universidad de león

DOCTORAL THESIS

BOTNET ACTIVITY SPOTTING WITH ARTIFICIAL INTELLIGENCE: EFFICIENT BOT MALWARE DETECTION AND SOCIAL BOT IDENTIFICATION

Submitted by

JAVIER VELASCO MATA

in fulfillment of the requirements for the Degree of

PHILOSOPHIÆDOCTOR (PH.D.)

DOCTORAL PROGRAM: PRODUCTION AND COMPUTER ENGINEERING

A dissertation supervised by

DR. VÍCTOR GONZÁLEZ-CASTRO,

PROF. DR. ENRIQUE ALEGRE GUTIÉRREZ

León, November 2023

universidad de león

TESIS DOCTORAL

DETECCIÓN DE ACTIVIDAD BOTNETS CON INTELIGENCIA ARTIFICIAL: DETECCIÓN EFICIENTE DE MALWARE BOT E IDENTIFICACIÓN DE BOTS SOCIALES

desarrollada por

JAVIER VELASCO MATA

a fin de optar al grado de

DOCTOR POR LA UNIVERSIDAD DE LEÓN

PROGRAMA DE DOCTORADO: INGENIERÍA DE PRODUCCIÓN Y COMPUTACIÓN

Tesis doctoral dirigida por

DR. VÍCTOR GONZÁLEZ-CASTRO,

Y EL PROF. DR. ENRIQUE ALEGRE GUTIÉRREZ

León, noviembre 2023

Abstract

In the cybercrime scope, botnets are networks of bots, automated entities that follow instructions from a cybercriminal. The capacity of these networks to operate en masse has made them one of the most popular tools to carry out malicious activities, from spam distribution to distributed denial-of-service (DDoS) attacks. This has made botnets one of the online threats with the most significant presence, causing billionaire losses to the global economy. The motivation of this PhD Thesis is to research and propose bot detection techniques. Specifically, it is focused on two types of bots: malware bots, i.e., virus programs that can be installed in the victims' devices without their notice; and social bots, i.e., fake accounts in Social Networks that try to masquerade as real humans to deceive regular users.

The first work is dedicated to the detection of network traffic produced by malware bots. In particular, we aim to improve the performance of botnet traffic classification using Machine Learning by selecting those features that further increase the detection rate. For this purpose, we employ two feature selection techniques, namely Information Gain and Gini Importance, which led to three candidate subsets of five, six and seven features. Then, we evaluate the three feature subsets with three models (Decision Tree, Random Forest and k-Nearest Neighbors). To test their performance, we generate two datasets based on the CTU-13 dataset, namely QB-CTU13 and EQB-CTU13. Finally, we measure the performance as the macro averaged F1 score over the computational time required to classify a sample. The results show that the highest performance is achieved by Decision Trees using a five-feature set, which obtained a mean F1 score of 0.850, classifying each sample in an average time of 0.78 microseconds.

Nowadays, there are networks with large bandwidths where vast amounts of traffic are generated every second, and it is hard to analyze all that information looking for threats, especially before large damage is done. Hence, the second work focuses on real-time detection of botnet traffic, even on high bandwidth networks. As a solution, we propose an approach capable of carrying out an ultra-fast network analysis (i.e. on time windows of one second), without a significant loss in the F1-score on botnet detection. We compared our model with other three literature proposals and it achieved the best performance: an F1 score of 0.926 with a processing time of 0.007 ms per sample. We also assessed the ro-

bustness of our model on saturated networks and on large bandwidths. In particular, our model is capable of working on networks with 10% of packet loss, and our results suggest that using commercial-grade cores of 2.4 GHz, our approach would only need four cores for bandwidths of 100 Mbps and 1 Gbps, and 19 cores on 10 Gbps networks.

The third and fourth works shift their focus towards social bots – fake accounts in Social Networks – which are a growing concern due to their promotion of fraudulent content and divisive ideologies. The damage caused by social media bots ranges from individual scams to affecting the whole society, as they may be used to contaminate the public debate with fake news, and thus can also influence the political sphere.

In the third work, we exploit the graph structure of Twitter to detect bots automatically. Specifically, we propose a novel pipeline approach, based on Kipf and Welling's Graph Convolutional Network model, which solves its limitations when used in graphs that are independent of the training data. We obtained an F1 score of 0.784 on the Cresci-rtbust dataset using a version of our proposal trained on seven completely independent datasets, a score 24% higher than the baseline. Furthermore, we present a novel seed-based cross-validation to generate class-balanced folds that minimize the intra-fold graph's edge loss. The new pipeline and cross-validation methods could be applied to any other problem that involves graph data.

We have realized that it is easy for a fake account to pose as a human with convincing metadata such as the user name, the account description, the location, and other public and editable information. It is also possible for bots to follow each other imitating the structure of real communities. Therefore, in the fourth work, we focus on building a Twitter bot detector based on the accounts' publication activity. For this purpose, we created a novel dataset of Twitter users that includes 17,945 manually labeled samples into bots or humans. Moreover, our dataset includes the users' public metadata, their who-follow-who relationships within the dataset while ensuring a dense connection between the users, and their most recent publication activity. To the best of our knowledge, our dataset is the largest in terms of completeness and manually labeled Twitter users into bots and humans. Our social bot detector proposal leverages BERTopic, a BERT-based topic predictor, to classify the tweets of the users into 102 categories. The resulting information is time-windowed at 15-minute intervals to characterize the users' activity and used to predict them into bots or humans using our proposed classifier, an ensemble of seven LSTM-based models. Our system reported an accuracy of 0.755 and an F1 of 0.777 on our new dataset.

Resumen

En el ámbito de la ciberdelincuencia, las botnets son redes de bots, autómatas que siguen las instrucciones de un ciberdelincuente. La capacidad de estas redes para operar en masa las han convertido en una de las herramientas más populares para llevar a cabo actividades maliciosas, desde la distribución de spam hasta ataques de denegación de servicio distribuido (DDoS, por su nombre en inglés). Esto ha hecho de las botnets una de las amenazas con mayor presencia en Internet, causando pérdidas multimillonarias a la economía mundial. La motivación de esta Tesis Doctoral es investigar y proponer técnicas de detección de bots. En concreto, esta Tesis se centra en dos tipos de bots: los bots malware, como virus informáticos que pueden instalarse en los dispositivos de las víctimas sin que éstas sean conscientes de ello; y los bots sociales, entendidos como cuentas falsas en redes sociales que intentan hacerse pasar por humanos reales para engañar a los usuarios *normales*.

El primer trabajo de investigación está dedicado a la detección del tráfico de red producido por bots. En particular, se pretende mejorar el rendimiento de la clasificación del tráfico de botnets mediante aprendizaje automático seleccionando aquellas características que mejoren la tasa de detección. Para ello, se emplearon dos técnicas de selección de características, la Ganancia de Información y la Importancia de Gini, cuyo uso condujo a tres subconjuntos candidatos de cinco, seis y siete características. A continuación, se evaluaron estos tres subconjuntos de características y tres modelos de clasificación (Árbol de Decisión, Bosque Aleatorio y k-Vecinos más Cercanos). Para comparar su rendimiento, se generaron dos conjuntos de datos basados en el conjunto de datos CTU-13, y que se llamaron QB-CTU13 y EQB-CTU13. Por último, se midió el rendimiento como la relación entre el macropromedio del valor F1 sobre el tiempo computacional medio necesario para clasificar una muestra. Los resultados muestran que el mayor rendimiento fue obtenido por un Árbol de Decisión utilizando el conjunto de cinco características, que consiguió un valor F1 medio de $0,850$ clasificando cada muestra en un tiempo medio de $0,78$ microsegundos.

Hoy en día existen redes de gran ancho de banda donde se generan grandes cantidades de tráfico por segundo, y es difícil analizar toda esa información en busca de amenazas, especialmente antes de que produzcan un gran daño. Por ello, el segundo trabajo

se enfoca en la detección en tiempo real del tráfico de botnets incluso en redes con un gran ancho de banda. Como solución, se propone un enfoque capaz de llevar a cabo un análisis ultrarrápido de la red (en ventanas temporales de un segundo), sin una pérdida significativa en el valor F1 en la detección de botnets. Se comparó el modelo con otras tres propuestas de la literatura, logrando el mejor rendimiento: un valor F1 de 0,926 con un tiempo de procesamiento de 0,007 ms por muestra. También se evaluó la robustez del modelo en redes saturadas y con grandes anchos de banda. En concreto, el modelo propuesto es capaz de funcionar en redes con una saturación del 10 % de pérdida de paquetes, y los resultados sugieren que, usando núcleos CPU comerciales de 2,4 GHz, el modelo solo necesitaría cuatro núcleos para anchos de banda de 100 Mbps y 1 Gbps, y 19 núcleos en redes de 10 Gbps.

Los trabajos tercero y cuarto cambian su enfoque hacia los bots sociales – cuentas falsas en las redes sociales –, que, dada promoción de contenidos fraudulentos e ideologías divisivas, son objeto de una creciente preocupación. El daño causado por los bots en las redes sociales va desde estafas a individuos, hasta afectar a toda la sociedad, ya que pueden ser utilizados para contaminar el debate público con noticias falsas, y por lo tanto también pueden influir en la esfera política.

En el tercer trabajo, para la detección automática de bots en Twitter se aprovecha la estructura de grafo de la red social. En particular, se propone un nuevo enfoque basado en un pipeline para utilzar el modelo de red convolucional de grafos de Kipf y Welling, resolviendo sus limitaciones cuando se utiliza en grafos no conectados con los datos de entrenamiento. Se obtuvo un valor F1 de 0,784 en el conjunto de datos Cresci-rtbust utilizando una versión de la propuesta entrenada en siete conjuntos de datos completamente independientes, lo que supuso una mejora del 24 % con respecto modelo de referencia. Además, se presenta un novedoso algoritmo basado en semillas para generar particiones del conjunto de datos y usarlos en validación cruzada, que minimiza la pérdida de enlaces entre nodos dentro de cada partición, a la vez que mantiene el balance en número de tipos de nodos. Los nuevos métodos de pipeline y validación cruzada pueden aplicarse a cualquier otro problema que implique datos estructurados en grafos.

La tercera línea de investigación dio indicios de la facilidad con la que una cuenta falsa puede hacerse pasar por humana con metadatos convincentes como el nombre de usuario, la descripción de la cuenta, la ubicación y otra información publica y rellenable. También es fácil para los bots seguirse unos a otros imitando el comportamiento humano, lo que dificulta su detección. Por ello, en el cuarto trabajo se enfoca en construir un detector de bots de Twitter basado en la actividad de publicación de las cuentas. Para este propósito, se creó un novedoso conjunto de datos de usuarios de Twitter que incluye 17,945 muestras etiquetadas manualmente como bots o humanos. Además, este nuevo conjunto de datos incluye los metadatos públicos de los usuarios, sus relaciones quién-sigue-a-quién dentro del conjunto de datos – garantizando una conexión densa entre los usuarios –, y la actividad de publicación más reciente de los usuarios muestreados. Hasta donde sabemos, el nuevo conjunto de datos es el mayor en términos de completitud y

de número de muestras. La propuesta de detector de bots sociales aprovecha BERTopic, un predictor de temas basado en BERT, para clasificar los tuits de los usuarios en 102 categorías. La información resultante se divide en ventanas de tiempo de 15 minutos para caracterizar la actividad de los usuarios y se utiliza para predecir si son bots o humanos mediante el clasificador propuesto: un ensamblado de siete redes neuronales basadas en LSTM. Este sistema obtuvo una precisión de 0,755 y un valor F1 de 0,777 % en el nuevo conjunto de datos.

Contents

Índice general

List of Figures

List of Tables

Acknowledgements

To my family, for supporting me.
To my friends, for being there.
To my research group, for their help.
To my directors Enrique and Víctor, for their guidance.
To Alan Turing, for making it possible.

Chapter 1

Introduction

1.1. Motivation

The original motive for this PhD Thesis started with an interest in using Artificial Intelligence to detect, and prevent, threats in the ambit of Distributed Denial of Service (DDoS) attacks, computer viruses propagation, and malicious activities over the Internet in general. This motivation later focused on botnets, i.e., a network of computers infected by malware specialized in remote-controlling the victims' devices under the control of an attacking party; as it was an interest of INCIBE (Instituto Nacional de Ciberseguridad, the Spanish National Cybersecurity Institute) and the work was partially funded by the Addendum 22, an agreement between the University of León and INCIBE, later renewed and renamed as Addendum 01. An initial work was completed during the Master Thesis, later improved, submitted and presented at an international conference [Velasco-Mata et al., 2019] . The first proposal, presented in Chapter 3, is a continuation of this work.

After our research on botnets as computer viruses, we noticed recent trends in distributed criminal activity on the Internet: Bots as fake accounts in social media, which were being used to promote spam, harassment towards normal users, or hate campaigns, among other examples. This served as motivation to continue the research on using Artificial Intelligence to detect online threat agents, but this time focusing on this type of bots.

The work done in this PhD thesis is divided into four main chapters: two related to the detection of botnets as networks of computer viruses, and two related to detecting "social bots", i.e., automated fake users in Social Media. In particular, the focus was on the Twitter platform, now called X. The purpose and motivation of these four research lines are more detailed as follows.

1.1.1. Efficient detection of botnet malware

This work focuses on the problem of detecting network traffic generated by a bot, i.e., a malicious program installed on the victim's device without their consent and usually without their knowledge [Padhiar et al., 2023; Garcia et al., 2014] . Proposing a solution is challenging due to several hardships.

First, botnet programs are typically designed to be stealthy, operating as quietly as

possible to evade detection [Koroniotis et al., 2019; Li et al., 2019; Alomari et al., 2023; Georgoulias et al., 2023] . The reason behind this is the regular cycle of botnets creation: cybercriminals begin by building a network of bots without attracting attention, and once it is large enough for their purposes [Lourenço and Marinos, 2020; Padhiar et al., 2023] , they direct the bots to initiate a coordinated and massive attack, such as a DDoS targeting a server [Li et al., 2023] . Detecting botnets in the initial phase is hard due to their inherently stealthy design.

Second, contemporary botnets use encryption protocols to cipher their communication [Anderson and McGrew, 2016; Chen et al., 2023] , hence detecting instructions in their messages is not possible unless a deciphering method is known. A generic approach should, therefore, concentrate on communication patterns rather than detecting patterns in plain data in order to identify them.

Third, and related to the second, it is hard to manually analyze a bot malware sample to create communication signatures, i.e., patterns to search for [Roesch et al., 1999; Albin and Rowe, 2012; Aydın et al., 2009; Zaheer et al., 2023] . This involves analyzing through the program's obfuscation if there is one [Liao et al., 2013; Rauti and Laato, 2023] , a method to decipher the messages if possible, and an examination of the instructions communicated between a bot and the rest of the botnet. It is preferable to have an automated approach to learn how to detect new botnets [Garcia-Teodoro et al., 2009; Moorthy and Nathiya, 2023] .

Fourth, at the time we conducted this research, we observed numerous published works implementing IPs as a feature in Machine Learning models to detect botnet traffic [Saad et al., 2011; Beigi et al., 2014; Maeda et al., 2019; Gahelot and Dayal, 2020; Joshi et al., 2020; Ismail et al., 2021; Ramesh and Thangaraj, 2023] . However, this approach is conceptually flawed, as bots typically don't set its device IP to avoid raising suspicions or breaking the connection. Moreover, in consumer products, the IP is typically assigned by the Internet Service Provider (IPS). IPs can be used in situations such as cluster creation, or to track packets in traffic connections or flows, but they should not be employed as a standalone feature. In fact, utilizing the IPs as a feature is akin to using the label as a feature in Supervised Learning: Suppose the dataset is composed of the communications between 50 computers, 25 of them infected and 25 clean. Each computer has its own IP, so the ML algorithm would learn that if the communication comes from one of the 25 clean IPs, the traffic is normal, and if the communication originates from the other 25 IPs then it might be infected and then would look to the other features (an infected computer also generates non-infected traffic, from the use of regular programs). This ML model would report a great performance over the dataset, but if this model were to be used in a real environment, the computers would have different IPs. Hence, the model would perform poorly. It could be argued that the destination IP could be a useful feature since all the bots could connect to the same server to get instructions. Nevertheless, these servers tend to rapidly change their IPs to evade reputation filtering [AlAhmadi and Martinovic, 2018; Georgoulias et al., 2023] . This sets up for us the problem of creating a detection sys-

tem that didn't use the IPs as a feature, while also proving that our approach performed better than the IP-user-models from the literature.

Fifth and last difficulty, we noticed a lack of research on efficient botnet detection in ML-based models, i.e., reaching an optimal trade-off between detection rate and speed.

Motivated by these five challenges, we aimed to create a detection system possessing the following attributes:

1. It should be based on the connection behavior, i.e., on characteristics like the frequency of communications, rather than on an analysis of the communicated instructions, so the detector can work even if the packets are ciphered.

2. The system should be able to quickly and automatically learn to detect traffic from new types of botnets, so it can be used right away against new threats. This is the main motivation to use Machine Learning models.

3. The model should not use IPs as a standalone feature, and we also had to provide evidence that this approach is better than the state of the art.

4. The model should be efficient, optimizing the trade-off between the detection rate and the model's operation speed.

1.1.2. Detection of botnet malware in real-time

Following the previous work, the next challenge we considered was detecting botnet traffic in real-time, a crucial task in critical environments where early detection is a must [Moustafa et al., 2018; Sudhakar and Kumar, 2023] . Bearing in mind these environments, two additional problems arise.

The first one is the use case where the network being monitored has a relatively high bandwidth, e.g. 10 Gbps, and real-time analysis is necessary to identify botnet traces. This leads to an important question: what if the network is saturated, i.e., the network is transmitting data at full capacity and it is dropping some packets? Later, the literature research revealed that a network is typically deemed no longer "operational" or usable if packet loss exceeds 10% [Kurose and Ross, 2010; Vyopta, 2018] . Then the challenge is to design an approach that not only can process the data of a high bandwidth network but also that is robust enough to identify botnet traffic even if the network drops up to 10% of the packets [Shah and Issac, 2018; Wei et al., 2023] .

The last problem is to determine the hardware requirements [Maimó et al., 2017; Gil Pérez et al., 2017; Maimó et al., 2018; Wei et al., 2023] for our solution: in particular, the number of CPU cores working in parallel at a given CPU frequency.

After considering these issues, the motivation for this contribution was to build a botnet traffic monitor such that:

1. It is capable of analyzing, in real-time, data from high bandwidth networks.

2. It is robust enough to handle packet loss in saturated networks.

3. Its hardware requirements are clearly defined.

Moreover, since in this work we had to compare our proposal against the literature, we continued with the motivation from the previous one to give evidence of the counter-productive use of the IPs as a standalone feature to detect botnet traffic.

