granules $\mathcal{B}$.

In order to optimize the parameters of the overall system to maximize the performance of the classifier, a genetic algorithm [41] is used. Each individual from the evolving population is identified by a genetic code of the form

$$\begin{bmatrix} Q & \eta & \tau_F & w_{\text{sub}}^{\text{node}} & w_{\text{ins}}^{\text{node}} & w_{\text{del}}^{\text{node}} & w_{\text{sub}}^{\text{edge}} & w_{\text{ins}}^{\text{edge}} & w_{\text{del}}^{\text{edge}} \end{bmatrix} \tag{12}$$

where:

- $Q \in [1, 500]$ is the maximum number of allowed clusters for the BSAS clustering algorithm;
- $\eta \in [0, 1]$ is the compactness-vs.-cardinality weight in Equation (7);
- $\tau_F \in [0, 1]$ is the threshold for discarding unpromising clusters when it comes to build the alphabet $\mathcal{A}$;
- $w_{\text{sub}}^{\text{node}}, w_{\text{ins}}^{\text{node}}, w_{\text{del}}^{\text{node}}, w_{\text{sub}}^{\text{edge}}, w_{\text{ins}}^{\text{edge}}, w_{\text{del}}^{\text{edge}} \in [0, 1]^6$ are the six GED weights for insertion, deletion, substitutions of nodes and edges.

Each individual from the evolving population:

1. runs the Granulator;
2. runs the Embedder;
3. runs the Classifier;

and the performance of the Classifier in correctly predicting the labels in $\mathbf{H}_{\text{vl}}$, joined with the alphabet cardinality, marks the fitness function of the individual.

At the end of the genetic optimization, one obtains the optimal alphabet $\mathcal{A}^\star$ and the two embedded matrices $\mathbf{H}_{\text{vl}}^\star$ and $\mathbf{H}_{\text{tr}}^\star$ against $\mathcal{A}^\star$. In order to further reduce the size of the alphabet, a second lightweight genetic algorithm can be employed for feature selection. In this case, the genetic code reads as a binary mask:

$$\mathbf{m} = \{0, 1\}^{|\mathcal{A}^\star|} \tag{13}$$

and each individual from the evolving population:

1. receives $\mathbf{H}_{\text{vl}}^\star$ and $\mathbf{H}_{\text{tr}}^\star$;

2. projects $\mathbf{H}^{\star}_{\mathrm{vl}}$ and $\mathbf{H}^{\star}_{\mathrm{tr}}$ on the subspace spanned by 1's in its genetic code (i.e., binary mask), yielding $\mathbf{H}^{\star\prime}_{\mathrm{vl}}$ and $\mathbf{H}^{\star\prime}_{\mathrm{tr}}$;
3. trains the classifier on $\mathbf{H}^{\star\prime}_{\mathrm{tr}}$, evaluates its performance on $\mathbf{H}^{\star\prime}_{\mathrm{vl}}$, and, along with the alphabet cardinality, yields the fitness function for the individual.

At the end of the second genetic optimization, we retain the reduced alphabet $\mathcal{A}^{\star\prime} \subset \mathcal{A}^{\star}$ and the training set instance matrix $\mathbf{H}^{\star\prime}_{\mathrm{tr}}$ embedded against $\mathcal{A}^{\star\prime}$. The classifier, trained on $\mathbf{H}^{\star\prime}_{\mathrm{tr}}$, is finally tested on $\mathbf{H}^{\star\prime}_{\mathrm{ts}}$ (i.e., instance matrix obtained by embedded $\mathcal{S}_{\mathrm{ts}}$ against $\mathcal{A}^{\star\prime}$), yielding the final performance on the test set of the synthesized classification model.

5. Genetically Optimized Graph Convolutional Neural Network

In this section, we introduce the Graph Neural Network (GNN) architecture adopted to solve the two supervised problems, and perform a comparison with the Granular Computing-based approach introduced in the previous section.

The GNN architecture starts from the same preprocessing strategy described before: that is, it takes sequences of frames, performs the split, processes the raw training set using the BSAS clustering algorithm and converts sequences to graphs. The next step is the training phase which is performed using an evolutionary optimization approach (see Figure 3). Finally, the testing phase is performed.

We have employed the same pre-processing, training, testing and optimization stages with respect to GRALG in order to ensure a fair comparison between the two systems. Computational results will allow us to perform a first evaluation of knowledge extracted from graphs in this kind of scenario where multiple labels are involved for each network graph instance.