1.1.3. Bot account detection on Twitter using graph data

The raising concerns about bot accounts in social media, and the malicious activities that can be accomplished with them [Rodríguez-Ruiz et al., 2020; Collins, 2018; Esposito et al., 2023; Xie and Li, 2023] , such as spam promotion or harassment towards users, was the original motivation for undertaking this research line. The background from the two works of the previous research line, where the bots interacted among themselves, inspired us to address this problem using graph data: Since the fake users handled by the same entity must avoid being banned if identified as bots, they need to mimic human behavior as closely as possible, including forming communities. This was subsequently confirmed in an initial literature review [Ali Alhosseini et al., 2019; Breuer et al., 2023] , and by noticing that there are bots following one another in publicly available Twitter datasets [Mazza et al., 2019; Feng et al., 2022] .

Our approach using the graph data to detect social bots, i.e., the information about who-follows-who, presented a new challenge: how to properly handle the cross-validation with this type of data. We discovered that usually, the selection of nodes into folds is made randomly, without any consideration for preserving the graph data [Teng et al., 2017; Zhou and Matteson, 2016; Li et al., 2016; Wang, 2021] . While this is not a problem when the nodes are densely connected, it is an issue when it is needed to validate a model on sparsely connected data. Hence, we were motivated to develop a new technique for partitioning the data into folds for cross-validation especially engineered to preserve better the graph data while also preserving the randomness of node selection.

Moreover, we encountered an issue that has been documented in the literature regarding Graph Convolutional Networks (GCNs) [Kipf and Welling, 2017; Hamilton et al., 2017] , which we selected as the foundational model for our classification of nodes in graph data. Specifically, these types of models are sensitive to the presence of nodes of known class near the nodes whose class is to be predicted [Zhang et al., 2019] . In other words, if the model was trained with a subnetwork of users, and used to classify accounts who are barely connected to the training network, the accuracy of the model would drop sharply. Even more if the nodes are not connected at all with the training network. This motivated research on how to address this problem eventually led to a proposal to improve the GCN model that could be applied to any node class prediction problem.

1.1.4. Bot account detection on Twitter based on users behavior

After finishing the previous work, one of the results we found is that the most valuable features to describe Twitter users and detect bot accounts were the ones that the user could not manipulate directly, such as the age of the account. Adding features relative to fields that could be modified by the user – and so the entity behind the social bots – such as the username, did not improve the models. Therefore, the motivation behind this research was to detect bot accounts on Twitter based on the user behavior [Mazza et al., 2019; Feng et al., 2021; Attia et al., 2022; Chawla and Kapoor, 2023; Arin and Kutlu, 2023] , i.e., their activity: their tweets, their retweets, and the liked tweets.

However, during the initial phase of our research, we found little to non-existent labeled datasets that included the tweets, retweets, and likes of the users; and those which included this data only stored a limited number of tweets. This prompted us to collect our own dataset from Twitter, which was then manually labeled. Since we also collected graph data and ensured a high connection density among the collected users, the resulting dataset is, to the best of our knowledge, the most complete and largest Twitter dataset in terms of manually labeled data classifying the users as either humans or bots.

1.2. Objectives

The principal aim of this Thesis is to develop efficient approaches and solutions to detect automated malicious agents that operate on the Internet. Specifically, we concentrate on botnets as networks, which are networks of infected devices (bots) that adhere to the instructions of a criminal entity; and social bots, which are fake accounts on Social Networks utilized to carry out potentially malicious activities.

Following the above-mentioned main objective and the problems and motivations highlighted in the previous Section, we defined the following objectives to propose solutions to these problems. These objectives are classified according to the two main branches of this PhD Thesis.

On bot malware detection:

1. To develop an efficient detector of botnet traffic that does not rely on communications decipherment, and that can be updated easily for new botnets. The aim of this model is to achieve an optimal trade-off between the detection rate and the detection speed.

2. To develop a model capable of detecting botnet traffic in real-time and on relatively high bandwidth networks (up to 10 Gbps) whose usage is aimed at critical infrastructures and backbone networks. This model must also be robust enough to detect bot traces even when up to 10% of the packets are lost due to network saturation.

3. To determine the hardware requirements for the previous model, in particular the necessary number of CPU cores at a given CPU frequency to achieve real-time performance.

4. To create a new dataset of network traffic to test our proposed detection systems.

5. To provide evidence against the use of IPs as a standalone feature to train Machine Learning models in the task of detecting botnet traffic.

On social bots detection:

1. To develop a model capable of detecting bot accounts in X, formerly known as Twitter.

2. To address the classification problem of Graph Convolutional Networks where its accuracy is impacted if the model tries to predict the class of nodes not directly connected, or not connected at all, to the training data.

3. To create a novel algorithm for folds creation in cross-validation problems involving graph-like data, that preserves better the graph structures even on sparsely connected data while maintaining the randomness in the node selection process.

4. To create a new dataset, by collecting data directly from the Social Network formerly known as Twitter and manually labeling it. This dataset would include information such as the public user data, the who-follows-who relationships between the users, or the users' latest publication activity.

1.3. Main contributions

This Section enumerates and summarizes the main contributions of this Thesis.

1. We created two new datasets of botnet and legitimate traffic based on the well-known CTU-13 dataset [Garcia et al., 2014] . The first one, QB-CTU13, addresses the issue of high-class imbalance when the traffic is split into network flows. It includes traffic from legitimate computer usage and traces of seven botnets. The second one, EQB-CTU13, extends QB-CTU13 by adding traces of three more botnets that were chosen because of the challenge their automatic detection poses, based on our literature review.

2. We presented an efficient botnet traffic detector based on Decision Tree (DT), capable of classifying traffic flows in an average of time of 0.78 microseconds each, while achieving an F1 score of 0.85 on the EQB-CTU13 dataset. To devise this model, we conducted (1) a study combining two feature selection methods, Gini Importance and Information Gain, and (2) a study on parameter optimization and model

selection among four Machine Learning classifiers: DT, Random Forest, k-Nearest Neighbors and SVM.

3. We presented a bi-modular architecture capable of detecting botnet traffic that can achieve real-time detection in large bandwidths – up to 10 Gbps. In particular, the interpretation of real-time here means to analyze one second of traffic considering the bandwidth is used at full capacity, taking one second to extract the features and another second to classify the traffic, totaling two seconds to give a report after the traffic is captured.

4. We provided a study analyzing the hardware requirements of our proposed real-time botnet traffic detector, on bandwidths of 100 Mbps, 1 Gbps, and 10 Gbps, and using commercial-grade CPUs operating at a frequency of 2.4 GHz.

5. We provided experimental evidence demonstrating why it is incorrect to use the IPs as a standalone feature in Supervised Learning for botnet traffic detection.

6. We developed a Twitter bot detector that utilizes the "follow" relationships of the accounts. Our detector achieves an F1 score of 0.784 on the Cresci-rtbust dataset [Mazza et al., 2019] , a 24% improvement compared to the baseline method.

7. We proposed a novel approach on Kipf and Welling's Graph Convolutional Network (GCN) [Kipf and Welling, 2017] to solve its limitations in accurately predicting the node class in regions of the graph where training labels are little or non-existent [Zhang et al., 2019] .

8. We developed a new algorithm to separate nodes from a graph (or a set of graphs) into folds for cross-validation. Our approach enhances the preservation of the graph data while ensuring randomness in the selection of nodes. We also provide a comparison of our algorithm with a realistic case to demonstrate its robustness.

9. We created a novel dataset of Twitter data from 2022 that includes the users' public metadata, the who-follows-who relationships among the dataset – thus ensuring relatively dense graph relationships –, and the most recent publication activity (up to 200 publications, and 100 likes). This dataset comprises 17945 samples manually labeled as either bots or humans, making it, to the best of our knowledge, the most complete, recent, and largest in terms of manually labeled data among all the publicly known Twitter datasets so far.

10. We developed a new approach to detect social bots on Twitter based on account publishing behavior. Our approach utilizes a BERT-based predictor for text topics and an ensemble of seven LSTM-based classifiers, which achieves an accuracy of 0.7550 and an F1 of 0.7767 over our newly developed dataset.

1.4. Publications and research results

This section presents the research results obtained during the completion of this doctoral Thesis.

1.4.1. Publications related to this Thesis

- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Efficient Detection of Botnet Traffic by Features Selection and Decision Trees. IEEE Access (2021). DOI: 10.1109/ACCESS.2021.3108222
- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Real-time botnet detection on large network bandwidths using machine learning. Scientific Reports (2023). DOI: 10.1038/s41598-023-31260-0
- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Averting Cold Start: Improved detection of Twitter bots on graph data using Graph Convolutional Networks. To be submitted to a Q1 journal.

1.4.2. Attended conferences

- Oral communication in International Congress: Javier Velasco-Mata, Eduardo Fidalgo, Víctor González-Castro, Enrique Alegre, Pablo Blanco-Medina. Botnet Detection on TCP Traffic Using Supervised Machine Learning. Hybrid Artificial Intelligent Systems (HAIS) 2019.
- Oral communication in International Congress: Sergio Merayo-Alba, Eduardo Fidalgo, Víctor González-Castro, Rocío Alaiz-Rodríguez, Javier Velasco-Mata. Use of Natural Language Processing to Identify Inappropriate Content in Text. Hybrid Artificial Intelligent Systems (HAIS) 2019.

1.4.3. Awards and grants

- FPU (Formación de Profesorado Universitario) grant of the Spanish Government with reference FPU18/05804.

1.4.4. Participation in projects

- "Acuerdo de Colaboración para la puesta en marcha de un equipo de investigación aplicada en visión artificial y reconocimiento de patrones". Addendum 22 to the Framework Agreement between INCIBE (Spanish National Cybersecurity Institute) and the University of León.

- "Acuerdo de Colaboración para la continuidad de los trabajos de un equipo de investigación aplicada en visión artificial y aprendizaje automático". Addendum 01 to the Framework Agreement between INCIBE (Spanish National Cybersecurity Institute) and the University of León.

1.5. Thesis structure

This Thesis is structured in seven Chapters, one Annex with extended results from Chapter 5, the bibliography, and a final Appendix with a summary of the Thesis in Spanish.

- Chapter 1 introduces the work conducted in the Thesis. It presents the motivation behind the four research lines, the corresponding objectives, and the primary contributions of our approaches to address the various considered problems. This Chapter also provides a record of the research outcomes produced during the Thesis, as well as a brief overview of its structure.
- Chapter 2 comprehensively reviews the latest findings in the fields considered in this Thesis. Its first Section compiles the systematic review of relevant literature. The following Sections are dedicated to the three distinct research topics explored in this Thesis: Botnets as networks of computer viruses, botnets as networks of fake social media accounts, and issues and solutions related to working with graph data.
- Chapter 3 presents our work on efficient detection of bot network traffic. It includes the methodology we used to create two new datasets by combining the publicly available CTU-13 dataset [Garcia et al., 2014] of botnet traffic with traffic captures from the Stratosphere IPS [Stratosphere Research Laboratory, 2023] . Furthermore, this Chapter describes the feature selection process to optimize our proposal based on Decision Tree.
- Chapter 4 presents our work on real-time botnet traffic detection. The methodology of this Chapter includes the evaluation of our proposal on saturated networks, i.e., where a percentage – up to 10% – of packets are lost; and the assessment of its hardware requirements.
- Chapter 5 is dedicated to our third work, which initiates the focus on social bots: Detecting fake accounts on Twitter through analyzing the graph formed by the who-follows-who relationships of the users. Here, we describe a novel method to assist GCN models in predicting the class of nodes on graph regions that are either barely connected or not connected at all with the training data, which is an identified weakness of GCNs [Li et al., 2018b; Zhang et al., 2019] . Moreover, in this Chapter we present our novel algorithm to generate folds for graph data that preserves the randomness of selection and a better graph representation.

- Chapter 6 continues the focus on social bots detection but shifts the detection method towards analyzing user behavior. It describes the process of collection and manual labeling of a new Twitter botnet dataset. This Chapter also presents our proposal to identify social bots through an LSTM-based model and an BERT-based encoding of the tweets.
- In Chapter 7 we summarize the work that was completed in this Thesis and discuss the open problems and future work that can be derived from it.
- The Annex A includes a full disclosure of the F1 scores attained by the different configurations of our model proposal of Chapter 5 in search for the most optimal one.
- Appendix 9 is a Summary of the PhD Thesis in Spanish in accordance with the "Normativa para la defensa de la Tesis Doctoral en la Universidad de León" applied to this Thesis.

Chapter 2

State of the art

This Chapter presents the State-of-the-Art review conducted for this Thesis. The initial Subsection outlines the systematic review carried out to select publications from two widely-used databases, namely, Web of Science and IEEE Xplore Digital Library. Section 2.2 provides an overview of works related to detecting malware bots. Following that, Section 2.3 covers the background on detecting social bots. Finally, Section 2.4 discussed the limitations found when using graph data during the research conducted in Chapter 5

2.1. Systematic review

The selection of papers for this State of the Art (SOTA) chapter follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) method[1]. We used two databases for this research: Web of Science (WoS), and IEEE Xplore Digital Library. Sections 2.2, 2.3 and 2.4 presents the result of the review following this methodology, detailed as follows:

Detection of bots as computer viruses For the research based on botnets related to malware, we searched the literature from the Web of Science database from the years 2016 to 2020, both included, searching for the term "botnet" or "botnets" in either the title, abstract or keywords. Then, we filtered the results whose thematic was related to social botnets, i.e., fake and automated accounts in social networks. We also excluded results related to "blockchain" operations, and the "Android" operating system, since the detection of related botnets was not based on network traffic. Finally, results related to the Humanities field were also excluded. The final, reproducible query for the WoS returned 1,642 results, and was the following:

```
( TITLE-ABS-KEY ( botnet ) OR TITLE-ABS-KEY ( botnets ) ) AND ( PUBYEAR
> 2015 AND PUBYEAR < 2021 ) AND ( LIMIT-TO ( SUBJAREA , "COMP" ) OR
LIMIT-TO ( SUBJAREA , "ENGI" ) OR LIMIT-TO ( SUBJAREA , "MATH" ) OR
LIMIT-TO ( SUBJAREA , "DECI" ) OR LIMIT-TO ( SUBJAREA , "MULT" ) ) AND
( LIMIT-TO ( LANGUAGE , "English" ) OR LIMIT-TO ( LANGUAGE , "Spanish"
```

[1] Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) http://www.prisma-statement.org/ Last accessed October 2023.

)) AND (EXCLUDE (EXACTKEYWORD , "Social Networking (online)") OR EXCLUDE (EXACTKEYWORD , "Social Bots") OR EXCLUDE (EXACTKEYWORD , "Social Media") OR EXCLUDE (EXACTKEYWORD , "Twitter") OR EXCLUDE (EXACTKEYWORD , "Android (operating System)") OR EXCLUDE (EXACTKEYWORD , "Blockchain") OR EXCLUDE (EXACTKEYWORD , "Online Social Networks") OR EXCLUDE (EXACTKEYWORD , "Social Aspects") OR EXCLUDE (EXACTKEYWORD , "Human Computer Interaction") OR EXCLUDE (EXACTKEYWORD , "Robotics") OR EXCLUDE (EXACTKEYWORD , "Information Dissemination") OR EXCLUDE (EXACTKEYWORD , "Chatbots") OR EXCLUDE (EXACTKEYWORD , "Android") OR EXCLUDE (EXACTKEYWORD , "Gender Profiling") OR EXCLUDE (EXACTKEYWORD , "Public Opinions"))

After a first preselection involving the reading of title and abstract, a total of 177 works from the WoS remained.

We also used the IEEE Xplore Digital Library to search for documents on the same topic and on the same time interval, from 2016 to 2020 both years included. In this case, the final query involved the search for works that included the term "machine learning", and either "bot" or "botnet". The following query yielded 402 documents:

(botnet OR bot) AND (machine learning) between 2016 - 2020

From these 402 documents, only 280 were not present in the previous search in the WoS database. After a preselection involving reading the title and abstract of the works, a total of 11 documents remained, that were added to the 177 works selected from the WoS search, consolidating a total of 188 works after the preselection.

After reading the body of the 188 preselected papers, a final set of 31 works were selected as relevant and not redundant for consolidating a background. Besides, 7 papers were added following the cascade methodology, i.e., papers that were cited by the selected papers, and were relevant to be included as well. Finally, after the research was done, a total of 61 papers were added to update to more recent years (2021-2023), and to improve the context in the Overview Sections of Chapters 3 and 4, and equivalently the Introduction Sections of the associated published papers [Velasco-Mata et al., 2021, 2023] , mainly adding works published before 2016. Some of the additions were included by request and advice from the peer-reviewers to improve those publications.

Detection of bots as fake users in Social Media This systematic review is related to bots as fake, automated users in Social Networks, in particular Twitter, for its popularity at the time of this research. Initially, this review of the literature only extended up to the year 2021, but was later expanded to include the year 2022 for the fourth work, expounded in

Chapter 6. After conducting this experimentation, we broadened our search to encompass publications up to September 2023 to enhance the State-of-the-Art overview. On one hand, using the WoS database, we crafted a query to search for the terms "twitter" or "social networks", and either "botnet", "bot", or "fake user". The years of publication included from 2019 to 2023, both included, resulting in the following query:

```
( TITLE-ABS-KEY ( ( twitter OR ( social AND networks ) ) AND ( botnet OR bot
OR ( fake AND user ) ) ) ) AND ( PUBYEAR > 2018 AND PUBYEAR < 2024 )
AND ( LIMIT-TO ( SUBJAREA , "COMP" ) OR LIMIT-TO ( SUBJAREA , "ENGI" )
OR LIMIT-TO ( SUBJAREA , "MATH" ) OR LIMIT-TO ( SUBJAREA , "DECI" ) OR
LIMIT-TO ( SUBJAREA , "MULT" ) ) AND ( LIMIT-TO ( LANGUAGE , "English" )
OR LIMIT-TO ( LANGUAGE , "Spanish" ) )
```

The query on the WoS database provided 1,844 results on October 1, 2023, and after a preselection considering the title and abstract of the works, 179 candidates remained.

On the other hand, we used the IEEE Xplore Digital Library database with a more simple query, that searched for documents mentioning "twitter" and "bot", on the same years of publication as with the WoS database:

```
twitter AND bot between 2019-2023
```

This query yield 143 documents on October 1, 2023. A preselection considering the title and abstract of the works left 11 documents that were not already present within in the 179 from the WoS selection, thus summing a total of 190 works.