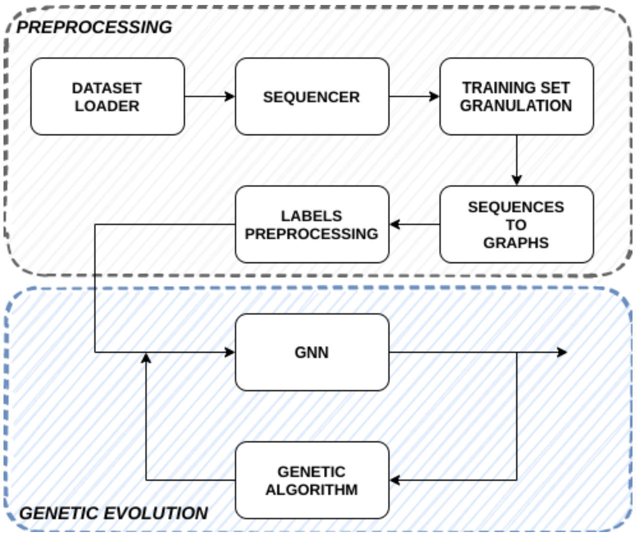

Figure 3. GNN optimization architecture.

5.1. Graph Convolutional Neural Network

The proposed GNN architecture is based on the Graph Convolutional Network (GCN) layers introduced in [42]. In this case, as already pointed out in Section 3.6, we are formally stating an inductive supervised graph classification problem, where a specific label or a set of labels that we aim to predict is associated with each graph.

We adopt a basic GNN built on $n^{(l)}$ GCN layers and multiple dense layers. GCN layers work using the widely employed message passing approach, which iteratively updates nodes' representations from layer to layer following Equation (14):

$$h_i^{l+1} = f(h_i^l, \{h_j^l\}_{j \in \mathcal{N}_i}) \tag{14}$$

where h_i^{l+1} is the feature vector of the i-th node for the $(l+1)$-th GCN layer, $j \in \mathcal{N}_i$ is the index referred to neighborhood nodes and $f(\cdot)$ is a function that specifies the behavior of the whole GNN. This equation works specifically for the i-th node, thus reducing overall processing complexity since it is independent of the graph size. This means that each GCN layer performs message passing through the neighborhood of each node in order to build a representation of the local graph structure related to each node. In this way, thanks to multiple GCN layers, a GNN is able to extend the local graph structure related to a specific node by taking into account multiple hops neighborhoods (see Figure 4).

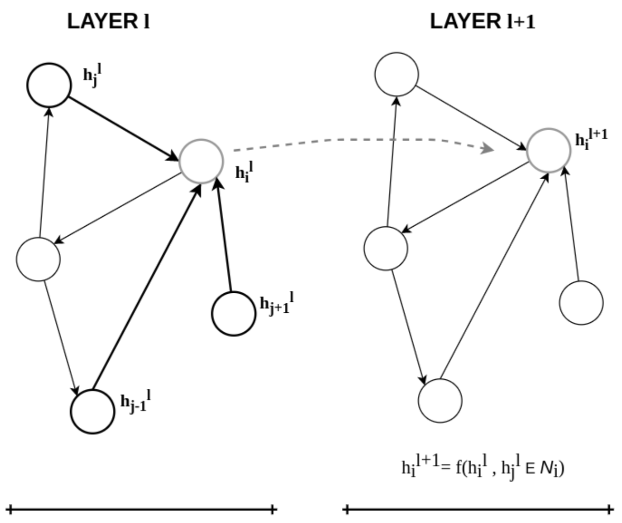

Figure 4. GNN basic message passing architecture.

5.2. Training

The proposed scheme to train the GNN is based on an evolutionary optimization algorithm that searches the optimal hyperparameters with a wrapper mode approach. The genetic code is shown in Equation (15):

$$\begin{bmatrix} \delta & \alpha & p & n^{(l)} \end{bmatrix} \tag{15}$$

where:

- $\delta \in [0, 0.8]$ is the dropout rate for the neural network;
- $\alpha \in [0.001, 0.05]$ is the parameter to setup the ADAM optimizer learning rate;
- $p \in [2, 20]$ is the patience delta rate;
- $n^{(l)} \in \{1, \ldots, 3\}$ is the number of GCN layers in the GNN topology.

The evaluation of the fitness function takes the granulated training set and trains the GNN whose hyperparameters are given by the particular genetic code instance. During this phase, we use $n_e = 100$ as the maximum number of training epochs. Next, the fitness function reads as the accuracy value obtained during the evaluation of the validation set by the GNN.

6. Results

In this section, we outline the experimental results obtained over the AWID dataset introduced in Section 3. Specifically, we show two sets of experiments, namely the binary classification problem and the non-exclusive one (see Section 3.6).

By considering the parameters involved in the whole processing stage, we identified six different simulation setups for $B \in \{10, 20, 50\}$ and $|\mathcal{S}_{\text{tr}}^{(g)}| \in \{1000, 10{,}000\}$.