After reading the body of these documents, we selected 18 to be part of the literature review according to their relevance, non-redundancy, and, more importantly, being reproducible. Moreover, to add context to the overview Sections of Chapters 5 and 6, and after the work in Chapter 5 was submitted as a paper for peer-review and received feed-back from the reviewers, we included 17 more works. Besides, we added 3 more papers following the cascade methodology, i.e., they were cited in the selected papers and were considered relevant. This process also added the works used in Subsection 2.4.1, the background for the problems on Graph Convolutional Networks, mainly employing the cascade method.

Methods for cross-validation on graph data This systematic review was conducted for the contribution in the fifth Chapter of this Thesis, where we proposed a novel crossvalidation method specialized for graph data. We searched for works that used graph data and did a crossvalidation, to see if a similar method already existed.

On the Web of Science database, we searched for works that included the terms "cross-validation" or "cross-validation", and "graph" or "network", and also that included either

of these five terms: "community", "communities", "set", "subset", or "subgraph". The search was limited to the years from 2016 to 2021, both included. The following query yielded 1,744 results, from which a preselection involving reading the title and abstract left us 27 documents relevant to our purpose:

> (TITLE-ABS-KEY ((crossvalidation OR cross-validation) AND (graph OR network) AND (community OR communities OR set OR subset OR subgraph))) AND (PUBYEAR > 2015 AND PUBYEAR < 2022) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "MATH") OR LIMIT-TO (SUBJAREA , "DECI") OR LIMIT-TO (SUBJAREA , "MULT")) AND (LIMIT-TO (LANGUAGE , "English") OR LIMIT-TO (LANGUAGE , "Spanish"))

On the other database we used, IEEE Xplore Digital Library, we used the equivalent query and years, which yielded 617 results, from which only 272 were different from the search on the WoS database. From these 272 documents, a reading on their title and abstract led us to a preselection of 9 works, that were added to the 27 from the preselection on the WoS results, i.e. a total of 36 documents. The IEEE Xplore reproducible query is:

> ("All Metadata":crossvalidation OR "All Metadata":cross-validation) AND ("All Metadata":graph OR "All Metadata":network) AND ("All Metadata":community OR "All Metadata":communities OR "All Metadata":set OR "All Metadata":subset OR "All Metadata":subgraph) between 2016 - 2021

From the 36 documents, a total of 7 documents were selected to consolidate the literature review regarding crossvalidation methods employed in graph data.

2.2. Detection of malware bots

This Section includes the literature review carried out for the works in Chapters 3 and 4. It is divided into three Subsections: The first one is related to the benefits of using a Machine Learning approach to detect botnet traffic, the second one on the optimization of such models through feature selection, and a third one focused on the use of Machine Learning models to detect bot network activity in real-time.

2.2.1. Botnet detection with Machine Learning

Early methods for botnet detection such as BotMiner [Gu et al., 2008a] or BotSniffer [Gu et al., 2008b] relied on statistical algorithms to detect botnet traffic. This was possible

by taking advantage of the time-space correlations of the traffic generated by the malware programs, as opposed to the traffic produced by real humans, which is more random.

Reviewing the latest publications on botnet detection at the time of our research, it can be observed that Machine Learning (ML) algorithms are the most common when the goal is to build multi-class traffic classifiers. Among all of them, models based on Decision Trees (DT) usually achieve the best results. For example, in the work of Gadelrab et al. [Gadelrab et al., 2018] , Support Vector Machine (SVM) and DT were compared to select the best classifier for a botnet detector implementation, called BoTCap. DT achieved the highest performances (i.e., an F1 score of 0.95) over a self-constructed dataset with traces of six botnets: Aryan, Ngr, Rxbot, Blackenergy, Zeus and Vertexnet. Moreover, DT-based algorithms also showed high performance on network attacks datasets, like the CICIDS2017 dataset [Panigrahi and Borah, 2018] , which collects online attacks such as DDoS or malware infections. This dataset was also used in the work of MacKay et al. [McKay et al., 2019] to construct three training sets and two test sets, employed to compare six algorithms: DT, Random Forest (RF), k-Nearest Neighbors (k-NN), Naïve Bayes (NB), Multi-Layer Perceptron (MLP) and One Rule (OneR). The results showed that DT achieved the best F1 score (i.e., 0.99) using the 78 features of the CICIDS2017 dataset, and the authors left feature selection as future work.

Machine Learning algorithms have also been used in other aspects to detect bot activity, such as over the logs of SSH sessions to identify malicious use of commands [Garre et al., 2021] . On the other hand, classifiers based on Deep Learning (DL) were recently proposed for binary classification in botnet detection. For example, the work of Maeda et al. [Maeda et al., 2019] used a Deep Neural Network (DNN) to identify the botnet traffic traces. Maeda et al. reported an accuracy of 0.992 in a dataset built by joining data from the CTU-13 dataset [Garcia et al., 2014] , the ISOT dataset [Saad et al., 2011] and self-captured not-malicious traffic. One of the downsides of detectors based on ML and DL is the possibility of adversarial attacks, when the malicious program output is specially designed to mimic the features of a non-malicious one [Demontis et al., 2019] . Therefore, another benefit of DL-based approaches is the use of Generative Adversarial Networks (GAN), which generate synthetic samples to be mixed within the dataset, so the trained detector is more resilient against crafted attacks. Yin et al. [Yin et al., 2018] proposed a framework to improve botnet detectors by using a GAN called Bot-GAN. They used the ISCX 2014 dataset [Biglar Beigi et al., 2014] divided into 491,381 samples for training and 348,452 samples for testing. They found that, while the F1 score of the detector was 0.6851 when no fake samples were used, they achieved a peak F1 score of 0.7059 if 500 fake samples were mixed with the training data.

To the best of our knowledge, the most used dataset to test botnet detectors is the CTU-13, either pure or with modifications. The CTU-13 is a public dataset created in the Czech Technical University (CTU) [Garcia et al., 2014] , which contains data from seven different botnets: Neris, Rbot, Virut, Murlo, NSIS, Donbot and Sogou. Besides, CTU-13 can either be used *as a whole* or *by scenario*: CTU-13 contains 13 different scenarios, i.e.,

traffic captures carried out in different situations, where the monitored botnet performs one or more actions such as communication between bots, generating SPAM, or performing a DDoS attack.

The current purpose of the CTU-13 dataset is not to improve the detection rate of a previous work. In the recent literature, we found that the minimal accuracy obtained in all the scenarios of the CTU-13 dataset is 0.9643, achieved with a Self Organizing Map (SOM) [Le et al., 2019] . Therefore, the current main utility of this dataset is to verify the detection ability of a solution.

Other botnet types can be used to test if a traffic classifier can improve a previous result. For example, we found that in the work of Abraham et al. [Abraham et al., 2018] the best F1 score obtained for the botnet Bunitu was 0.90 with an ensemble of classifiers. Moreover, the work of AlAhmadi & Martinovic [AlAhmadi and Martinovic, 2018] reported that the recall obtained for the NotPetya botnet using an RF classifier was 0.60, and that 25% of the samples of this botnet were wrongly classified as produced by the botnet Miuref.

2.2.2. Feature selection for optimization

In general, feature selection techniques can be categorized as wrapper, filter, and embedded methods [Li et al., 2018a] .

Wrapper methods search the most optimal feature subset for a given learning algorithm through iteration and evaluating the performance of feature subsets with that algorithm. Their main downside is that the feature subset they obtained is specifically selected for that learning algorithm, and it is not guaranteed to be working well for any different algorithm [Wang et al., 2015] .

Filter methods overcome this problem by not evaluating the feature subset with a learning algorithm. Instead, they rely on characteristics of the data itself, such as entropy or feature correlations. As a result, these methods are usually more computationally efficient than wrapper methods, but the obtained feature subset may not be the most optimal for a given learning algorithm [Ambusaidi et al., 2016] .

Finally, the embedded methods entail a mixture of the two previous ones. The performances of the feature subsets are evaluated with a pre-selected learning algorithm, but it does not use an iterative search as wrapper methods do. Instead, they assign weights to the initial features and try to minimize (or nullify) the value of the weights, which represent the feature importance [Jović et al., 2015] .

In the context of feature selection for botnet detection, different methods have been applied. For example, the work of Gadelrab et al. [Gadelrab et al., 2018] used the Weka framework to implement an embedded feature selector using “BestFirst” as search algorithm and “CfsSubsetEval” as evaluating method. This allowed the creation of feature subsets, which were afterward evaluated with a DT classifier and three SVM classifiers (i.e., with linear, polynomial or Radial Basis Function (RBF) kernels). With an F1 score of

0.95, the best result was achieved with a subset of five features and the DT classifier. The dataset used was built by the authors and was limited to IRC and HTTP based botnets.

Alauthaman et al. [Alauthaman et al., 2018] tested three feature reduction methods: entropy impurity, RelieF and Principal Component Analysis (PCA). The target classification algorithm was a Neural Network (NN), and the most efficient result was obtained using the entropy impurity. The achieved accuracy was 0.9920 after reducing the feature set from 29 to 10 features. The data used was a mixture of traffic traces from the ISOT [Saad et al., 2011] and the ISCX [Shiravi et al., 2012] datasets.

Moreover, Ferhat O. Catak [Catak, 2018] compared five classifiers for malware traffic detection: DT, RF, NN, AdaBoost and NB. They used a filter-based feature selection based on the L1-norm to reduce the feature pool from 36 to 11 features, achieving the highest accuracy with RF (0.8904) followed by DT (0.8677). The dataset used was created by the Australian Centre for Cyber Security (ACCS) [Moustafa and Slay, 2015] .

The work of Letteri et al. [Letteri et al., 2019] focused on improving the F1 score through feature reduction on a NN with two feature selection methods: a Gini Importance derived selection, and Information Gain (also known as Mutual Information). The features were based on time windows by counting the number and type of packets sent to a certain IP inside a time interval. The best results achieved an F1 score of 0.9730 with both the Gini Importance and Information Gain based selections. The dataset used in this work was generated from traffic captures offered by the Stratosphere IPS [Stratosphere Research Laboratory, 2023] .

Feng et al. [Feng et al., 2018] used PCA and an embedded method based on SVM to determine the importance of an initial set of 55 features for cyberattacks detection. The selected datasets were the C08, C09 and C13 ones from the CCC dataset collection [Takata, 2019] , and the feature subsets were evaluated using SVM and RF. They determined the top 10 most important features and concluded that they could reduce the total set of 55 features to 40 without experiencing a significant decrease in the detection rate. After this work, Muhammad et al. [Muhammad et al., 2020] used the CCC dataset collection and also selected a subset of 40 features, but using PCA and Information Gain for feature selection. They tested more models, i.e., RF, SVM, Logistic Regression (LR) and MLP, and were able to overcome the result of Feng et al. using RF and to obtain the highest accuracy of 0.99.

In addition, DL methods can also be used in feature reduction and selection. For example, the work of Parker et al. [Parker et al., 2019] used an AutoEncoder (AE) to obtain a set of 50 new features derived from the original set of 154 features. They combined the two sets and ranked the importance of the 204 features with the Information Gain method. To conclude, they determined that the most optimal solution was an LR classifier using five features, obtaining an F1 score of 0.9801 with a computational training cost of 603.33 seconds on a computer with a 3.1 GHz Intel Core i7 CPU and 8 GB of RAM. The dataset they used was the reduced CLS portion of the AWID dataset [Kolias et al., 2015] .

Finally, state-of-the-art works show that it is possible to implement botnet detectors

using a hyper-reduced set of features. Joshi et al. [Joshi et al., 2020] used five feature selection techniques, producing each a reduced feature subset from their original set. The smallest subset was obtained using PCA, and it had only two features: the source IP and the protocol. Besides, they tested the feature subsets with four classifiers, SVM, LR, k-NN and DT. Using the CTU-13 dataset, k-NN yielded the best results: 0.99 of accuracy using only the subset with two features. A year later, Ismail et al. [Ismail et al., 2021] conducted a similar research using more recent data. They used seven traffic captures from the Stratosphere IPS [Stratosphere Research Laboratory, 2023] , and they also tested more classifiers: Bayesian Network, NB, k-NN, Ada Boost, SVM, J48, RF, Ripper and PART. They concluded that SVM attained the best true positive rate on detecting botnet data across the seven traffic captures, using a subset of only three features: the destination and source IPs and the protocol. Despite these works providing insights on reduced feature sets, they did not analyze the computational cost of their proposals.

2.2.3. Detection of botnet traffic in real-time

Several works have tackled the problem of optimizing botnet detection from different perspectives and considering different aspects of the traffic. While our work is focused on the traffic generated in communications between bots or with the botmaster – the device that controls the bots –, other noteworthy approaches [Alieyan et al., 2017; Vinayakumar et al., 2020; Zago et al., 2020] focused on detecting suspicious DNS requests: in this case, the bots include a string generator or Domain Generator Algorithm (DGA) and the server of the botmaster is hidden behind one or more of the domains that would be generated. Specifically, the bots would try to connect with the botmaster by making DNS requests with each of the generated domains, until one works. Recently, Yin et al. [Yin et al., 2019] developed ConnSpoiler, a system optimized to detect this kind of DNS requests on IoT networks. The detector was based on the Threshold Random Walk (TRW) algorithm, and it was tested over a private dataset made with traces from a Chinese ISP network, which offered Internet services for the education, research, scientific and technical communities, relevant government departments and Hi-Tech Enterprises. The system ran over a single CPU of 2.00 GHz, and it only needed 3.3% of its capacity per device to be monitored, with a false positive rate of about 0.13%. Highnam et al. [Highnam et al., 2021] developed the *Bilbo the "bagging"* model, which combined two neural networks, a CNN and a LSTM, to determine whether a URL is legitimate or generated with a DGA. In their experimentation on four hours of real traffic, Bilbo discovered five potential botnets that commercial tools did not warn about. Finally, DNS-based approaches can be complemented with other methods, creating frameworks such as BotDet [Ghafir et al., 2018] , which combines a DGA detector with other three modules based on blacklists for TOR servers and malicious IPs and SSL certificates.

Regarding approaches that analyze the inter-bot communications, one of them is to analyze traffic from various locations at the same time. This is specially relevant in dis-

tributed environments. One example is BotGuard [Chen et al., 2017a] , a lightweight analyzer for SDN. It gathers the traffic data from switches in the network, which is then analyzed with a detection engine based on a Convex Lens Imaging graph (CLI-graph). According to their experiments on self-generated traffic from an SDN and using the computational power of an Intel i5 CPU, their system is able to detect botnet activity with an accuracy higher than 0.90 and a delay lower than 56 ms.

In IoT communications, Borges et al. [Borges et al., 2022] developed an anomaly detector that only required to measure the number of packets sent by a device in time windows of just 0.1 s. In more detail, the concatenation of these measures makes a time series that is firstly transformed in a series of *symbolic patterns* through an ordinal pattern transformation, which is then analyzed by the detector in search for anomalous patterns. According to their results on the N-BaIoT dataset, they achieved accuracies between 0.985 and 1.000 when they used time series of length $m \geq 400$ measures. As a limitation, they reported that constructing and processing the time series required at least 1 minute before performing the detection. After that, the detection time was 24 ms in the best case and 40 ms in the worst case scenario.

Singh et al. [Singh et al., 2014] developed a scalable implementation using open-source tools such as Hadoop [Apache Software Foundation, 2023a] , Hive [Apache Software Foundation, 2023b] and Mahout [Apache Software Foundation, 2023c] to process high volumes of data. The traffic classification algorithm was Random Forest, and it was tested with CAIDA datasets [Center for Applied Internet Data Analysis, 2023] . This approach was able to extract the features of 1 GB pcap files in 51 s, and, thus, it could process high bandwidths of data in quasi-real-time with a 5 to 30 s of delay, while attaining an accuracy of 0.997.

It is also possible to develop models that work on generic traffic [Gaonkar et al., 2020] such as BotMiner [Gu et al., 2008a] . A frequent strategy to optimize these detectors is the selection of the best features: it reduces the characteristics to extract from the data and also the complexity of the classifier. In [Chen et al., 2017b] the features are selected using the Gini Importance metric, and the classification is made with a Random Forest (RF) classifier. Over the well-known CTU-13 dataset [Garcia et al., 2014] , this approach achieved an accuracy of 0.90 and the RF was able to classify 204,711 samples in 27.1 seconds.

The approach used in Chapter 4 is based on reducing the set of packets considered as a flow, so the feature calculation can be triggered earlier. The most usual way to achieve this is through using time windows: In [Kirubavathi and Anitha, 2016] , Kirubavathi and Anitha use a mixture of data from the ISOT [Alenazi et al., 2017] and CAIDA datasets as well as traces collected from a private setup of infected machines, to test different sizes of time-windows in botnet detection (i.e., from 60 to 300 seconds). They found that the highest performance is achieved on the time windows of 180 s, with a Naive Bayes (NB) classifier that obtained 0.99 of accuracy and 0.02% of false positive rate. Besides, Zhao et al. [Zhao et al., 2013] also studied the feasibility of detecting botnets using a limited number

of packets via time windows. For their work, they used two datasets from The Honeynet Project [The Honeynet Project, 2023] for the botnet traces, and other two datasets from the Traffic Lab at Ericsson Research and from the Lawrence Berkeley National Laboratory (LBNL) for non-malicious traces. Using a Decision Tree (DT) classifier, they found that the false positive rate (FPR) and the true positive rate (TPR) started becoming stable in time windows of 60 s, and that the best performance was achieved at 180 seconds. In this case, the DT achieved a TPR of over 90% and an FPR under 5%. Moreover, Mai and Noh [Mai and Noh, 2018] experimented with the size of the time windows in botnet detection, finding that too many flows were generated if the lengths of the time windows were shorter than 150 seconds, and that if the length was greater than 300 seconds, the accuracy was reduced. The experiments were carried out on the ISOT dataset with an ensemble model that combined k-means and DT, and the peak accuracy obtained was 0.994, being relatively stable around this value if the time windows were inside the range between 150 and 300 seconds. In the same line, a recent work by Nguyen et al. [Nguyen et al., 2022] proposed a collaborative detector that used features based on both network traffic and on computer processes, which also divided the activity on time windows. Their model also peaked in performance using time windows of 180 s.

Finally, other works used time windows shorter than 180 s by combining the flow analysis with other techniques. For example, the multilayer framework proposed by Ibrahim et al. [Ibrahim et al., 2021] combined two modules to detect Command and Control (C&C) servers, which denote the external servers that control botnets and give them commands. The first Filtering module clusters the traffic into known and unknown classes using a semi-supervised Kmeans; and the second one is a "Detecting C&C Server Module" that works on the unknown class. This framework used time windows of one second and achieved a 0.92 F1-score on the CTU-13 dataset. However, due to the clustering process, the detection is not made in real-time.