As regards the classification capabilities of the two models, final results are expressed in terms of mean accuracy and Jaccard similarity score (binary and non-exclusive classification problem, respectively). Conversely, as regards the model complexity, alphabet cardinality and the number of GCN layers serve as performance indices for GRALG and GNN, respectively.

Table 4 shows that, despite the difficulty to deal with such a complex problem, results are comparable with those obtained in [3]. This is particularly true if the granulated training set is big enough to bring a fair amount of information from the entire training graphs. Indeed, both solutions seem sensitive to the number of graphs in the training set. The GRALG classification system outperforms the GNN when the buffer length B used to build the graphs is bigger than or equal to 20 frames.

Table 4. Mean accuracy and complexity obtained for the binary classification problem as a function of both the buffer length B and the maximum compressed training set size $Q_{\max} = |\mathcal{S}_{\text{tr}}^{(g)}|$.

-	**GRALG**		**GNN**	
Buffer Setup	$\boldsymbol{Q_{\max} = 1000}$	$\boldsymbol{Q_{\max} = 10{,}000}$	$\boldsymbol{Q_{\max} = 1000}$	$\boldsymbol{Q_{\max} = 10{,}000}$
$B = 10$	0.8910 (187)	0.9101 (151)	0.8939 (3)	0.8291 (2)
$B = 20$	0.8773 (129)	0.9012 (164)	0.4562 (2)	0.8203 (3)
$B = 50$	0.8625 (146)	0.9397 (138)	0.4414 (2)	0.7798 (2)

The model complexity obtained for the binary supervised problem is shown in Table 4. Since the number of dense layers is fixed, the GNN complexity reads as the number of GCN layers chosen by the genetic algorithm in the interval $\{1, \ldots, 3\}$. Our results show that the optimization procedure converges to 2 or 3 layers.

More precisely, we can obtain an estimate of the asymptotic complexity of the GNN by taking into account Graph Convolution Layer and Dense Layer sizes:

$$\text{complexity}_{GNN} = \mathcal{O}\left(n^{(l)} \cdot |\mathcal{E}| \cdot C \cdot F + n^{(l)} \cdot |\mathcal{V}|^2 + n^{(d1)} \cdot n^{(d2)} \cdot n^{(d3)}\right) \tag{16}$$

where $n^{(l)}$ is the number of GCN layers of the network, C is the numbers of input channels, F is the number of features maps, $n^{(d1)}$, $n^{(d2)}$ and $n^{(d3)}$ are the sizes of the first, second and last dense layers, respectively (in our case, 32, 16 and 1, respectively).

As GRALG is concerned, instead, the alphabet cardinality marks the dimensionality of the embedding space. If compared with our previous analysis (i.e., performed by buffering a sequence of packets with no transformation towards the graph domain [3]), we can see that the resulting alphabet is larger, yielding embedding space(s) with more than 120 features (i.e., information granules), after the feature selection step. This represents a physiological increase in terms of model complexity due to the deeper topological information brought by graphs. Formally, we can formulate an upper-bound complexity estimate for GRALG with the following expression:

$$\text{complexity}_{GRALG} = \mathcal{O}\left(|\mathcal{A}| \cdot \left(|\mathcal{V}| \cdot d^{\text{edge}} + |\mathcal{V}|^2 \cdot d^{\text{node}}\right)\right) \tag{17}$$

where $|\mathcal{A}|$ is the alphabet size, and $|\mathcal{V}| \cdot d^{\text{edge}}$ and $|\mathcal{V}|^2 \cdot d^{\text{node}}$ are contributions related to the two nBMF GED stages (i.e., edge matching and node matching, respectively).

Equations (16) and (17) are a parametric estimate of the complexity of both systems, useful to assess the trend of computing time as the size of the involved model and/or data grows. They show that the computational complexity of the proposed classification approach is polynomial in all system size parameters.

Numerical results shown in this paper have been obtained by using the Python programming language. A real-time implementation of the inference algorithm provided by the trained model requires a compiled high performance programming language such as C++, *Rust* or *Go* and a dedicated hardware platform, correctly provisioned with respect to the throughput of the network under attack.

Figure 5 illustrates a scheme of the processing flow of the proposed anomaly detection algorithms when applied in test mode, i.e., when looking at network traffic and using the trained classification algorithm to produce an inference. We believe that an implementation of the proposed model is affordable for most network device platforms (e.g., on board a WiFi router or a high-end WiFi Access Point). A lightweight design could also be conceived, trading-off the accuracy of the model predictions with its complexity, e.g., by reducing the cardinality of the symbol alphabet with genetic optimization, where the weight of complexity versus accuracy is used as a knob to manage the trade-off.

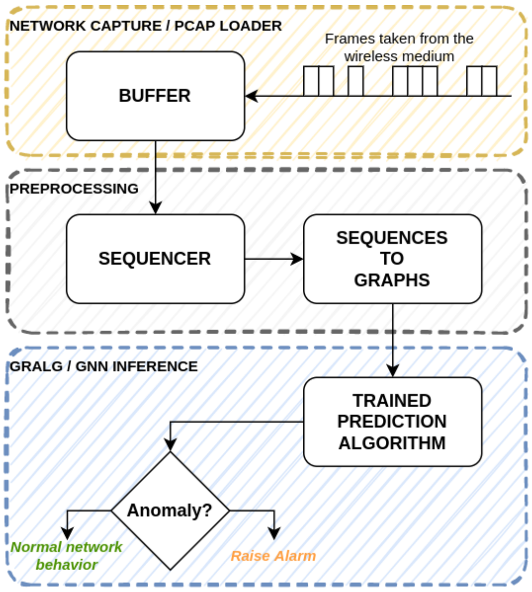

Figure 5. Processing flow of the proposed classification algorithms in test mode.

We can also observe that, in this first batch of simulations, the cardinality of the alphabet does not seem to be correlated to the buffer length. This is probably due to the fact that, given the small size of the network, the vast majority of the frames on the medium are from legit terminals. The graphs' sizes seems to increase when the network is subject to *flooding* attacks, which try to disrupt correct communications behavior by sending multiple management or control frames (*rts*, *cts* or *power_saving*).

Table 5 shows the results on the non-exclusive supervised problem, where both solutions achieve a Jaccard similarity score of $\sim$70%, with the best ones achieved for buffer length $B \geq 20$. A similar phenomenon has been observed in [3], where a larger buffer length yields a larger amount of *normal* traffic, which constitutes the majority of instances in the dataset.

Table 5. Jaccard similarity score and complexity obtained for the non-exclusive supervised problem as a function of both the buffer length B and the maximum compressed training set size $Q_{\max} = |\mathcal{S}_{\text{tr}}^{(g)}|$.

-	GRALG		GNN	
Buffer Setup	$Q_{\max} = 1000$	$Q_{\max} = 10{,}000$	$Q_{\max} = 1000$	$Q_{\max} = 10{,}000$
$B = 10$	0.6722 (150)	0.5609 (133)	0.5348 (2)	0.7108 (2)
$B = 20$	0.7318 (173)	0.6657 (159)	0.6971 (3)	0.7243 (2)
$B = 50$	0.6598 (147)	0.6979 (165)	0.5914 (2)	0.6233 (3)

The compressed training set cardinality plays a key role also in this case, with the GRALG classification system that is able to achieve interesting results with fewer examples, conversely to the GNN which requires a larger set of training graphs to learn from. Moreover, mean complexity value tends to be lower in this case than the binary supervised problem, with alphabet cardinality of $\sim$150 symbols and a number of GCN layers often equal to 2. This is mainly due to the greater complexity of the non-exclusive supervised problem, thus requiring further study and developments in classifying multi-labels. Moreover, decision surfaces of the non-exclusive problem are less defined and precise than those taken from the binary classification problem, resulting in slightly better performances in the latter case. As regards the model complexities, GRALG's alphabet cardinality is high during the first optimization stage (in the order of hundreds of symbols), and correctly filtered in the second stage. Results also show that the GNN needs to process graphs mainly considering the second order neighborhood in order to maximize the performances: we think that this aspect is linked to the size of the network graphs themselves provided by the AWID dataset.

As anticipated before, if we consider the different nature of the supervised problems analyzed in this work, it is not possible to perform a direct and fair comparison with other works (a summary of which can still be found in Table 6). This is due to multiple reasons: first, the supervised problem itself is defined in a completely different domain, in which aggregated data are represented with a graph structure; second, preprocessing patterns in the graph domain require the definition of a non-exclusive supervised problem, where a set of labels should be predicted for each graph instance; lastly, the purpose of the proposed Granular Computing approach consists of both building a human-interpretable white/grey box model and going towards state-of-the-art performances. Anyhow, we can make some considerations regarding currently used approaches and the impact of results. Thus, as a further performance assessment, in the following, we compare the results provided by our approach against those proposed in [11,14,43,44].

In [10], Kolias et al. proposed the AWID dataset as a first benchmarking tool for WiFi network intrusion detection analysis. They performed a first analysis of supervised algorithms behavior on this kind of data. Results showed that *impersonation* and *flooding* attacks are the most difficult attack classes to recognize, along with the *normal* class itself. In [1], this result has been confirmed, while at the same time improving final accuracy and providing a first approach to automatic knowledge discovery, by means of a genetic optimization-based feature selection.