2.3. Detection of social bots

This Section includes the literature review of Chapters 5 and 6 related to the detection of social bots, i.e., fake accounts in Social Networks.

To detect automated accounts on a social network such as Twitter, there are two main aspects that can be considered: (i) anomalous user features – e.g., an unusual distribution of following and followed users –, and (ii) anomalous behavior – e.g., publishing tweets on short and usually uniform time intervals of a few hours.

Regarding the anomalous features of a user, we noted that this approach has been more widely studied in the literature. The reason lay in the Twitter API, which set limitations on the request for user's data to prevent abuses.

Some examples of these limitations are the impossibility to obtain more than 15 GET requests per time window of 15 minutes to know the followers of an account, or the limit

of 100 statuses per request[2]. This led to a majority of publicly available datasets only containing users metadata, and not the data concerning the user activity, such as its publication activity. Moreover, a module that only needs the user metadata, would need lesser access through the API, making it more efficient.

An alternative using the Twitter API and publicly available datasets is to utilize Botometer[3] to label Twitter users as bots or humans. Botometer acts both as a public repository of datasets from several researchers, and as an online tool to detect Twitter bots, trained with those datasets. Some works have used this tool as the ground truth to train and evaluate their models. For instance, Gera and Sinha [Gera and Sinha, 2022] introduced their T-Bot framework and juxtaposed its efficiency with Botometer, stating that the Botometer API results validate 90% of the suspected bots identified by T-Bot. However, this validation method relies on the precision of a third-party model, Botometer, rather than on a thoroughly tested dataset. As noted by Gallwitx and Kreil [Gallwitz and Kreil, 2022] , Botometer is susceptible to false positives. They analyzed several Botometer-based studies by systematically examining hundreds of Twitter accounts identified as social bots by Botometer, and discovered that the majority of those accounts were undoubtedly operated by human users.

During our literature review, we noted an increasing difficulty to detect Twitter botnets over the years, showing an evolving sophistication. For example, the model of Pham et al. [Pham et al., 2022] Bot2Vec, which combines the word2vec algorithm to create embeddings from the Twitter user data, and a Support Vector Machine (SVM) to use the embeddings, attained an F1 score of 0.9840 when is applied on data from 2015. Schnebly and Sengupta [Schnebly and Sengupta, 2019] obtained an accuracy of 0.9025 with a Random Forest on a mix of data from 2015 and 2017. The data from 2015 was larger (1946 humans, 3457 bots) than the one from 2017 (445 humans and 3202 bots). A posterior work by Fonseca et al. [Abreu et al., 2020] only used data from 2017, and they reported a homogeneous mean accuracy of 0.8549 among three classifiers, RF, SVM and Naïve Bayes.

The ever evolving sophistication of social bots has driven research into more sophisticated detection tools. To cope with the data limitations of the Twitter API, some earlier works tried workarounds that do not need the full records of the publications, i.e., their detection is based on statistics and does not require a deep analysis of the tweet texts. For example, Mazza et al. [Mazza et al., 2019] exploited the time series of a user's activity to detect suspicious patterns. Here, a time series was formed by measuring how much time it took for the user to retweet a given tweet. The expected behavior of a bot is to retweet within a certain time delay, whereas a human can retweet anytime, resulting in a more random pattern. To use these time series, they used an LSTM autoencoder to create embeddings, which were classified using the clustering algorithm HDBSCAN. They called their whole framework RTbust (from Retweet-Buster), which achieved an F1 score of 0.87. The dataset used was self-collected and included 1,000 Italian-speaking users, 51% bots

[2] `https://developer.twitter.com/en/docs/twitter-api` Last time consulted October 2023

[3] Botometer website: `https://botometer.osome.iu.edu/`. Last accessed October 2023.

and 49% humans. Chavoshi et al. [Chavoshi et al., 2016] also developed a detector based on the users' publication activity. In particular, they focused on bots that publish the same content, with only a regular delay between two given bots. To do this, their detector measured the Dynamic Time Warping (DTW) correlation between pairs of users, marking as bots the users whose activity showed a correlation higher than 0.99, which allowed a precision of 0.94. This module worked by retrieving users' data directly from Twitter, and was able to analyze thousands of users per day. Other approaches utilized the information about the users' neighbors (i.e., followers and followed users) in the social network. Alhosseini et al. [Ali Alhosseini et al., 2019] used a 2-layer GCN over a dataset published in 2013 [Yang et al., 2013] which achieved an 0.84 F1 score.

Using the neighbors relations to detect social bots have also been used in other Social Networks. Breuer et al. [Breuer et al., 2023] introduced Preferential Attachment k-class Classifier (PreAttack) to identify bots on the Facebook social network. In particular, their model focuses on the initial attempts to gain connections right after joining the network thorough the friend request feature. Their model achieved an AUC score of 0.9 over their own evaluation framework set to use a single country of $\sim$ 1 million Facebook users.

Later works utilizing datasets containing tweet records propose approaches with deeper analysis of the tweets. Attia et al. [Attia et al., 2022] proposed a dual-input approach to detect Twitter bots using their publications. The first input is the N-grams representation of the tweets' content, and the second input relies solely on the count of words in the tweets. Each input is fed to a CNN model, and the output of both is concatenated to feed a final output layer. They tested their model on the CLEF 2019 dataset [Kosmajac and Keselj, 2020] achieving an accuracy of 0.9325, supposing an increment of 3.22% over their baseline. Moreover, Feng et al. [Feng et al., 2021] combined a deep analysis of the tweet texts using BERT with the graph data information of the Twitter network using a 2-layer GCN obtaining an F1 score 0.87 over dataset they collected in 2020.

Recent works published in parallel to and after our experimentation in Chapter 6 continue the trend in using the publication content to identify social bots on Twitter. In their study, Chawla and Kapoor [Chawla and Kapoor, 2023] propose a model that combines DNA encodings with pretrained BERT embeddings of the tweets to distinguish between spambots and genuine accounts. They achieved an accuracy of 0.7523 on the DynaSent dataset[4]. Moreover, Arin and Kutlu [Arin and Kutlu, 2023] proposed an architecture that combined pretrained GloVe embeddings with LSTM models to utilize the tweets and the description of the Twitter account, along with the user's metadata, to predict if the account is a social bot. They evaluated their proposal on five datasets hosted by Botometer published from 2011 to 2019. The achieved F1 score was 0.80, with 0.83 accuracy.

Finally, a related research line to detecting social bots involves identifying fake news or content on platforms through sentiment analysis. Notably, humans such as trolls or scammers, in addition to social bots, can post such content [Esposito et al., 2023] . In

[4]DynaSent dataset repository `https://github.com/cgpotts/dynasent`. Last accessed October 2023.

their publication, Gupta et al. [Gupta et al., 2023] proposed a hybrid approach that combines LSTM and CNN models to analyze either text or images of news and estimate their possible falsehood through sentiment analysis. They tested their approach on two Kaggle datasets[5]: the MICC-F1220 for images and "Liar, Liar Pants on Fire" for texts, achieving respective accuracies of 0.9136 and 0.9610. Additionally, Xie and Li [Xie and Li, 2023] presented an RNN-based model for detecting fake news texts from various Social Networks. The researched gathered texts from Twitter and Weibo, labeling them manually as either fake or truthful news. In particular, prior to labeling, they used search keywords such as "Really?" or "Is this true?" to obtain more samples suspicious of being fake. Their model achieved an overall F1 score of 0.976.

2.4. Limitations encountered working with graph data

This Section is dedicated to the State of the Art on the limitations we found in the work presented in Chapter 5, particularly on working with graph data. It is divided in two Subsections, the first one related to the limitations of Graph Convolutional Networks (GCN), and the second one on methods to carry out a cross-validation with graph data.

2.4.1. Limitations of GCNs

The idea behind the GCN proposed by Kipf et al. [Kipf and Welling, 2017] is a simplified convolution layer for graph data, which is actually a special form of Laplacian smoothing [Li et al., 2018b] . In particular, instead of using the normalized Laplacian $D^{-1}L$, the convolution is equivalent to using the symmetrically normalized Laplacian $D^{-1/2}LD^{-1/2}$, as shown in Equation (2.1)

$$X^{l+1} = \sigma\left(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}X^{l}W^{l}\right), \tag{2.1}$$

where $\tilde{A} = I + A$ is the adjacency matrix with added self-loops, W^l is the l-th layer specific trainable weight matrix, σ is the activation function, and D is the diagonal degree matrix of $\tilde{A}$, such as $D_{ii} = \sum_j \tilde{A}_{ij}$ [Kipf and Welling, 2017; Li et al., 2018b] .

The smoothing operation mixes the features of a node and its direct neighbors, an operation that makes the embeddings in the same cluster similar. This is both the reason GCN works so well and the source of its limitations. First, the memory needs of training in large graphs could be costly. To address this issue, later reimplementations such as GraphSAGE enabled an efficient mini-batch training [Hamilton et al., 2017] . Secondly, repeating the Laplacian operation could lead to over-smoothing, meaning that the embeddings from different clusters could become indistinguishable. In practice, it has been observed to happen if more than two convolutional layers are used [Zhang et al., 2019; Hamilton et al., 2017] . On a side note, isolated nodes would not suffer this effect as their

[5]Kaggle datasets `https://www.kaggle.com/datasets`. Last accessed October 2023

embeddings are not generated in relation to neighbors but only on their own data. Third, and lastly, the purpose of the model is to serve as a semi-supervised algorithm that assigns labels to previously unlabeled data points, a process known as label propagation. However, GCNs are highly dependent on the validation process: If the GCN is optimized without using the validation set, its performance significantly decays in comparison with a GCN optimized with validation [Li et al., 2018b] . In other words, the GCN ability to propagate labels to the rest of the graph data decays significantly in regions where little or no labels are given. This is even worse if the model were to be applied on entirely different graphs [Zhang et al., 2019] . To solve this problem, our pipeline approach uses a classifier to propose labels on the test graph, helping to optimize the GCN on this set before applying the label propagation.

2.4.2. Cross-validation methods for graph data

There is no predefined method for performing cross-validation on graph data, mainly because there is no unique type of problem with this type of data. For example, the three most common problems with graph data are node prediction [Da Silva et al., 2021] , edge prediction [Hoff, 2007] and community prediction [Developer, 2023b] .

Only the problem of link classification, i.e., to predict whether there will be a link between two nodes, or the class of that link, has a well-known and accepted method to split the data: In essence, since the classifier would take the information of just two nodes to make the prediction, then the nodes can be grouped by pairs and then randomly split the pairs into folds [Hoff, 2007] .

A more complete approach was proposed by K. Chen and J. Lei, called Network Cross-Validation (NCV) approach [Chen and Lei, 2018] . This method focuses on creating testing and training sets rather than creating folds for cross-validation, and it works as follows. First, the nodes are split randomly into two groups, N_1 and N_2. Then, the algorithm observes the node pairs $\{(i,j) : i \in N_1, j \in N_1 \cup N_2\}$, and collects them for the training set. Finally, the node pairs that satisfy $(i,j) : i \in N_2, j \in N_2$ are used as the testing set. Hence, this method preserves the network communities intact, while the training and test sets of edges are independent.

However, for the problems of predicting the class of a node, or a community of nodes, we found neither a consensus nor an approach proven robust. The most simple approach consists of randomly selecting nodes for each fold, and then adding the edges between nodes that belong to the same fold [Developer, 2023a] . This approach can be useful with densely connected data but, outside that case, most of the nodes present a significant decrease in their degree, which leads to data that does not represent the original one with fidelity.

In the case where the node features are already selected, Da Silva et al. recently proposed a spatial based cross-validation [Da Silva et al., 2021] . To split the data, they first use a PCA to set the principal features in which the graph data can be spatially rep-

resented, and then they divide the nodes into folds based on proximity. However, this method is not suitable for a feature selection, as for each feature set, the PCA would give a different spatial representation.

Chapter 3

Efficient botnet detection

This Chapter addresses the efficiency of botnet detection in online traffic data via Machine Learning, particularly tackling two problems. The first one is the feature selection, which we addressed with an adapted embedded method. The second one is to compensate the botnet class imbalance of one of the most used datasets, CTU-13, which lead us to create two new datasets derived from this one, QB-CTU13 and EQB-CTU13.

Due to copyright issues, this chapter has been removed from this Thesis. Here are the details of the corresponding published article:

J. Velasco-Mata, V. González-Castro, E. Fidalgo and E. Alegre, "Efficient Detection of Botnet Traffic by Features Selection and Decision Trees," in IEEE Access, vol. 9, pp. 120567-120579 (2021). DOI: 10.1109/ACCESS.2021.3108222.

Chapter 4

Real-time botnet detection

This Chapter extends the work of the previous one on efficiently detecting botnet traffic, to more challenging scenarios. The main focus is to develop a system to detect botnets on high-capacity networks, and to test it we simulated traffic of up to 10 Gbps by concurrently joining multiple traffic captures. Moreover, we considered the scenario where the network is saturated and starts dropping packets. This could affect a detector trained on traffic flows, i.e., on a series of packets. For this reason, we also tested the robustness of our proposal in this type of scenario.

Due to copyright issues, this chapter has been removed from this Thesis. Here are the details of the corresponding published article:

J. Velasco-Mata, V. González-Castro, E. Fidalgo and E. Alegre, "Real-time botnet detection on large network bandwidths using machine learning", in Scientific Reports, vol. 13, article number 4282 (2023). DOI: 10.1038/s41598-023-31260-0.

Chapter 5

Detection of fake users in Social Networks using graph data

This Chapter introduces a new line of research in this Thesis: the detection of botnets in Social Networks. These types of bots open fake accounts and pose as humans, using the API – or something alike – to interact with the network. It is easy to make a fake account that looks "human" at first glance: the easiest way is to copy the profile of a real account. For this reason, the focus on this contribution has shifted to something that a fake account cannot completely control: the relationships between users. To work with graph data, we improved the baseline model Graph Convolutional Network (GCN) [Ali Alhosseini et al., 2019] . We also developed a new way to generate folds for cross-validation, specifically designed to preserve the graph structure of the data.

Due to copyright issues, as the corresponding manuscript will be submitted for publication, this chapter has been removed from this Thesis.

Chapter 6

User behavior based social bot detection

This Chapter extends the research made on social botnets in Chapter 5, in particular on the Twitter platform, currently known as X, at the time this research was conducted. The main objective is to distinguish bot from human users based on their behavior, represented by their publications. For this purpose, we created a novel Twitter dataset by collecting samples from the social network and manually labeling them. This dataset is, to the best of our knowledge, the largest Twitter dataset to the day of publishing this Thesis in terms of manually labeled samples.

Due to copyright issues, as the corresponding manuscript will be submitted for publication, this chapter has been removed from this Thesis.

Chapter 7

Conclusions and outlook

7.1. Work summary

The focus of this PhD Thesis is the identification of different types of bots, which are automated entities that operate following the remote commands of a botmaster, often with malicious purposes, hence the motivation for their detection. The four works conducted during the PhD program can be categorized into two primary research lines: detection of malware botnets, and identification of social botnets.

The first research line concerns detecting malware botnets, which are computer viruses that operate on the victims' devices. It encompasses the first two works. The first one focuses on optimizing Machine Learning classifiers by selecting appropriate features to create an efficient detector of bot network traffic. This work is detailed in Chapter 3. The procedure to select the best subset of features involved the combination of two feature ranking methods, i.e., Gini Importance and Information Gain. The output indicated that the optimal subset would include the top five to seven most informative features. With this information, we analyzed the performance of three classifiers, namely Decision Tree, Random Forest, and k-Nearest Neighbors with each of the three feature subsets. The Decision Tree utilizing a five feature subset performed the best among the candidates, obtaining an average F1 score of 0.85 and classifying each sample in a mean of 0.78 microseconds.

The second work, developed in Chapter 4, builds upon the previous research on detecting bot malware, shifting its focus to attaining real-time detection. Specifically, the aim was to develop a Machine Learning classifier that could extract the features of one second of network traffic at most one second later, and identify bot connections in that traffic at most one additional second later, i.e., needing at most two seconds to give the report. Moreover, we also required the model to be able to work in high bandwidth networks – we simulated up to 10 Gbps –, and also to be robust enough to detect bot traces even in saturated conditions, meaning that the bandwidth was used up to 110% of its maximum capacity, thus the network was dropping up to 10% of the packets. Our model succeeded in these requirements, attaining an F1 score of 0.926 over the EQB-CTU13 dataset we crafted on the first research of this PhD. The estimated hardware requirements for our model to work in networks of 10 Gbps are 19 CPU cores operating at 2.4 GHz.

The second research line, comprising the work done about the detection of fake, au-

tomated accounts in social media, usually called social bots, includes the last two of the four works.

The initial objective of the third work was to develop a bot account detector for Twitter, enhancing the baseline that employed Kipf and Welling's Graph Convolutional Network (GCN) [Kipf and Welling, 2017] . However, upon further investigation, we identified two issues. Firstly, GCNs have constraints in predicting the class of nodes from regions of the graph with low or no edge connections to the training nodes [Zhang et al., 2019] . And secondly, we found a lack of methods to split graph data into folds that ensured adequate preservation of the graph structure. We proposed solutions for these problems that can be applied to any type of graphs, generalizing from working with social network data. With our ideas, we developed a model that outperformed the baseline F1 score by 24%.

The fourth and last work shifts focus towards detecting fake accounts on Twitter through user behavior analysis. Our proposed detector considers the topic and the frequency of the account publications, analyzed by means of BERT and LSTM models, respectively. At the time of our experimentation, the existing datasets had drawbacks that did not make it possible to carry out an analysis combining both the relationships between samples (i.e., the existing connections between them), and their publications. Either they contained little information, i.e., mostly just metadata from Twitter accounts or, if they did contain information about the publications, labels were incomplete or were not annotated manually. Hence, in order to ensure that our model was evaluated on a dataset containing all the required information, such as a record of the users' most recent publications, we created a new dataset with manually labeled Twitter data. This dataset is the largest among the publicly available Twitter datasets in terms of manually labeled data to the best of our knowledge. Our model achieved an accuracy of 0.7550 and an F1 score of 0.7767 on this dataset.

7.2. Summary of the contributions

The main contributions presented in this Thesis are:

1. We built two new datasets of botnet and legitimate traffic based on the well-known CTU-13 dataset [Garcia et al., 2014] . Our first dataset, QB-CTU13, aims to address the problem of high-class imbalance when traffic is divided into network flows. It encompasses traffic originating from legitimate computer usage, followed by manually recognized traces from seven botnets. The EQB-CTU13 extends QB-CTU13 by incorporating traces from three additional botnets, which were selected based on their difficulty in being detected by automatic detectors, according to our literature review.

2. We propose an efficient botnet traffic detector based on a Decision Tree algorithm that can classify traffic flows in an average time of 0.78 microseconds each, achiev-

ing an F1 score of 0.85 over the EQB-CTU13 dataset.

3. We present a bi-modular architecture based on Decision Tree capable of analyzing the traffic in real time to identify botnet activity. It can be escalated to operate in large bandwidths (i.e., up to 10 Gbps), requiring 19 CPU cores operating at a frequency of 2.4 GHz.

4. We propose a novel approach built upon Kipf and Welling's Graph Convolutional Network (GCN) [Kipf and Welling, 2017] to solve its limitations in accurately predicting the node class in regions of the graph where the connections to the training nodes are little to non-existent [Zhang et al., 2019] . Applied to the problem of social bot detection, our detector improves the F1 score of the baseline GCN [Ali Alhosseini et al., 2019] by 24% on the Cresci-rtbust dataset [Mazza et al., 2019] .

5. We have developed a new algorithm for partitioning nodes from a graph (or a set of graphs) into folds for cross-validation. Our approach improves the preservation of edge data while ensuring randomness in the selection of nodes. Additionally, we present a study to demonstrate the robustness of our algorithm.

6. We built a novel dataset of Twitter data from 2022 that includes users' public metadata, follower relationships across the dataset – ensuring a relatively dense graph structure – and recent publication activity (up to 200 publications and 100 likes). This dataset comprises 17,945 samples which were manually labeled as either bots or humans, making it, to the best of our knowledge, the most complete, recent and largest in terms of manually labeled data of all publicly known Twitter datasets.

7. We have developed a novel method to identify social bots on Twitter through their behavior. Our system uses a BERT-based predictor to identify the topics of tweets and an ensemble of seven LSTM-based classifiers to analyze the publication frequency. Our detector achieves an accuracy of 0.7550 and an F1 score of 0.7767 on our new dataset.

7.3. Open problems and future work

The first part of this Thesis, on malware bot detection using Machine Learning, concluded with two proposals, an efficient detector and a real-time detector. The experiments and the comparison with other models were all done using Python 3. Although these results are enough for a proof of concept, the models could become even more efficient in the time dimension if they were built using a compiled language such as C/C++ or Rust.

Furthermore, an advantage of Machine Learning classifiers is that new instances can be retrained with new data and replace the older instances. This allows these types of

detectors to quickly adapt to new variations of the known botnets, or even to detect new types of bots. A new line of research would involve measuring the evolution of the efficiency of such a scheme over time.

Our work on detecting malware bots has focused on analyzing traffic at the flow level, i.e., analyzing communications between two devices. However, bots operate in networks, and thus there is relevant information that can be obtained by observing the entire network. This type of analysis is not particularly suitable for real-time detection, as it would require monitoring the network for longer periods of time and extracting correlations. But it could be complementary to the researched flow-based detectors, so that sysadmins could get two types of reports on this aspect of the network: real-time reports for early detection, and more detailed reports later.

In the second part of this Thesis, we proposed two social bot identifiers, specially designed for Twitter. In future work, these models could be adapted for other social networks.

The second proposal of a social bot detector focuses on the accounts' behavior, in particular on their publication history and the discussed topics. However, this proposal did not differentiate between malicious bots and benign or legitimate bots, for example accounts to report the weather, events or other information of interest. In future works, new proposals can be designed to make this distinction.

Moreover, there are more dimensions to analyze in terms of behavior, such as conversations engaged in by users. This opens up a wide area of research to explore new ways to characterize the behavior and employ it for bot detection.

Annex A: F1 scores of the pipelines

This appendix includes the tables with the complete disclosure of F1 scores of the pipelines on the three cross-validations conducted in Chapter 5, and the test over the Cresci-rtbust dataset. In each test or validation over a set, the experiment was conducted five times to account for the initial randomness of the neural network before training. Because of this reason, the means and the standard deviations are the results of 25 experiments in the case of the five-fold cross-validation, 15 experiments in the case of the two three-fold cross-validations, and five experiments in the case of the test over Cresci-rtbust.

Table A.1: Results on 5-fold seed-based cross-validation of the pipeline RF-GCN

Pipeline RF-GCN		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.963 ± 0.003	0.958 ± 0.004	0.952 ± 0.006	0.941 ± 0.008
	botdetective	$\mathbf{0.970 \pm 0.003}$	0.966 ± 0.004	0.960 ± 0.009	0.955 ± 0.010
	complete	0.969 ± 0.003	0.968 ± 0.003	0.963 ± 0.007	0.958 ± 0.008
botdetective	detectme	0.962 ± 0.006	0.958 ± 0.006	0.953 ± 0.005	0.937 ± 0.012
	botdetective	0.964 ± 0.006	0.960 ± 0.006	0.956 ± 0.006	0.946 ± 0.011
	complete	0.968 ± 0.005	0.964 ± 0.005	0.960 ± 0.007	0.954 ± 0.007
complete	detectme	0.963 ± 0.006	0.957 ± 0.007	0.951 ± 0.008	0.940 ± 0.011
	botdetective	0.966 ± 0.006	0.962 ± 0.007	0.956 ± 0.009	0.949 ± 0.009
	complete	0.967 ± 0.005	0.964 ± 0.007	0.960 ± 0.007	0.952 ± 0.007

Table A.2: Results on 5-fold seed-based cross-validation of the pipeline RF-GCN+RF

Pipeline RF-GCN+RF		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.978 ± 0.002	0.976 ± 0.003	0.973 ± 0.004	0.968 ± 0.007
	botdetective	0.978 ± 0.004	0.977 ± 0.005	0.974 ± 0.005	0.968 ± 0.009
	complete	$\mathbf{0.979 \pm 0.004}$	0.978 ± 0.005	0.975 ± 0.005	0.968 ± 0.008
botdetective	detectme	0.978 ± 0.004	0.975 ± 0.003	0.973 ± 0.004	0.967 ± 0.006
	botdetective	0.977 ± 0.004	0.975 ± 0.005	0.971 ± 0.005	0.963 ± 0.010
	complete	0.978 ± 0.004	0.975 ± 0.005	0.972 ± 0.006	0.965 ± 0.008
complete	detectme	0.978 ± 0.003	0.976 ± 0.005	0.972 ± 0.004	0.966 ± 0.005
	botdetective	0.978 ± 0.004	0.975 ± 0.005	0.971 ± 0.006	0.964 ± 0.010
	complete	0.978 ± 0.004	0.975 ± 0.006	0.972 ± 0.006	0.965 ± 0.010

Table A.3: Results on 5-fold seed-based cross-validation of the pipeline RF-GCN+SVM

Pipeline+SVM RF-GCN+SVM		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.976 ± 0.003	0.974 ± 0.004	0.970 ± 0.007	0.961 ± 0.011
	botdetective	0.977 ± 0.005	0.974 ± 0.009	0.970 ± 0.012	0.960 ± 0.020
	complete	$\mathbf{0.978 \pm 0.005}$	0.974 ± 0.009	0.970 ± 0.011	0.962 ± 0.017
botdetective	detectme	0.977 ± 0.006	0.974 ± 0.005	0.969 ± 0.006	0.958 ± 0.015
	botdetective	0.975 ± 0.007	0.971 ± 0.009	0.964 ± 0.015	0.956 ± 0.020
	complete	0.976 ± 0.008	0.972 ± 0.010	0.966 ± 0.014	0.956 ± 0.022
complete	detectme	0.977 ± 0.004	0.973 ± 0.007	0.968 ± 0.009	0.956 ± 0.017
	botdetective	0.975 ± 0.008	0.970 ± 0.013	0.965 ± 0.016	0.955 ± 0.022
	complete	0.976 ± 0.008	0.970 ± 0.013	0.964 ± 0.019	0.955 ± 0.024

Table A.4: Results on test set of the pipeline RF-GCN

Pipeline RF-GCN		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.665 ± 0.017	0.630 ± 0.016	0.605 ± 0.022	0.667 ± 0.024
	botdetective	0.608 ± 0.005	0.600 ± 0.007	0.601 ± 0.021	0.604 ± 0.006
	complete	0.614 ± 0.011	0.599 ± 0.008	0.607 ± 0.013	0.587 ± 0.032
botdetective	detectme	$\mathbf{0.697 \pm 0.015}$	0.675 ± 0.015	0.631 ± 0.014	0.642 ± 0.028
	botdetective	0.608 ± 0.011	0.619 ± 0.021	0.615 ± 0.007	0.581 ± 0.009
	complete	0.609 ± 0.011	0.629 ± 0.013	0.600 ± 0.017	0.579 ± 0.033
complete	detectme	0.672 ± 0.017	0.657 ± 0.009	0.607 ± 0.050	0.608 ± 0.035
	botdetective	0.592 ± 0.008	0.600 ± 0.011	0.588 ± 0.024	0.582 ± 0.014
	complete	0.629 ± 0.007	0.612 ± 0.006	0.598 ± 0.024	0.575 ± 0.033

Table A.5: Results on test set of the pipeline RF-GCN+RF

Pipeline RF-GCN+RF		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.773 ± 0.008	0.771 ± 0.008	0.753 ± 0.007	0.708 ± 0.029
	botdetective	0.742 ± 0.006	0.742 ± 0.006	0.750 ± 0.004	0.733 ± 0.008
	complete	0.739 ± 0.007	0.750 ± 0.010	0.749 ± 0.011	0.743 ± 0.015
botdetective	detectme	$\mathbf{0.784 \pm 0.013}$	0.753 ± 0.010	0.725 ± 0.016	0.688 ± 0.015
	botdetective	0.740 ± 0.009	0.738 ± 0.011	0.744 ± 0.006	0.731 ± 0.013
	complete	0.739 ± 0.006	0.741 ± 0.006	0.738 ± 0.008	0.725 ± 0.009
complete	detectme	0.763 ± 0.006	0.732 ± 0.017	0.717 ± 0.022	0.694 ± 0.018
	botdetective	0.741 ± 0.008	0.743 ± 0.012	0.740 ± 0.003	0.715 ± 0.023
	complete	0.743 ± 0.006	0.746 ± 0.013	0.741 ± 0.020	0.712 ± 0.034

Table A.6: Results on test set of the pipeline RF-GCN+SVM

Pipeline RF-GCN+SVM		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.731 ± 0.004	$\mathbf{0.732 \pm 0.006}$	0.723 ± 0.007	0.720 ± 0.015
	botdetective	0.721 ± 0.009	0.714 ± 0.009	0.645 ± 0.075	0.691 ± 0.017
	complete	0.717 ± 0.005	0.714 ± 0.025	0.710 ± 0.010	0.691 ± 0.023
botdetective	detectme	0.731 ± 0.006	0.724 ± 0.009	0.724 ± 0.007	0.715 ± 0.009
	botdetective	0.713 ± 0.009	0.699 ± 0.014	0.697 ± 0.019	0.617 ± 0.110
	complete	0.719 ± 0.006	0.711 ± 0.021	0.710 ± 0.011	0.685 ± 0.027
complete	detectme	0.731 ± 0.012	0.731 ± 0.006	0.714 ± 0.012	0.711 ± 0.010
	botdetective	0.712 ± 0.008	0.706 ± 0.018	0.695 ± 0.027	0.647 ± 0.050
	complete	0.714 ± 0.010	0.697 ± 0.034	0.715 ± 0.013	0.615 ± 0.117

Table A.7: Results on the 3-fold origin-based cross-validation of the pipeline RF-GCN

Pipeline RF-GCN		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.810 ± 0.110	0.807 ± 0.111	0.807 ± 0.100	0.785 ± 0.091
	botdetective	0.817 ± 0.112	0.802 ± 0.115	$\mathbf{0.819 \pm 0.115}$	0.749 ± 0.141
	complete	0.814 ± 0.113	0.811 ± 0.112	0.784 ± 0.123	0.761 ± 0.131
botdetective	detectme	0.792 ± 0.113	0.786 ± 0.111	0.782 ± 0.108	0.769 ± 0.095
	botdetective	0.742 ± 0.077	0.719 ± 0.107	0.728 ± 0.098	0.765 ± 0.110
	complete	0.754 ± 0.073	0.744 ± 0.097	0.733 ± 0.109	0.737 ± 0.134
complete	detectme	0.776 ± 0.121	0.765 ± 0.124	0.782 ± 0.108	0.748 ± 0.061
	botdetective	0.715 ± 0.110	0.704 ± 0.109	0.741 ± 0.098	0.737 ± 0.114
	complete	0.714 ± 0.125	0.712 ± 0.119	0.729 ± 0.115	0.735 ± 0.139

Table A.8: Results on the 3-fold origin-based cross-validation of the pipeline RF-GCN+RF

Pipeline RF-GCN+RF		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.846 ± 0.098	$\mathbf{0.847 \pm 0.095}$	0.842 ± 0.092	0.827 ± 0.091
	botdetective	0.834 ± 0.101	0.829 ± 0.100	0.819 ± 0.100	0.800 ± 0.104
	complete	0.834 ± 0.101	0.830 ± 0.098	0.821 ± 0.097	0.798 ± 0.106
botdetective	detectme	0.828 ± 0.103	0.818 ± 0.105	0.815 ± 0.101	0.819 ± 0.097
	botdetective	0.795 ± 0.113	0.785 ± 0.120	0.777 ± 0.124	0.762 ± 0.134
	complete	0.801 ± 0.111	0.788 ± 0.118	0.778 ± 0.124	0.778 ± 0.121
complete	detectme	0.816 ± 0.106	0.801 ± 0.111	0.811 ± 0.101	0.818 ± 0.093
	botdetective	0.798 ± 0.113	0.785 ± 0.120	0.770 ± 0.131	0.763 ± 0.131
	complete	0.799 ± 0.114	0.793 ± 0.116	0.782 ± 0.121	0.773 ± 0.126

Table A.9: Results on the 3-fold origin-based cross-validation of the pipeline RF-GCN+SVM

Pipeline RF-GCN+SVM		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	$\mathbf{0.855 \pm 0.099}$	0.854 ± 0.096	0.850 ± 0.092	0.844 ± 0.088
	botdetective	0.832 ± 0.102	0.818 ± 0.102	0.816 ± 0.097	0.796 ± 0.105
	complete	0.828 ± 0.102	0.825 ± 0.099	0.819 ± 0.098	0.790 ± 0.111
botdetective	detectme	0.843 ± 0.102	0.844 ± 0.099	0.844 ± 0.095	0.838 ± 0.092
	botdetective	0.784 ± 0.117	0.779 ± 0.121	0.774 ± 0.126	0.764 ± 0.129
	complete	0.791 ± 0.117	0.782 ± 0.121	0.777 ± 0.123	0.782 ± 0.113
complete	detectme	0.845 ± 0.100	0.841 ± 0.102	0.838 ± 0.095	0.836 ± 0.091
	botdetective	0.798 ± 0.112	0.787 ± 0.118	0.783 ± 0.119	0.773 ± 0.119
	complete	0.794 ± 0.116	0.792 ± 0.116	0.786 ± 0.115	0.779 ± 0.119

Table A.10: Results on 3-fold seed-based cross-validation of the pipeline RF-GCN

Pipeline RF-GCN		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.943 ± 0.018	0.938 ± 0.019	0.930 ± 0.022	0.921 ± 0.021
	botdetective	0.950 ± 0.014	0.949 ± 0.014	0.943 ± 0.016	0.938 ± 0.017
	complete	0.953 ± 0.012	0.950 ± 0.013	0.949 ± 0.014	0.942 ± 0.015
botdetective	detectme	0.943 ± 0.022	0.936 ± 0.022	0.930 ± 0.022	0.912 ± 0.033
	botdetective	0.950 ± 0.011	0.945 ± 0.013	0.937 ± 0.014	0.926 ± 0.024
	complete	$\mathbf{0.954 \pm 0.012}$	0.946 ± 0.015	0.941 ± 0.016	0.938 ± 0.015
complete	detectme	0.944 ± 0.019	0.935 ± 0.022	0.928 ± 0.022	0.918 ± 0.024
	botdetective	0.951 ± 0.012	0.941 ± 0.017	0.938 ± 0.017	0.929 ± 0.019
	complete	0.951 ± 0.011	0.946 ± 0.014	0.945 ± 0.012	0.933 ± 0.017

Table A.11: Results on 3-fold seed-based cross-validation of the pipeline RF-GCN+RF

Pipeline RF-GCN+RF		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.968 ± 0.007	0.965 ± 0.008	0.961 ± 0.007	0.954 ± 0.008
	botdetective	0.970 ± 0.007	0.968 ± 0.008	0.965 ± 0.007	0.958 ± 0.009
	complete	$\mathbf{0.971 \pm 0.007}$	0.967 ± 0.007	0.965 ± 0.008	0.958 ± 0.010
botdetective	detectme	0.968 ± 0.007	0.965 ± 0.008	0.962 ± 0.007	0.953 ± 0.011
	botdetective	0.969 ± 0.006	0.966 ± 0.006	0.961 ± 0.006	0.955 ± 0.010
	complete	0.969 ± 0.006	0.967 ± 0.007	0.962 ± 0.007	0.955 ± 0.010
complete	detectme	0.967 ± 0.008	0.964 ± 0.008	0.960 ± 0.007	0.951 ± 0.011
	botdetective	0.969 ± 0.007	0.967 ± 0.006	0.962 ± 0.007	0.953 ± 0.011
	complete	0.970 ± 0.006	0.967 ± 0.007	0.963 ± 0.006	0.954 ± 0.011

Table A.12: Results on 3-fold seed-based cross-validation of the pipeline RF-GCN+SVM

Pipeline RF-GCN+SVM		Threshold			
ML feat. set	GCN feat. set	0.80	0.85	0.90	0.95
detectme	detectme	0.968 ± 0.009	0.964 ± 0.010	0.960 ± 0.010	0.946 ± 0.018
	botdetective	0.967 ± 0.011	0.966 ± 0.011	0.962 ± 0.013	0.948 ± 0.025
	complete	0.967 ± 0.012	0.965 ± 0.012	0.960 ± 0.016	0.947 ± 0.026
botdetective	detectme	0.968 ± 0.010	0.964 ± 0.010	0.959 ± 0.012	0.947 ± 0.018
	botdetective	0.968 ± 0.010	0.962 ± 0.012	0.957 ± 0.015	0.943 ± 0.029
	complete	0.968 ± 0.010	0.964 ± 0.012	0.956 ± 0.016	0.943 ± 0.028
complete	detectme	$\mathbf{0.968 \pm 0.009}$	0.965 ± 0.010	0.956 ± 0.013	0.940 ± 0.023
	botdetective	0.966 ± 0.012	0.962 ± 0.012	0.954 ± 0.019	0.941 ± 0.027
	complete	0.969 ± 0.009	0.961 ± 0.015	0.954 ± 0.019	0.938 ± 0.032

Bibliography

Abraham, B., Mandya, A., Bapat, R., Alali, F., Brown, D. E., and Veeraraghavan, M. (2018). A comparison of machine learning approaches to detect botnet traffic. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Abreu, J. V. F., Ralha, C. G., and Gondim, J. J. C. (2020). Twitter bot detection with reduced feature set. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE.