These results has been further confirmed in [11], where, despite the resulting accuracy of $\sim$78%, the proposed algorithm has been able to build human readable *IF-THEN* rules. In this work, instead of explicitly outputting human-readable rules, we aim to return the information granules of the optimal alphabet $\mathcal{A}$ resulting from the optimization process. This approach allows us to give insight into the network status, thus isolating key pieces of attack patterns and improving model and classes interpretability. A key aspect that we think is fundamental in this kind of scenario concerns the possibility to further exploit the knowledge discovery capabilities, applying them to analyze peculiar complex classes, e.g., the *normal* one.

Multiple works included different approaches to implement data preprocessing and classification pipeline.

For instance, Qin et al. [43] proposed a two-dimensional data cleaning preprocessing strategy able to introduce a new supervised problem which is then tackled with Support Vector Machines. Despite some difficulties in the correct recognition of *impersonation* and *flooding* attacks, it has shown interesting accuracy for the *normal* traffic class. Both the GRALG classification system and the GNN are able to outperform Qin's work in *impersonation* recognition, for both the binary supervised problem and the non-exclusive one, with different values of the buffer length B and the maximum number of clusters $Q_{\max}$.

A further analysis has been carried out in [14], where, after a starting statistical analysis, Hidden Markov Models have been coupled with Kernel Density Estimation approaches to target the prediction of a subset of classes. Results obtained shows interesting precision and recall values for multiple classes (i.e., *arp*, *deauthentication*), despite the false alarm rate that tends to be high for *impersonation* attacks.

Both [14,43] did not cope with white box models, thus reducing knowledge discovery capabilities in their respective approaches. Although the use of queues in [14] allowed the authors to perform processing on a higher semantic level, the knowledge discovery capabilities are limited due to the lack of an adaptive process. For this reason, in this work, we employed suitable evolutionary optimization procedures to find the best alphabet for each problem, yielding key subgraphs that characterize target anomalies.

In [44], the authors employed a similar statistical preprocessing strategy to perform feature selection and fed training instances to an ensemble of classifiers, similarly as in [1]. The whole classifier is made as an ensemble of techniques including Bagging [45], Random Forests [46] and Gradient Boosting [47], from which the best is taken specifically for each class. Reported results include high level of accuracy and F1-score, and yet confusion matrices still show that *impersonation* and *flooding* attack introduce errors both in terms of false positives and false negatives. Despite processing single instances having the possibility to be effective with specific classes, it does not allow for discovering recurrent features on higher semantic levels, or taking into account the whole structure of the input instances' space. This implies a loss of information when dealing with classes composed by structured exchange of data, thus limiting classification performances for complex anomalies and attacks such as *cafe_latte* or *evil_twin*. In fact, in this case, white box modeling capabilities are realized by means of random forests, which allow for exploiting key features during the training phase.

In order to improve knowledge discovery capabilities, in this work, we propose to take into account the possibility to model data transmission between network nodes. This is achieved by firstly performing a preprocessing strategy which aims to represent instances in the graph domain and then adopting a Granular Computing-based approach. In this way, the output of the training algorithm is able to return insights about the status of the network by means of recurrent and meaningful subgraphs, a feature completely lacking in all approaches that follow a single-packet analysis.

Table 6. Performance comparison with related state-of-the-art works.

Reference	Dataset Size	Classification Problem	Method	Feature Selection Strategy	Performance Index	Performance (%)
[10]	Entire AWID dataset	Both using macro-classes and single attack labels	Random Tree J48	Manual	Accuracy	96.1982
[11]	Entire AWID dataset	Both using macro-classes and single attack labels	Particle Swarm Optimization-based	Both manual and automatic	Accuracy	77.8
[48]	AWID subset (∼2.4 millions patterns)	4 macro-classes	Deep Learning Stacked Autoencoders	Automatic	Accuracy	98.6688
[43]	AWID subset (∼1.3 millions patterns)	4 macro-classes	SVM	Both manual and automatic	Accuracy	89–99 (depending on class)
[14]	Modified subset of the AWID dataset	7 attack classes	HMM and KDE	Automatic	Precision	92.28
[44]	AWID subset (∼2.3 millions patterns)	4 macro-classes	Bagging, Random Forests ExtraTrees and XGBoost	Both manual and automatic	Accuracy	99.1
[1]	AWID subset (∼84 millions patterns)	14 attack classes	BSAS and *K*-NN driven by GA	Both manual and automatic	Accuracy	92.446 (false alarm rate 12.1)
[1]	AWID subset (∼84 millions patterns)	14 attack classes	BSAS and Min-Max Neural Network	Manual	Accuracy	89–90 (false alarm rate 11.6)
[3]	AWID subset (∼565,000 patterns)	6 attack classes	Granular Computing-based approach driven by GA	Both manual and automatic	Reliability	89.3 ($L_{sub} = [6,7]$, $B = 10$)
This work	AWID subset (∼565,000 patterns)	14 attack classes	Granular Computing-based approach and GNN driven by GA	Both manual and automatic	Accuracy	93.9 ($B = 50$)
This work	AWID subset (∼565,000 patterns)	14 attack classes	Granular Computing-based approach and GNN driven by GA	Both manual and automatic	Jaccard Similarity Score	73.2 ($B = 20$)