AlAhmadi, B. A. and Martinovic, I. (2018). Malclassifier: Malware family classification using network flow sequence behaviour. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–13. IEEE.

Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., and Hossain, M. A. (2018). A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, 29(11):991–1004.

Albin, E. and Rowe, N. C. (2012). A realistic experimental comparison of the suricata and snort intrusion-detection systems. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 122–127. IEEE.

Alenazi, A., Traore, I., Ganame, K., and Woungang, I. (2017). Holistic model for HTTP botnet detection based on DNS traffic analysis. In Springer, editor, *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pages 1–18.

Ali Alhosseini, S., Bin Tareaf, R., Najafi, P., and Meinel, C. (2019). Detect me if you can: Spam bot detection using inductive representation learning. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 148–153.

Alieyan, K., ALmomani, A., Manasrah, A., and Kadhum, M. M. (2017). A survey of botnet detection based on DNS. *Neural Computing and Applications*, 28(7):1541–1558.

Alomari, D., Anis, F., Alabdullatif, M., and Aljamaan, H. (2023). A survey on botnets attack detection utilizing machine and deep learning models. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 493–498.

Alshamkhany, M., Alshamkhany, W., Mansour, M., Khan, M., Dhou, S., and Aloul, F. (2020). Botnet attack detection using machine learning. In *2020 14th International Conference on Innovations in Information Technology (IIT)*, pages 203–208. IEEE.

Ambusaidi, M. A., He, X., Nanda, P., and Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE transactions on computers*, 65(10):2986–2998.

Anderson, B. and McGrew, D. (2016). Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pages 35–46. ACM.

Apache Software Foundation (2023a). Hadoop. https://hadoop.apache.org Last accessed Sep 2022.

Apache Software Foundation (2023b). Hive. https://hive.apache.org/ Last accessed Sep 2023.

Apache Software Foundation (2023c). Mahout. https://mahout.apache.org/ Last accessed Sep 2023.

Arin, E. and Kutlu, M. (2023). Deep learning based social bot detection on twitter. *IEEE Transactions on Information Forensics and Security*, 18:1763–1772.

Attia, S. M., Mattar, A. M., and Badran, K. M. (2022). Bot detection using multi-input deep neural network model in social media. In *2022 13th international conference on electrical engineering (ICEENG)*, pages 71–75. IEEE.

Aydın, M. A., Zaim, A. H., and Ceylan, K. G. (2009). A hybrid intrusion detection system design for computer network security. *Computers & Electrical Engineering*, 35(3):517–526.

Bederna, Z. and Szadeczky, T. (2019). Cyber espionage through botnets. *Security Journal*, pages 1–20.

Beigi, E. B., Jazi, H. H., Stakhanova, N., and Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255. IEEE.

Besel, C., Echeverria, J., and Zhou, S. (2018). Full cycle analysis of a large-scale botnet attack on twitter. In IEEE, editor, *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 170–177.

Biglar Beigi, E., Hadian Jazi, H., Stakhanova, N., and Ghorbani, A. A. (2014). Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255.

Borges, J. B., Medeiros, J. P., Barbosa, L. P., Ramos, H. S., and Loureiro, A. A. (2022). Iot botnet detection based on anomalies of multiscale time series dynamics. *IEEE Transactions on Knowledge and Data Engineering*.

Breuer, A., Khosravani, N., Tingley, M., and Cottel, B. (2023). Preemptive detection of fake accounts on social networks via multi-class preferential attachment classifiers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 105–116, New York, NY, USA. Association for Computing Machinery.

Catak, F. (2018). Two-layer malicious network flow detection system with sparse linear model based feature selection. *Journal of the National Science Foundation of Sri Lanka*, 46(4).

Center for Applied Internet Data Analysis (2023). CAIDA Data Collection. https://catalog.caida.org/ Last accessed Sep 2023.

Chavoshi, N., Hamooni, H., and Mueen, A. (2016). Debot: Twitter bot detection via warped correlation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 817–822. IEEE.

Chawla, V. and Kapoor, Y. (2023). A hybrid framework for bot detection on twitter: Fusing digital dna with bert. *Multimedia Tools and Applications*, pages 1–24.

Chen, J., Cheng, X., Du, R., Hu, L., and Wang, C. (2017a). BotGuard: lightweight real-time botnet detection in software defined networks. *Wuhan University Journal of Natural Sciences*, 22(2):103–113.

Chen, K. and Lei, J. (2018). Network cross-validation for determining the number of communities in network data. *Journal of the American Statistical Association*, 113(521):241–251.

Chen, L., Wang, Q., Song, Y., and Chen, J. (2023). Security is readily to interpret: Quantitative feature analysis for botnet encrypted malicious traffic. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 753–758. IEEE.

Chen, R., Niu, W., Zhang, X., Zhuo, Z., and Lv, F. (2017b). An effective conversation-based botnet detection method. *Mathematical Problems in Engineering*, 2017.

Collins, T. (2018). Twitter's huge bot problem is out of the bag. `https://www.cnet.com/tech/tech-industry/twitter-bot-fake-followers-problem-out-of-the-bag/` Last accessed Jan 2023.

Conger, K. and Hirsh, L. (2022). Elon musk completes $44 billion deal to own twitter. New York Times.

Corse, A., Eaton, C., and Purnell, N. (2023). Elon musk replaces twitter's blue bird with an 'X'. Wall Street Journal.

Da Silva, T. P., Parmezan, A. R. S., and Batista, G. E. A. P. A. (2021). A graph-based spatial cross-validation approach for assessing models learned with selected features to understand election results. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 909–915.

Dean, B. (2022). How many people use twitter in 2022? Published in Backlinko `https://backlinko.com/twitter-users` Last accessed Jan 2023.

Demontis, A., Melis, M., Biggio, B., Maiorca, D., Arp, D., Rieck, K., Corona, I., Giacinto, G., and Roli, F. (2019). Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Transactions on Dependable and Secure Computing*, 16(4):711–724.

Developer, N. (2023a). Node classification with GDSL. `https://neo4j.com/developer/graph-data-science/node-classification/` Last accessed Jan 2023.

Developer, N. (2023b). Same community. https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/same-community/ Last accessed Jan 2023.

Esposito, C., Moscato, V., and Sperlì, G. (2023). Detecting malicious reviews and users affecting social reviewing systems: A survey. *Computers & Security*, 133:103407.

FBI (2022). 2021 Internet Crime Report. FBI's Internet Crime Complaint Center (IC3).

Feng, S., Tan, Z., Wan, H., Wang, N., Chen, Z., Zhang, B., Zheng, Q., Zhang, W., Lei, Z., Yang, S., et al. (2022). Twibot-22: Towards graph-based twitter bot detection. *Advances in Neural Information Processing Systems*, 35:35254–35269.

Feng, S., Wan, H., Wang, N., and Luo, M. (2021). BotRGCN: Twitter bot detection with relational graph convolutional networks. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 236–239.

Feng, Y., Akiyama, H., Lu, L., and Sakurai, K. (2018). Feature selection for machine learning-based early detection of distributed cyber attacks. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 173–180. IEEE.

Gadelrab, M. S., ElSheikh, M., Ghoneim, M. A., and Rashwan, M. (2018). BotCap: Machine learning approach for botnet detection based on statistical features. *International Journal of Computer Network and Information Security (IJCNIS)*, 10(3):563–579.

Gahelot, P. and Dayal, N. (2020). Flow based botnet traffic detection using machine learning. In Springer, editor, *Proceedings of ICETIT 2019*, pages 418–426.

Gallwitz, F. and Kreil, M. (2022). Investigating the validity of botometer-based social bot studies. In *Multidisciplinary International Symposium on Disinformation in Open Online Media*, pages 63–78. Springer.

Gao, Y., Xu, G., Li, L., Luo, X., Wang, C., and Sui, Y. (2022). Demystifying the underground ecosystem of account registration bots. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 897–909.

Gaonkar, S., Dessai, N. F., Costa, J., Borkar, A., Aswale, S., and Shetgaonkar, P. (2020). A survey on botnet detection techniques. In IEEE, editor, *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–6.

Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28.

Garre, J. T. M., Pérez, M. G., and Ruiz-Martínez, A. (2021). A novel machine learning-based approach for the detection of SSH botnet infection. *Future Generation Computer Systems*, 115:387–396.

Georgoulias, D., Pedersen, J. M., Falch, M., and Vasilomanolakis, E. (2023). Botnet business models, takedown attempts, and the darkweb market: a survey. *ACM Computing Surveys*, 55(11):1–39.

Gera, S. and Sinha, A. (2022). T-bot: Ai-based social media bot detection model for trend-centric twitter network. *Social Network Analysis and Mining*, 12(1):76.

Ghafir, I., Prenosil, V., Hammoudeh, M., Baker, T., Jabbar, S., Khalid, S., and Jaf, S. (2018). Botdet: A system for real time botnet command and control traffic detection. *IEEE Access*, 6:38947–38958.

Gil Pérez, M., Huertas Celdrán, A., Ippoliti, F., Giardina, P. G., Bernini, G., Marco Alaez, R., Chirivella-Perez, E., García Clemente, F. J., Martínez Pérez, G., Kraja, E., et al. (2017). Dynamic reconfiguration in 5G mobile networks to proactively detect and mitigate botnets. *IEEE Internet Computing*, 21(5):28–36.

Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008a). BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th conference on Security symposium*, pages 139–154.

Gu, G., Zhang, J., and Lee, W. (2008b). BotSniffer: Detecting botnet command and control channels in network traffic. *Computer Science and Engineering Faculty Publications*.

Gupta, S., Verma, B., Gupta, P., Goel, L., Yadav, A. K., and Yadav, D. (2023). Identification of fake news using deep neural network-based hybrid model. *SN Computer Science*, 4(5):679.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Highnam, K., Puzio, D., Luo, S., and Jennings, N. R. (2021). Real-time detection of dictionary DGA network traffic using deep learning. *SN Computer Science*, 2(2):1–17.

Hoff, P. (2007). Modeling homophily and stochastic equivalence in symmetric relational data. *Advances in neural information processing systems*, 20.

Ibrahim, W. N. H., Anuar, S., Selamat, A., Krejcar, O., Crespo, R. G., Herrera-Viedma, E., and Fujita, H. (2021). Multilayer framework for botnet detection using machine learning algorithms. *IEEE Access*, 9:48753–48768.

Ismail, Z., Jantan, A., Yusoff, M. N., and Kiru, M. U. (2021). The effects of feature selection on the classification of encrypted botnet. *Journal of Computer Virology and Hacking Techniques*, 17(1):61–74.

Jiménez, J. M. H. and Goseva-Popstojanova, K. (2018). The effect on network flows-based features and training set size on malware detection. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–9. IEEE.

Joshi, C., Bharti, V., and Ranjan, R. K. (2020). Analysis of feature selection methods for p2p botnet detection. In *International Conference on Advances in Computing and Data Sciences*, pages 272–282. Springer.

Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205. IEEE.

Keenan, A. (2022). Elon musk texted before twitter deal that 'drastic' action was needed to tackle bots. Yahoo! Finance.

Khan, R. U., Zhang, X., Kumar, R., Sharif, A., Golilarz, N. A., and Alazab, M. (2019). An adaptive multilayer botnet detection technique using machine learning classifiers. *Applied Sciences*, 9(11):2375.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pages 1–14.

Kirubavathi, G. and Anitha, R. (2016). Botnet detection via mining of traffic flow characteristics. *Computers & Electrical Engineering*, 50:91–101.

Kolias, C., Kambourakis, G., Stavrou, A., and Gritzalis, S. (2015). Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208.

Koroniotis, N., Moustafa, N., and Sitnikova, E. (2019). Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions. *IEEE Access*, 7:61764–61785.

Kosmajac, D. and Keselj, V. (2020). Twitter user profiling: Bot and gender identification: Notebook for pan at clef 2019. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*, pages 141–153. Springer.

Kouvela, M., Dimitriadis, I., and Vakali, A. (2020). Bot-detective: An explainable twitter bot detection service with crowdsourcing functionalities. In *Proceedings of the 12th International Conference on Management of Digital EcoSystems*, pages 55–63.

Kurose, J. and Ross, K. (2010). *Computer Networking: A Top-Down Approach*. New York, Addison-Wesley.

Le, D. C., Zincir-Heywood, A. N., and Heywood, M. I. (2019). Unsupervised monitoring of network and service behaviour using self organizing maps. *Journal of Cyber Security and Mobility*, 8(1):15–52.

Lee, S., Abdullah, A., Jhanjhi, N., and Kok, S. (2021). Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning. *PeerJ Computer Science*, 7:e350.

Letteri, I., Della Penna, G., and Caianiello, P. (2019). Feature selection strategies for HTTP botnet traffic detection. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 202–210. IEEE.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2018a). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94.

Li, Q., Han, Z., and Wu, X.-M. (2018b). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 3538–3545.

Li, W., Jin, J., and Lee, J.-H. (2019). Analysis of botnet domain names for IoT cybersecurity. *IEEE Access*, 7:94658–94665.

Li, Y., Zhu, M., Luo, X., Yin, L., and Fu, Y. (2023). A privacy-preserving botnet detection approach in largescale cooperative iot environment. *Neural Computing and Applications*, 35(19):13725–13737.

Li, Z., Liu, Z., Zhong, W., Huang, M., Wu, N., Xie, Y., Dai, Z., and Zou, X. (2016). Large-scale identification of human protein function using topological features of interaction network. *Scientific reports*, 6(1):1–11.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

Lourenço, M. B. and Marinos, L. (2020). ENISA Threat Lited by 1andscape 2020 - Botnet. Technical Report ISBN: 978-92-9204-354-4, ENISA, Attiki, Greece.

Maeda, S., Kanai, A., Tanimoto, S., Hatashima, T., and Ohkubo, K. (2019). A botnet detection method on SDN using deep learning. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE.

Mai, L. and Noh, D. K. (2018). Cluster ensemble with link-based approach for botnet detection. *Journal of Network and Systems Management*, 26(3):616–639.

Maimó, L. F., García Clemente, F. J., Gil Pérez, M., and Martínez Pérez, G. (2017). On the performance of a deep learning-based anomaly detection system for 5G networks. In IEEE, editor, *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–8.

Maimó, L. F., Gómez, Á. L. P., Clemente, F. J. G., Pérez, M. G., and Pérez, G. M. (2018). A self-adaptive deep learning-based system for anomaly detection in 5g networks. *IEEE Access*, 6:7700–7712.

Martínez Torres, J., Iglesias Comesaña, C., and García-Nieto, P. J. (2019). Machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics*, 10(10):2823–2836.

Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., and Tesconi, M. (2019). Rtbust: Exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, page 183–192, New York, NY, USA. Association for Computing Machinery.

McKay, R., Pendleton, B., Britt, J., and Nakhavanit, B. (2019). Machine learning algorithms on botnet traffic: Ensemble and simple algorithms. In *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, pages 31–35. ACM.

Moorthy, R. S. S. and Nathiya, N. (2023). Botnet detection using artificial intelligence. *Procedia Computer Science*, 218:1405–1413.

Moustafa, N., Adi, E., Turnbull, B., and Hu, J. (2018). A new threat intelligence scheme for safeguarding industry 4.0 systems. *IEEE Access*, 6:32910–32924.

Moustafa, N. and Slay, J. (2015). UNSW-NB15: a comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6. IEEE.

Muhammad, A., Asad, M., and Javed, A. R. (2020). Robust early stage botnet detection using machine learning. In *2020 International Conference on Cyber Warfare and Security (ICCWS)*, pages 1–6. IEEE.

Nembrini, S., König, I. R., and Wright, M. N. (2018). The revival of the gini importance? *Bioinformatics*, 34(21):3711–3718.

Nguyen, G. L., Dumba, B., Ngo, Q.-D., Le, H.-V., and Nguyen, T. N. (2022). A collaborative approach to early detection of iot botnet. *Computers & Electrical Engineering*, 97:107525.

Padhiar, S., Shah, A., and Patel, R. (2023). The hidden enemy: A botnet taxonomy. In *Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences: PCCDS 2022*, pages 93–100. Springer.

Panigrahi, R. and Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482.

Parker, L., Yoo, P., Asyhari, T., Chermak, L., Jhi, Y., and Taha, K. (2019). DEMISe: Interpretable deep extraction and mutual information selection techniques for IoT intrusion detection. In *ARES'19 Proceedings of the 14th International Conference on Availability, Reliability and Security*. ACM.

Pham, P., Nguyen, L. T., Vo, B., and Yun, U. (2022). Bot2vec: a general approach of intra-community oriented representation learning for bot detection in different types of social networks. *Information Systems*, 103:101771.

Ramesh, R. B. and Thangaraj, S. J. J. (2023). Analyzing and detecting botnet attacks using anomaly detection with machine learning. In *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 911–915. IEEE.

Rauti, S. and Laato, S. (2023). Understanding software obfuscation and diversification as defensive measures for the cybersecurity of internet of things. In *Hawaii International Conference on System Sciences*, pages 6645–6654.

Rodríguez-Ruiz, J., Mata-Sánchez, J. I., Monroy, R., Loyola-González, O., and López-Cuevas, A. (2020). A one-class classification approach for bot detection on twitter. *Computers & Security*, 91:101715.

Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238.

Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W., Felix, J., and Hakimian, P. (2011). Detecting P2P botnets through network behavior analysis and machine learning. In *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pages 174–180. IEEE.

Schnebly, J. and Sengupta, S. (2019). Random forest twitter bot classifier. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0506–0512. IEEE.

Shafi, Q. and Basit, A. (2019). DDoS botnet prevention using blockchain in software defined Internet of Things. In IEEE, editor, *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 624–628.

Shah, S. A. R. and Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems*, 80:157–170.

Shevtsov, A., Oikonomidou, M., Antonakaki, D., Pratikakis, P., Kanterakis, A., Fragopoulou, P., and Ioannidis, S. (2022). Discovery and classification of twitter bots. *SN Computer Science*, 3:255.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374.

Singh, K., Guntuku, S. C., Thakur, A., and Hota, C. (2014). Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences*, 278:488–497.

Srinivas, P., Das, A., and Pulabaigari, V. (2022). Fake spreader is narcissist; real spreader is machiavellian prediction of fake news diffusion using psycho-sociological facets. *Expert systems with applications*, 207:117952.

Stevanovic, M. and Pedersen, J. M. (2014). An efficient flow-based botnet detection using supervised machine learning. In *2014 international conference on computing, networking and communications (ICNC)*, pages 797–801. IEEE.

Stratosphere Research Laboratory (2023). Stratosphere IPS. https://www.stratosphereips.org/ Last accessed Sep 2023.

Sudhakar and Kumar, S. (2023). Abbdiot: Anomaly-based botnet detection using machine learning model in the internet of things network. In *International Conference on IoT, Intelligent Computing and Security: Select Proceedings of IICS 2021*, pages 235–245. Springer.

Takata, Y. (last visited October 2019). MWS datasets 2016 anti-malware engineering workshop (MWS). `www.iwsec.org/mws/2016/20160714-takata-dataset.pdf`.

Tardelli, S., Avvenuti, M., Tesconi, M., and Cresci, S. (2022). Detecting inorganic financial campaigns on twitter. *Information Systems*, 103:101769.

Teng, F., Tang, R., Yang, Y., Wang, H., and Dai, R. (2017). Temporal prediction model for social information propagation. In *Rough Sets*, pages 465–476, Cham. Springer International Publishing.

The Honeynet Project (2023). Projects. https://www.honeynet.org/ Last accessed Sep 2023.

TS, S. M. and Sreeja, P. (2022). Bert-lstm for fake news detection on facebook using svd. In *2022 International Conference on Connected Systems & Intelligence (CSI)*, pages 1–7. IEEE.

Unhackthevote (2019). Twitter's algorithms help push bots into real accounts. `https://www.unhackthevote.com/twitters-algorithms-help-push-bots-into-real-accounts/` Last accessed Jan 2023.

Velasco-Mata, J., Fidalgo, E., González-Castro, V., Alegre, E., and Blanco-Medina, P. (2019). Botnet detection on TCP traffic using supervised machine learning. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 444–455. Springer.

Velasco-Mata, J., González-Castro, V., Fidalgo, E., and Alegre, E. (2023). Real-time botnet detection on large network bandwidths using machine learning. *Scientific Reports*, 13(1):4282.

Velasco-Mata, J., González-Castro, V., Fernández, E. F., and Alegre, E. (2021). Efficient detection of botnet traffic by features selection and decision trees. *IEEE Access*, 9:120567–120579.

Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q.-V., Padannayil, S. K., and Simran, K. (2020). A visualized botnet detection system based deep learning for the internet of things networks of smart cities. *IEEE Transactions on Industry Applications*, 56(4):4436–4456.

Vyopta (2018). Troubleshooting packet loss: How much is an acceptable amount? https://www.vyopta.com/blog/video-conferencing/understanding-packet-loss/ Last accessed Nov 2022.

Wainwright, P. and Kettani, H. (2019). An analysis of botnet models. In *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, pages 116–121. ACM.

Wang, A., An, N., Chen, G., Li, L., and Alterovitz, G. (2015). Accelerating wrapper-based feature selection with k-nearest-neighbor. *Knowledge-Based Systems*, 83:81–91.

Wang, Y. (2021). Predicting the degradation of covid-19 mrna vaccine with graph convolutional networks. In *2021 6th International Conference on Machine Learning Technologies*, ICMLT 2021, page 111–116, New York, NY, USA. Association for Computing Machinery.

Wei, C., Xie, G., and Diao, Z. (2023). A lightweight deep learning framework for botnet detecting at the iot edge. *Computers & Security*, page 103195.

Wen, P.-P., Shi, S.-P., Xu, H.-D., Wang, L.-N., and Qiu, J.-D. (2016). Accurate in silico prediction of species-specific methylation sites based on information gain feature optimization. *Bioinformatics*, 32(20):3107–3115.

Xie, B. and Li, Q. (2023). Detecting fake news by rnn-based gatekeeping behavior model on social networks. *Expert Systems with Applications*, page 120716.

Yang, C., Harkreader, R., and Gu, G. (2013). Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293.

Yin, C., Zhu, Y., Liu, S., Fei, J., and Zhang, H. (2018). An enhancing framework for botnet detection using generative adversarial networks. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 228–234.

Yin, L., Luo, X., Zhu, C., Wang, L., Xu, Z., and Lu, H. (2019). ConnSpoiler: Disrupting C&C communication of IoT-Based botnet through fast detection of anomalous domain queries. *IEEE Transactions on Industrial Informatics*.

Zago, M., Gil Pérez, M., and Martínez Pérez, G. (2020). Umudga: A dataset for profiling dga-based botnet. *Computers & Security*, 92:101719.

Zaheer, A., Tahir, S., Almufareh, M. F., and Hamid, B. (2023). A hybrid model for botnet detection using machine learning. In *2023 International Conference on Business Analytics for Technology and Security (ICBATS)*, pages 1–8. IEEE.

Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., and Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39:2–16.

Zhou, Z. and Matteson, D. S. (2016). Predicting melbourne ambulance demand using kernel warping. *The Annals of Applied Statistics*, 10(4):1977–1996.

Appendix

SUMMARY OF THE THESIS IN SPANISH

RESUMEN DE LA TESIS EN CASTELLANO

En cumplimiento de la disposición transitoria de la "Normativa para la defensa de la Tesis Doctoral en la Universidad de León", aprobada por Consejo de Gobierno de 9 de noviembre de 2017 y modificada por Consejo de Gobierno de 16 de julio de 2021, y del artículo 19.3 del "Reglamento de las enseñanzas oficiales de doctorado y del título de doctor de la Universidad de León", aprobado por acuerdo del Consejo de Gobierno de fecha 25 de septiembre de 2012, se incluye un resumen de esta Tesis en castellano en este anexo. Asimismo, para dar cumplimiento completo al citado artículo, se ha incluido un índice en castellano al comienzo del documento.

1 Introducción

1.1 Motivación

El motivo original de esta Tesis Doctoral partió del interés por utilizar la Inteligencia Artificial para detectar y prevenir amenazas en el ámbito de los ataques de Denegación de Servicio Distribuido (DDoS), propagación de virus informáticos, y actividades maliciosas a través de Internet en general. Esta motivación se centró posteriormente en las botnets, es decir, redes de ordenadores infectados por malware especializado en controlar remotamente los dispositivos de las víctimas bajo el control de un atacante; ya que era un interés del INCIBE (Instituto Nacional de Ciberseguridad) y el trabajo estaba parcialmente financiado por la Adenda 22, un convenio entre la Universidad de León y el INCIBE, posteriormente renovado y renombrado como Adenda 01. Previamente a iniciar el doctorado, durante el Trabajo Fin de Máster se realizó una investigación sobre botnets que posteriormente fue mejorada, enviada y presentada en un congreso internacional [Velasco-Mata et al., 2019] . El primer trabajo presentado en el Capítulo 3 de esta Tesis es una continuación del trabajo presentado en dicha conferencia.

Tras la investigación sobre las redes de bots como virus informáticos, se puso el foco de atención en una de las tendencias más recientes relativas a la actividad delictiva distribuida en Internet: Bots como cuentas falsas en redes sociales, que estaban siendo utilizadas para promover spam, acoso hacia usuarios normales, o campañas de odio, entre otros ejemplos. Esto sirvió de motivación para continuar la investigación hacia el uso de Inteligencia Artificial para detectar este tipo de cuentas.

El trabajo realizado en esta tesis doctoral se divide en cuatro capítulos principales: dos relacionados con la detección de botnets como redes de ordenadores infectados por este tipo de malware, y dos relacionados con la detección de "social bots", es decir, usuarios falsos automatizados en Social Media. En concreto, se ha centrado en la plataforma Twitter, ahora llamada X. A continuación se detalla con más detalle el propósito y la motivación de estas cuatro líneas de investigación.

1.1.1 Deteccion eficiente de malware botnet

Este trabajo se centra en el problema de detectar el tráfico de red generado por un bot, es decir, un programa malicioso instalado en el dispositivo de la víctima sin su consentimiento y normalmente también sin su conocimiento [Padhiar et al., 2023; Garcia et al., 2014] . Proponer una solución es un reto debido a varias dificultades.

En primer lugar, los programas *bot* suelen estar diseñados para ser sigilosos, operando lo más silenciosamente posible para evadir la detección [Koroniotis et al., 2019; Li et al., 2019; Alomari et al., 2023; Georgoulias et al., 2023] . El motivo puede encontrarse en el ciclo habitual de creación de redes de bots: los ciberdelincuentes empiezan por crear una red de bots sin llamar la atención y, una vez que es lo suficientemente grande para sus fines [Lourenço and Marinos, 2020; Padhiar et al., 2023] , dirigen los bots en un ataque coordinado y masivo, por ejemplo un DDoS dirigido a un servidor [Li et al., 2023] . Detectar las redes de bots en la fase inicial es difícil debido a su diseño intrínsecamente sigiloso.

En segundo lugar, las botnets contemporáneas utilizan protocolos de cifrado en sus comunicaciones [Anderson and McGrew, 2016; Chen et al., 2023] , por lo que detectar instrucciones en sus mensajes no es posible a menos que se conozca un método de descifrado. Un enfoque genérico debería, por tanto, centrarse en los patrones en el flujo de paquetes de red en lugar de detectar patrones en datos descifrados.

En tercer lugar, y relacionado con el segundo, es difícil analizar manualmente una muestra de malware bot para crear firmas, es decir, patrones que buscar que delatan la actividad de un bot [Roesch et al., 1999; Albin and Rowe, 2012; Aydın et al., 2009; Zaheer et al., 2023] . Esto implica analizar a través de la ofuscación del programa si la hay [Liao et al., 2013; Rauti and Laato, 2023] , obtener un un método para descifrar los mensajes si es posible, y examinar de las instrucciones comunicadas entre un bot y el resto de su botnet. Es preferible contar con un enfoque automatizado para aprender a detectar nuevas redes de bots [Garcia-Teodoro et al., 2009; Moorthy and Nathiya, 2023] .

En cuarto lugar, en el momento en que realizamos esta investigación, observamos numerosos trabajos publicados que implementaban las direcciones IP como característica en modelos de Aprendizaje Automático para detectar el tráfico de botnets [Saad et al., 2011; Beigi et al., 2014; Maeda et al., 2019; Gahelot and Dayal, 2020; Joshi et al., 2020; Ismail et al., 2021; Ramesh and Thangaraj, 2023] . Sin embargo, este enfoque es conceptualmente defectuoso, ya que los bots no suelen configurar la IP de su dispositivo para evitar levantar sospechas o romper la conexión. Además, en los productos de consumo, la IP suele asignarla el proveedor de servicios de Internet (IPS). Las IPs pueden utilizarse en situaciones como la creación de clústeres o para rastrear paquetes en conexiones o flujos de tráfico, pero no deben emplearse como una función independiente. De hecho, utilizar las IP como característica es similar a utilizar la etiqueta como característica en el aprendizaje supervisado: Supongamos que el conjunto de datos se compone de las comunicaciones entre 50 ordenadores, 25 de ellos infectados y 25 limpios. Cada ordenador

tiene su propia IP, por lo que el algoritmo de Aprendizaje Automático aprendería que si la comunicación procede de una de las 25 IP limpias, el tráfico es normal, y si la comunicación se origina en las otras 25 IP, entonces podría estar infectada y entonces recurriría a las otras características para decidir (un ordenador infectado también genera tráfico no infectado, por el uso de programas normales). Este modelo tendría un gran rendimiento en el conjunto de datos, pero si se utilizara en un entorno real, los ordenadores tendrían IP diferentes. Por lo tanto, el modelo funcionaría mal. Podría argumentarse que la IP de destino podría ser una característica útil, ya que todos los bots podrían conectarse al mismo servidor para obtener instrucciones. Sin embargo, estos servidores tienden a cambiar rápidamente sus IP para eludir ser filtrados por mala reputación [AlAhmadi and Martinovic, 2018; Georgoulias et al., 2023] . Esto planteó el problema de crear un sistema de detección que no utilizara las IP como característica, al tiempo que demostró que nuestro enfoque funcionaba mejor que los modelos IP-usuario de la bibliografía.

En quinto y último lugar, se observó una falta de trabajos de investigación en la literatura sobre la detección eficiente de botnets en modelos basados en Aprendizaje Automático, es decir, alcanzar un equilibrio óptimo entre tasa de detección y velocidad.

Debido a por estos cinco aspectos, se propuso crear un sistema de detección que poseyera las siguientes propiedades:

1. Debe basarse en el comportamiento de la conexión, es decir, en características como la frecuencia de las comunicaciones, más que en un análisis de las instrucciones comunicadas, para que el detector pueda funcionar aunque los paquetes de red estén cifrados.

2. El sistema debe ser capaz de aprender rápida y automáticamente a detectar el tráfico de nuevos tipos de botnets, de modo que pueda utilizarse de inmediato contra las nuevas amenazas. Esta es la principal motivación para utilizar modelos de Aprendizaje Automático.

3. El modelo no debe utilizar las IP como característica independiente, y se tienen que aportar pruebas de que este enfoque es mejor que el estado de la técnica.

4. El modelo debe ser eficiente, optimizando el compromiso entre la tasa de detección y la velocidad de funcionamiento del modelo.

1.1.2 Detección de malware botnet en tiempo real

Siguiendo el trabajo anterior, el planteamiento del siguiente fue detectar el tráfico de botnets en tiempo real, una tarea crucial en entornos críticos donde la detección temprana es imprescindible [Moustafa et al., 2018; Sudhakar and Kumar, 2023] . Teniendo en cuenta estos entornos, surgen dos problemas adicionales.

El primero es el caso de uso en el que la red vigilada tiene un ancho de banda relativamente alto, por ejemplo 10 Gbps, y es necesario realizar análisis en tiempo real para

identificar rastros de actividad botnet. Esto lleva a una pregunta importante: ¿qué pasa si la red está saturada, es decir, la red está transmitiendo datos a plena capacidad y está dejando caer algunos paquetes? Posteriormente, la investigación bibliográfica reveló que normalmente se considera que una red ya no es "operativa" o utilizable si la pérdida de paquetes supera el 10% [Kurose and Ross, 2010; Vyopta, 2018] . Entonces el reto es diseñar un enfoque que no sólo pueda procesar los datos de una red de gran ancho de banda, sino que también sea lo suficientemente robusto como para identificar el tráfico de botnets incluso si la red pierde hasta un 10% de los paquetes [Shah and Issac, 2018; Wei et al., 2023] .

El último problema consiste en determinar los requisitos de hardware [Maimó et al., 2017; Gil Pérez et al., 2017; Maimó et al., 2018; Wei et al., 2023] para nuestra solución: en concreto, el número de núcleos de CPU que trabajan en paralelo a una frecuencia de CPU determinada.

Después de considerar estas cuestiones, la motivación de esta contribución fue construir un monitor de tráfico de botnet tal que:

1. Es capaz de analizar, en tiempo real, datos procedentes de redes de gran ancho de banda.

2. Es lo suficientemente robusto como para soportar la pérdida de paquetes en redes saturadas.

3. Sus requisitos de hardware están claramente definidos.

Además se continuó el objetivo presentado en el anterior trabajo de dar pruebas del uso contraproducente de las IP como característica independiente para detectar el tráfico de botnets.

1.1.3 Detección de cuentas bot en Twitter usando datos de tipo grafo

La creciente preocupación debida al creciente numero de cuentas bot en las redes sociales, y las actividades maliciosas que se pueden realizar con ellas [Rodríguez-Ruiz et al., 2020; Collins, 2018; Esposito et al., 2023; Xie and Li, 2023] , como por ejemplo la promoción de spam o el acoso a los usuarios, fue la motivación original para emprender esta línea de investigación. Los antecedentes de los dos trabajos de la línea de investigación anterior, en los que los bots interactuaban entre sí, sirvieron de inspiración para abordar este problema utilizando datos de grafos: Dado que los usuarios falsos manejados por la misma entidad deben evitar ser baneados si se les identifica como bots, necesitan imitar el comportamiento humano lo más fielmente posible, incluyendo la formación de comunidades. Esto se confirmó posteriormente en una revisión bibliográfica inicial [Ali Alhosseini et al., 2019; Breuer et al., 2023] , y al observar que hay bots que se siguen unos a otros en conjuntos de datos de Twitter disponibles públicamente [Mazza et al., 2019; Feng et al., 2022] .

Esta aproximación de usar de los datos de grafos para detectar bots sociales, es decir, la información sobre quién-sigue-a-quién, presentó un nuevo reto: cómo manejar adecuadamente la validación cruzada con este tipo de datos. Se notó que, por lo general, la selección de nodos en particiones se realiza de forma aleatoria, sin tener en cuenta la preservación de los datos del grafo [Teng et al., 2017; Zhou and Matteson, 2016; Li et al., 2016; Wang, 2021] . Si bien esto no es un problema cuando los nodos están densamente conectados, es un problema cuando se necesita para validar un modelo en datos escasamente conectados. Por lo tanto, se desarrolló una nueva técnica para dividir los datos en pliegues para la validación cruzada, especialmente diseñada para preservar mejor los datos de los gráficos y, al mismo tiempo, preservar la aleatoriedad de la selección de nodos.

Además, durante la experimentación se encontró con un problema que había sido documentado en la literatura con respecto a las redes convolucionales de grafos (GCN) [Kipf and Welling, 2017; Hamilton et al., 2017] , el cual se seleccionó como modelo base para nuestra clasificación de nodos en datos de grafos. En particular, este tipo de modelos son sensibles a la presencia de nodos de clase conocida cerca de los nodos cuya clase se quiere predecir [Zhang et al., 2019] . En otras palabras, si el modelo se entrenara con una subred de usuarios y se utilizara para clasificar cuentas que apenas están conectadas a dicha red de entrenamiento, la precisión del modelo caería notablemente. Más aún si los nodos no están conectados en absoluto con la red de entrenamiento. Esta investigación motivada sobre cómo abordar este problema condujo finalmente a una propuesta para mejorar el modelo GCN que podría aplicarse a cualquier problema de predicción de clases de nodos.

1.1.4 Detección de cuentas bot en Twitter basado en el comportamiento de los usuarios

Uno de los resultados que se desprendieron del trabajo anterior es que las características más valiosas para describir a los usuarios de Twitter y detectar cuentas de bots eran aquellas que el usuario no podría manipular directamente, por ejemplo la antigüedad de la cuenta. Añadir características relativas a campos que sí elusuario si podía modificar – y por tanto la entidad detrás de los bots sociales –, por ejemplo el nombre de usuario, no mejoraba los modelos. Por lo tanto, la motivación de esta investigación era detectar cuentas de bots en Twitter basándose en el comportamiento del usuario [Mazza et al., 2019; Feng et al., 2021; Attia et al., 2022; Chawla and Kapoor, 2023; Arin and Kutlu, 2023] , es decir, su actividad: sus tweets, retweets y likes.