7. Conclusions

In this paper, graph-based classification algorithms have been developed and applied to network traffic anomaly identification and analysis. The proposed Computational Intelligence approaches showed good performance capabilities of both systems (GRALG and GNN) when dealing with non-exclusive labeled graphs.

Specifically, we described a preprocessing procedure able to define the original traffic analysis (supervised) problem in the graph domain, adopting non-exclusive labelling strategies. Later, we introduced two machine learning models based on Granular Computing and Graph Neural Networks, respectively, both automatically optimized with evolutionary meta-heuristics procedures. Results show interesting interpretability capabilities of Granular Computing-based models, thus obtaining output alphabets able to consistently describe the network traffic behavior of the target classes. Despite differences in the supervised problem definition, input domain definition, processing system modeling approach and dataset usage, results have been compared with state-of-the-art works both in terms of specific performance metrics and model interpretability. Moreover, performance comparison between the two algorithms showed interesting Jaccard similarity scores and, at the same time, different levels of human-readability. GRALG has been able to isolate subgraphs relevant to the automatic traffic analysis problem at hand, while the GNN requires further processing techniques [49].

On the application domain side, we showed that anomalous/attack traffic in WiFi networks can be identified with high reliability even in the case of complex traffic patterns. We considered a classification approach based on *sequences* of captured packets, trying to extract the relevant traffic structure. Thus, we went beyond the single packet-by-packet inspection and classification that is commonly applied. The proposed algorithms can be viewed as a contribution to building traffic analysis tools that dig into the traffic flow structure to understand the high-level "meaning" of network traffic, without inspecting payloads. This is of the utmost importance, given that an increasingly large fraction of data is being carried through encrypted tunnels (e.g., secure Virtual Private Networks based on IPsec or Transport Layer Security connections, e.g., in HyperText Transfer Protocol over TLS). Moreover, the computational complexity of the proposed algorithms has been evaluated. This analysis hints at the feasibility of a real-time implementation, highlighting that the critical factor lies with the cardinality of the symbol alphabet. Pushing for a lightweight implementation of the classifier calls for a trade-off on accuracy. An in-depth investigation is required to design a real-time version of the proposed algorithms, which is out of the scope of the present work and can be part of future work.

Lastly, further research activities are going on in order to build a direct association between information granules and specific labels, and to cope with unsupervised problems for traffic analysis and anomaly detection.

Author Contributions: Conceptualization, G.G.; methodology, G.G., A.R. and A.M.; software, G.G. and A.M.; validation, G.G., A.M. and A.B.; formal analysis, G.G. and A.M.; investigation, G.G. and A.M.; resources, A.R.; data curation, G.G.; writing—original draft preparation, G.G., A.M. and A.B.; writing—review and editing, G.G., A.M., A.R. and A.B.; visualization, G.G.; supervision, A.R. and A.B.; project administration, A.R. and A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was founded by the University of Rome "La Sapienza" project "Distributed Evolutionary Swarm Intelligence and Granular Computing Techniques for Nested Complex Systems Modelling" under grant agreement RM120172B519E6C4.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The AWID dataset can be requested at https://icsdweb.aegean.gr/awid accessed on 20 August 2022.

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

Abbreviations

The following abbreviations are used in this manuscript:

AP	Access Point
AWID	Aegean WiFi Intrusion Detection (dataset)
BSAS	Basic Sequential Algorithmic Scheme
DTW	Dynamic Time Warping
FIFO	First-In-First-Out
GCN	Graph Convolutional Networks
GED	Graph Edit Distance
GNN	Graph Neural Network
GRALG	Granular Computing Approach for Labelled Graphs
K-NN	K-Nearest Neighbours
nBMF	node Best Match First
SDN	Software Defined Networking
SOHO	Small Office Home Office
WEP	Wired Equivalent Privacy (protocol)

References

1. Rizzi, A.; Granato, G.; Baiocchi, A. Frame-by-frame Wi-Fi attack detection algorithm with scalable and modular machine-learning design. *Appl. Soft Comput.* **2020**, *91*, 106188. [CrossRef]
2. Granato, G.; Martino, A.; Baldini, L.; Rizzi, A. Intrusion Detection in Wi-Fi Networks by Modular and Optimized Ensemble of Classifiers. In Proceedings of the 12th International Joint Conference on Computational Intelligence-NCTA, INSTICC, SciTePress, Budapest, Hungary, 2–4 November 2020; pp. 412–422. [CrossRef]
3. Granato, G.; Martino, A.; Rizzi, A. A Granular Computing Approach for Multi-Labelled Sequences Classification in IEEE 802.11 Networks. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–9. [CrossRef]
4. Kakkavas, G.; Stamou, A.; Karyotis, V.; Papavassiliou, S. Network Tomography for Efficient Monitoring in SDN-Enabled 5G Networks and Beyond: Challenges and Opportunities. *IEEE Commun. Mag.* **2021**, *59*, 70–76. [CrossRef]
5. Kafetzis, D.; Vassilaras, S.; Vardoulias, G.; Koutsopoulos, I. Software-Defined Networking Meets Software-Defined Radio in Mobile ad hoc Networks: State of the Art and Future Directions. *IEEE Access* **2022**, *10*, 9989–10014. [CrossRef]
6. Segura, G.A.N.; Chorti, A.; Margi, C.B. Centralized and Distributed Intrusion Detection for Resource-Constrained Wireless SDN Networks. *IEEE Internet Things J.* **2022**, *9*, 7746–7758. [CrossRef]
7. Khan, J.A.; Chowdhury, M.M. Security Analysis of 5G Network. In Proceedings of the 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 14–15 May 2021. [CrossRef]
8. Dietz, K.; Gray, N.; Seufert, M.; Hossfeld, T. ML-based Performance Prediction of SDN using Simulated Data from Real and Synthetic Networks. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–7. [CrossRef]
9. Saha, T.; Aaraj, N.; Jha, N.K. Machine Learning Assisted Security Analysis of 5G-Network-Connected Systems. *IEEE Trans. Emerg. Top. Comput.* **2022**. [CrossRef]
10. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 184–208. [CrossRef]
11. Kolias, C.; Kolias, V.; Kambourakis, G. TermID: A Distributed Swarm Intelligence-based Approach for Wireless Intrusion Detection. *Int. J. Inf. Secur.* **2017**, *16*, 401–416. [CrossRef]
12. Kulin, M.; Kazaz, T.; De Poorter, E.; Moerman, I. A Survey on Machine Learning-Based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer. *Electronics* **2021**, *10*, 318. [CrossRef]
13. Anton, S.D.D.; Fraunholz, D.; Schotten, H.D. Using Temporal and Topological Features for Intrusion Detection in Operational Networks. In Proceedings of the 14th International Conference on Availability, Reliability and Security ARES '19, Canterbury, UK, 26–29 August 2019; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
14. Sethuraman, S.C.; Dhamodaran, S.; Vijayakumar, V. Intrusion detection system for detecting wireless attacks in IEEE 802.11 networks. *IET Netw.* **2019**, *8*, 219–232. [CrossRef]
15. Bunke, H. Graph matching: Theoretical foundations, algorithms, and applications. In Proceedings of the 13th Vision Interface, Montreal, QC, Canada, 14–17 May 2000; pp. 82–88.
16. Martino, A.; Baldini, L.; Rizzi, A. On Information Granulation via Data Clustering for Granular Computing-Based Pattern Recognition: A Graph Embedding Case Study. *Algorithms* **2022**, *15*, 148. [CrossRef]
17. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Networks* **2009**, *20*, 61–80. [CrossRef] [PubMed]

18. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*; IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks—Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: Piscataway, NJ, USA, 2016; pp. 1–3534. [CrossRef]
19. De Santis, E.; Martino, A.; Rizzi, A. On component-wise dissimilarity measures and metric properties in pattern recognition. *PeerJ Comput. Sci.* **2022**, *8*, e1106. [CrossRef]
20. Granato, G.; Martino, A.; Baldini, L.; Rizzi, A. Intrusion Detection in Wi-Fi Networks by Modular and Optimized Ensemble of Classifiers: An Extended Analysis. *SN Comput. Sci.* **2022**, *3*, 310. [CrossRef]
21. Sakoe, H.; Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **1978**, *26*, 43–49. [CrossRef]
22. Bunke, H.; Allermann, G. Inexact graph matching for structural pattern recognition. *Pattern Recognit. Lett.* **1983**, *1*, 245–253. [CrossRef]
23. Bunke, H. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognit. Lett.* **1997**, *18*, 689–694. [CrossRef]
24. Levenshtein, V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **1966**, *10*, 707–710.
25. Martino, A.; Rizzi, A. An Enhanced Filtering-Based Information Granulation Procedure for Graph Embedding and Classification. *IEEE Access* **2021**, *9*, 15426–15440. [CrossRef]
26. Sechidis, K.; Tsoumakas, G.; Vlahavas, I. On the Stratification of Multi-label Data. In *Machine Learning and Knowledge Discovery in Databases*; Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–158.
27. Theodoridis, S.; Koutroumbas, K. *Pattern Recognition*, 4th ed.; Academic Press: Cambridge, MA, USA, 2008.
28. Martino, A.; Rizzi, A.; Frattale Mascioli, F.M. Distance Matrix Pre-Caching and Distributed Computation of Internal Validation Indices in k-medoids Clustering. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8. [CrossRef]
29. Jaccard, P. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bull. Société Vaudoise Sci. Nat.* **1901**, *37*, 241–272. [CrossRef]
30. Pedrycz, W. Granular computing: An introduction. In Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, BC, Canada, 25–28 July 2001; Volume 3, pp. 1349–1354. [CrossRef]
31. Yao, Y. Perspectives of granular computing. In Proceedings of the 2005 IEEE International Conference on Granular Computing, Beijing, China, 25–27 July 2005; Volume 1, pp. 85–90.
32. Bianchi, F.M.; Livi, L.; Rizzi, A.; Sadeghian, A. A Granular Computing approach to the design of optimized graph classification systems. *Soft Comput.* **2014**, *18*, 393–412. [CrossRef]
33. Baldini, L.; Martino, A.; Rizzi, A. Stochastic Information Granules Extraction for Graph Embedding and Classification. In Proceedings of the 11th International Joint Conference on Computational Intelligence-NCTA, (IJCCI 2019), Vienna, Austria, 17–19 September 2019; pp. 391–402. [CrossRef]
34. Baldini, L.; Martino, A.; Rizzi, A. Complexity vs. Performance in Granular Embedding Spaces for Graph Classification. In Proceedings of the 12th International Joint Conference on Computational Intelligence-NCTA, Budapest, Hungary, 2–4 November 2020; pp. 338–349. [CrossRef]
35. Baldini, L.; Martino, A.; Rizzi, A. Exploiting Cliques for Granular Computing-based Graph Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–9. [CrossRef]
36. Baldini, L.; Martino, A.; Rizzi, A. Relaxed Dissimilarity-based Symbolic Histogram Variants for Granular Graph Embedding. In Proceedings of the 13th International Joint Conference on Computational Intelligence-NCTA, Online, 25–27 October 2021; pp. 221–235. [CrossRef]
37. Baldini, L.; Martino, A.; Rizzi, A. Towards a Class-Aware Information Granulation for Graph Embedding and Classification. In Proceedings of the 11th International Joint Conference on Computational Intelligence, IJCCI 2019, Vienna, Austria, 17–19 September 2019; Revised Selected Papers; Merelo, J.J., Garibaldi, J., Linares-Barranco, A., Warwick, K., Madani, K., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 263–290. [CrossRef]
38. Baldini, L.; Martino, A.; Rizzi, A. A class-specific metric learning approach for graph embedding by information granulation. *Appl. Soft Comput.* **2022**, *115*, 108199. [CrossRef]
39. Martino, A.; Rizzi, A.; Frattale Mascioli, F.M. Efficient Approaches for Solving the Large-Scale k-Medoids Problem: Towards Structured Data. In Proceedings of the 9th International Joint Conference on Computational Intelligence, IJCCI 2017, Funchal-Madeira, Portugal, 1–3 November 2017; Revised Selected Papers; Sabourin, C., Merelo, J.J., Madani, K., Warwick, K., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 199–219. [CrossRef]
40. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
41. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed.; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.
42. Kipf, T.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:1609.02907.
43. Qin, Y.; Li, B.; Yang, M.; Yan, Z. Attack Detection for Wireless Enterprise Network: A Machine Learning Approach. In Proceedings of the 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Qingdao, China, 14–16 September 2018; pp. 1–6. [CrossRef]

44. Thomas, R.; Gupta, R. Design and Development of an Efficient Network Intrusion Detection System Using Machine Learning Techniques. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 1–35. [CrossRef]
45. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
46. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
47. Friedman, J. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
48. Thing, V.L.L. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–6.
49. Liu, N.; Feng, Q.; Hu, X. Interpretability in Graph Neural Networks. In *Graph Neural Networks: Foundations, Frontiers, and Applications*; Wu, L., Cui, P., Pei, J., Zhao, L., Eds.; Springer: Singapore, 2022; pp. 121–147.