Sin embargo, durante la fase inicial de esta investigación se encontraron escasos conjuntos de datos etiquetados que incluyeran los tweets, retweets y likes de los usuarios; y los que incluían estos datos sólo almacenaban un número limitado de tweets. Esto nos llevó a recopilar nuestro propio conjunto de datos de Twitter, que luego etiquetamos manualmente. Dado que también recopilamos datos gráficos y garantizamos una alta densidad de conexiones entre los usuarios recogidos, el conjunto de datos resultante es, hasta

donde sabemos, el más completo y grande de Twitter en cuanto a datos etiquetados manualmente que clasifican a los usuarios como humanos o bots.

1.2 Objetivos

El objetivo principal de esta Tesis es desarrollar enfoques y soluciones eficientes para detectar agentes maliciosos automatizados que operan en Internet. En concreto, nos centramos en las botnets, que son redes de dispositivos infectados (bots) que se adhieren a las instrucciones de una entidad criminal; y en los bots sociales, que son cuentas falsas en redes sociales utilizadas para llevar a cabo actividades potencialmente maliciosas.

Siguiendo el objetivo principal mencionado, y los problemas y motivaciones destacados en la Sección anterior, se definieron los siguientes objetivos para proponer soluciones a esos problemas. Estos objetivos se clasifican según las dos ramas principales de esta Tesis Doctoral.

Sobre detección de malware bot:

1. Desarrollar un detector eficiente del tráfico de botnets que no dependa del descifrado de las comunicaciones y que pueda actualizarse fácilmente para detectar nuevas botnets. El objetivo de este modelo es lograr un equilibrio óptimo entre la tasa de detección y la velocidad de detección.

2. Desarrollar un modelo capaz de detectar el tráfico de botnets en tiempo real y en redes con un ancho de banda relativamente elevado (hasta 10 Gbps) cuyo uso esté dirigido a infraestructuras críticas y redes troncales. Este modelo también debe ser lo suficientemente robusto como para detectar rastros de bots incluso cuando se pierda hasta un 10 % de los paquetes debido a la saturación de la red.

3. Determinar los requisitos de hardware para el modelo anterior, en particular el número necesario de núcleos de CPU a una frecuencia de CPU determinada para lograr un rendimiento en tiempo real.

4. Crear un nuevo conjunto de datos de tráfico de red para probar nuestros sistemas de detección propuestos.

5. Proporcionar pruebas en contra del uso de IP como característica independiente para entrenar modelos de Aprendizaje Automático en la tarea de detectar tráfico de botnets.

Sobre detección de bots sociales:

1. Desarrollar un modelo capaz de detectar cuentas bots en la red social X, anteriormente conocida como Twitter.

2. Abordar el problema de clasificación de Redes Convolucionales de Grafos donde su precisión se ve afectada si el modelo intenta predecir la clase de nodos no conectados directamente, o no conectados en absoluto, a los datos de entrenamiento.

3. Crear un nuevo algoritmo para la creación de particiones en problemas de validación cruzada con datos tipo grafo, que preserve mejor las estructuras de grafo incluso en datos escasamente conectados, manteniendo la aleatoriedad en el proceso de selección de nodos.

4. Crear un nuevo conjunto de datos, recogiendo muestras directamente de la red social antes conocida como Twitter y etiquetarlas manualmente. Este conjunto de datos incluiría información sobre los datos públicos de los usuarios, las relaciones quién-sigue-a-quién entre los usuarios o la última actividad de publicación de los usuarios.

1.3 Contribuciones Principales

Esta subsección enumera y resume las principales contribuciones de esta Tesis.

1. Se crearon dos nuevos conjuntos de datos con tráfico botnet y legítimo basados en el conocido conjunto de datos CTU-13 [Garcia et al., 2014] . El primero, QB-CTU13, aborda el problema del desequilibrio de clases cuando el tráfico se divide en flujos de red. Incluye tráfico procedente del uso legítimo de ordenadores y rastros de siete botnets. El segundo, EQB-CTU13, amplía QB-CTU13 añadiendo rastros de otras tres redes de bots que se eligieron por el reto que supone su detección automática, basándose en la revisión de la literatura.

2. Se presenta un detector de tráfico de botnets eficiente basado en Decision Tree (DT), capaz de clasificar flujos de tráfico en un tiempo medio de 0,78 microsegundos cada uno, a la vez que logra un valor F1 de 0,85 en el conjunto de datos EQB-CTU13. Para idear este modelo, realizamos (1) un estudio evaluando la combinación de dos métodos de selección de características, Importancia Gini y Ganancia de Información, y (2) un estudio par optimizar parámetros y seleccionar el mejor modelos entre los generados por cuatro clasificadores de Aprendizaje Automático: DT, Random Forest, k-Nearest Neighbors y SVM.

3. Se presenta una arquitectura bimodular capaz de detectar tráfico de botnets que puede alcanzar la detección en tiempo real en grandes anchos de banda – hasta 10 Gbps. En concreto, la interpretación de tiempo real aquí significa analizar un segundo de tráfico considerando que el ancho de banda se utiliza a plena capacidad, tardando un segundo en extraer las características y otro segundo en clasificar el tráfico, suponiendo un total de dos segundos para dar un informe tras la captura de dicho segundo de tráfico.

4. Se presentan los resultados de un estudio en el que se analizan los requisitos de hardware de la propuesta de detector de tráfico de botnets en tiempo real, en anchos de banda de 100 Mbps, 1 Gbps y 10 Gbps, y utilizando CPUs comerciales que funcionan a una frecuencia de 2,4 GHz.

5. Se proporcionan pruebas experimentales que demuestran por qué es incorrecto utilizar las IP como característica independiente en el aprendizaje supervisado para la detección del tráfico de botnets.

6. Se desarrolló un detector de bots de Twitter que utiliza las relaciones "follow" de las cuentas. Dicho detector alcanza un valor F1 de 0,784 en el conjunto de datos Crescirtbust [Mazza et al., 2019] , una mejora del 24 % en comparación con el método base de la literatura.

7. Se propuso un enfoque novedoso sobre la red convolucional de grafos (GCN) de Kipf y Welling [Kipf and Welling, 2017] para resolver sus limitaciones a la hora de predecir con precisión la clase de nodo en regiones del grafo donde las etiquetas de entrenamiento son escasas o inexistentes [Zhang et al., 2019] .

8. Se desarrolló un nuevo algoritmo para separar los nodos de un grafo (o un conjunto de grafos) en particiones para la validación cruzada. Este enfoque mejora la preservación de los datos del grafo al tiempo que garantiza la aleatoriedad en la selección de los nodos. También proporcionamos una comparación de nuestro algoritmo con un caso realista para demostrar su robustez.

9. Se creó un nuevo conjunto de datos de Twitter con mensajes enviados a partir del año 2022 que incluye los metadatos públicos de los usuarios, las relaciones quién-sigue-a-quién entre el conjunto de datos – asegurando así unas relaciones de grafos relativamente densas –, y la actividad de publicación más reciente (hasta 200 publicaciones, y 100 me-gusta). Este conjunto de datos incluye 17,945 muestras etiquetadas manualmente como bots o humanos lo que lo convierte, hasta donde se sabe de acuerdo a la revisión de la literatura realizada, en el más completo, reciente y amplio en términos de datos etiquetados manualmente de todos los conjuntos de datos de Twitter publicados hasta la fecha.

10. Se desarrolló un nuevo enfoque para detectar bots sociales en Twitter basado en el comportamiento de publicación de cuentas. Este enfoque utiliza un predictor de tópicos de texto basado en BERT y un ensamblado de siete clasificadores basados en LSTM, que alcanza una precisión de 0,7550 y un F1 de 0,7767 en el nuevo conjunto de datos mencionado.

1.4 Publicaciones y resultados de investigaciones

Esta subsección presenta los resultados de investigación obtenidos durante el desarrollo de esta Tesis.

1.4.1 Publicaciones Relacionadas con esta Tesis

- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Efficient Detection of Botnet Traffic by Features Selection and Decision Trees. IEEE Access (2021). DOI: 10.1109/ACCESS.2021.3108222

- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Real-time botnet detection on large network bandwidths using machine learning. Scientific Reports (2023). DOI: 10.1038/s41598-023-31260-0

- Javier Velasco-Mata, Víctor González-Castro, Eduardo Fidalgo, Enrique Alegre. Averting Cold Start: Improved detection of Twitter bots on graph data using Graph Convolutional Networks. Será enviado a una revista científica de cuartil Q1.

1.4.2 Asistencia a congresos

- Exposición oral en congreso internacional: Javier Velasco-Mata, Eduardo Fidalgo, Víctor González-Castro, Enrique Alegre, Pablo Blanco-Medina. Botnet Detection on TCP Traffic Using Supervised Machine Learning. Hybrid Artificial Intelligent Systems (HAIS) 2019.

- Exposición oral en congreso internacional: Sergio Merayo-Alba, Eduardo Fidalgo, Víctor González-Castro, Rocío Alaiz-Rodríguez, Javier Velasco-Mata. Use of Natural Language Processing to Identify Inappropriate Content in Text. Hybrid Artificial Intelligent Systems (HAIS) 2019.

1.4.3 Premios y becas recibidas

- Beca FPU (Formación de Profesorado Universitario) del Gobierno de España con referencia FPU18/05804.

1.4.4 Participación en proyectos

- "Acuerdo de Colaboración para la puesta en marcha de un equipo de investigación aplicada en visión artificial y reconocimiento de patrones". Adenda 22 sobre el acuerdo de colaboración entre el INCIBE (Instituto Nacional de Ciberseguridad de España) y la Universidad de León.

- "Acuerdo de Colaboración para la continuidad de los trabajos de un equipo de investigación aplicada en visión artificial y aprendizaje automático". Adenda 01 sobre el acuerdo de colaboración entre el INCIBE (Instituto Nacional de Ciberseguridad de España) y la Universidad de León.

1.5 Estructura de la Tesis

Esta Tesis se estructura en siete Capítulos, un Anexo que extiende los resultados del Capítulo 5, la bibliografía, y este Apéndice final con el resumen de la Tesis en español.

- El Capítulo 1 introduce el trabajo realizado en la Tesis. Presenta la motivación que subyace a las cuatro líneas de investigación, los objetivos que se desprendieron de esta y las principales aportaciones propuestas para solucionar los diversos problemas considerados. Este capítulo también proporciona un registro de los resultados de investigación producidos durante la Tesis, así como una breve visión general de su estructura.

- En el Capítulo 2 se revisan exhaustivamente los últimos descubrimientos en los campos considerados en esta Tesis. Su primera sección recopila la revisión sistemática de la literatura relacionada y reciente. Sus siguientes secciones están dedicadas a los tres temas de investigación explorados en esta tesis: Las botnets como redes de virus informáticos, las botnets como redes de cuentas falsas en redes sociales, y los problemas y soluciones relacionados con trabajar con datos de tipo grafo.

- El Capítulo 3 presenta el trabajo enfocado en la detección eficiente del tráfico de redes de bots. Incluye la metodología que se utilizó para crear dos nuevos conjuntos de datos combinando el conjunto de datos de acceso abierto CTU-13 [Garcia et al., 2014] sobre tráfico de redes de bots con capturas de tráfico del Stratosphere IPS [Stratosphere Research Laboratory, 2023] . Además, este Capítulo describe el proceso de selección de características para optimizar la propuesta basada en Árboles de Decisión.

- El Capítulo 4 presenta el trabajo realizado sobre la detección en tiempo real de tráfico de botnets. La metodología de este Capítulo incluye la evaluación de nuestra propuesta en redes saturadas, es decir, en las que se pierde un porcentaje de paquetes que puede llegar hasta el 10 %; y la evaluación de sus requisitos de hardware.

- El Capítulo 5 está dedicado al tercer trabajo, el cual inicia el enfoque hacia los bots sociales: detectar cuentas falsas en Twitter mediante el análisis del grafo formado por las relaciones quién-sigue-a-quién de los usuarios. Aquí, se describe un novedoso método para ayudar a los modelos Red Convolucional de Grafo (GCN por su nombre en inglés) en la predicción de la clase de nodos en las regiones del grafo que están escasamente conectadas o no conectadas en absoluto con los datos de

entrenamiento, la cual es un punto débil conocido de las GCNs [Li et al., 2018b; Zhang et al., 2019] . Además, en este capítulo se presenta un novedoso algoritmo para generar particiones a partir grafos que preserva la aleatoriedad de la selección y una mejor representación de la conexiones entre nodos.

- El Capítulo 6 continúa centrándose en la detección de bots sociales, pero cambia el método de detección hacia el análisis del comportamiento de los usuarios. Aquí se presenta una propuesta para identificar bots sociales mediante un modelo basado en LSTM y una codificación de los tweets basada en BERT. Además, se describe el proceso de recopilación y etiquetado manual para la cración de un nuevo conjunto de datos de botnets de Twitter.

- En el Capítulo 7 se resume el trabajo realizado en esta Tesis y se discuten los problemas abiertos y el trabajo futuro que puede derivarse de él.

- El Anexo A incluye una exposición completa de los valores F1 alcanzados por las diferentes configuraciones del modelo propuesto en el Capítulo 5.

- El Apéndice 9 es un resumen de la Tesis Doctoral en español de acuerdo con la "Normativa para la defensa de la Tesis Doctoral en la Universidad de León" aplicada a esta Tesis.

2 Estado de la técnica

2.1 Detección de malware bots

Varios trabajos han abordado el problema de optimizar la detección de botnets desde diferentes perspectivas y considerando distintos aspectos del tráfico. Mientras que el trabajo realizado en esta Tesis sobre malware bots se centra en el tráfico generado por las comunicaciones entre bots o entre un bot y su botmaster –el dispositivo que controla los bots–, otros enfoques destacables se han centrado en la detección de peticiones DNS sospechosas [Alieyan et al., 2017; Vinayakumar et al., 2020; Zago et al., 2020] : en este caso, los bots incluyen un generador de cadenas de caracteres o Algoritmo Generador de Dominios (DGA por su nombre en inglés,) y el servidor del botmaster se oculta tras uno o varios de los dominios que se generan. En concreto, los bots intentarían conectarse con el botmaster realizando peticiones DNS con cada uno de los dominios generados, hasta que uno funcione.

Los primeros métodos para la detección de botnets como BotMiner [Gu et al., 2008a] o BotSniffer [Gu et al., 2008b] se basaban en algoritmos estadísticos para detectar el tráfico de botnets. Esto fue posible aprovechando las correlaciones espacio-temporales del tráfico generado por los programas maliciosos, a diferencia del tráfico producido por humanos reales, que es más aleatorio.

Revisando las últimas publicaciones sobre detección de botnets al realizar esta Tesis, se puede observar que los algoritmos de Aprendizaje Automático son los más comunes cuando el objetivo es construir clasificadores de tráfico multiclase. Entre todos ellos, los modelos basados en Árboles de Decisión (DT por su nombre en inglés) suelen obtener los mejores resultados. Por ejemplo, en el trabajo de Gadelrab et al. [Gadelrab et al., 2018] , se compararon Máquinas de Vectores de Soporte (SVM por su nombre en inglés) y DT para seleccionar el mejor clasificador para la implementación de un detector de botnets, llamado BoTCap. DT logró los mejores resultados (es decir, un valor F1 de 0,95) en un conjunto de datos creado por los autores con rastros de seis redes de bots: Aryan, Ngr, Rxbot, Blackenergy, Zeus y Vertexnet. Además, los algoritmos basados en DT también mostraron un alto rendimiento en conjuntos de datos de ataques de red, como el conjunto de datos CICIDS2017 [Panigrahi and Borah, 2018] , que recopila ataques en línea como DDoS o infecciones de malware. Este conjunto de datos también se utilizó en el trabajo de MacKay et al. [McKay et al., 2019] para construir tres conjuntos de entrenamiento y dos conjuntos de prueba, empleados para comparar seis algoritmos: DT, Bosque Aleatorio (RF por su nombre en inglés), k-Vecinos Cercanos (k-NN por su nombre en inglés), clasificador Bayesiano Ingenuo (NB por su nombre en inglés), Perceptrón Multicapa (MLP por su nombre en inglés) y Una Regla (OneR por su nombre en inglés). Los resultados mostraron que DT obtuvo la mejor puntuación F1 (es decir, 0,99) utilizando las 78 características del conjunto de datos CICIDS2017, y los autores dejaron la selección de características como trabajo futuro.

Por último, trabajos más recientes han implementado detectores de botnets utilizando un conjunto hiperreducido de características. Joshi et al. [Joshi et al., 2020] utilizaron cinco técnicas de selección de características, produciendo cada una un subconjunto de características a partir del conjunto original. El subconjunto más pequeño se obtuvo utilizando PCA, y sólo tenía dos características: la dirección IP de origen y el protocolo. Los autores probaron los subconjuntos de características con cuatro clasificadores: SVM, LR, k-NN y DT. Utilizando el conjunto de datos CTU-13, k-NN obtuvo los mejores resultados: 0,99 de precisión utilizando únicamente el mencionado subconjunto de solo dos características. Un año después, Ismail et al. [Ismail et al., 2021] realizaron una investigación similar utilizando datos más recientes. Utilizaron siete capturas de tráfico del Stratosphere IPS [Stratosphere Research Laboratory, 2023] , y también probaron más clasificadores, en concreto Red Bayesiana, NB, k-NN, Ada Boost, SVM, J48, RF, Ripper y PART. Llegaron a la conclusión de que SVM obtuvo la mejor tasa de verdaderos positivos en la detección de datos de botnets en las siete capturas de tráfico, utilizando un subconjunto de sólo tres características: las IP de destino y origen y el protocolo. A pesar de que estos trabajos proporcionan información sobre conjuntos de características reducidos, no analizaron el coste computacional de sus propuestas.

En comunicaciones IoT, Borges et al. [Borges et al., 2022] desarrollaron un detector de anomalías que solo requería medir el número de paquetes enviados por un dispositivo en ventanas de tiempo de tan solo 0,1 s. Más en detalle, la concatenación de estas medi-

das da lugar a una serie temporal que primero es transformada en una serie de *patrones simbólicos* mediante una *transformación ordinal de patrones*, que luego es analizada por el detector en busca de patrones anómalos. Según sus resultados en el conjunto de datos N-BaIoT, alcanzaron precisiones entre 0,985 y 1,000 cuando utilizaron series temporales de longitud $m \geq 400$ medidas. Como limitación, informaron de que la construcción y el procesamiento de las series temporales requerían al menos un minuto antes de realizar la detección. Después, el tiempo de detección era de 24 ms en el mejor de los casos y de 40 ms en el peor.

También es posible desarrollar modelos que funcionen sobre tráfico genérico, es decir, que no estén limitados a que las comunicaciones usen un protocolo particular o se den entre dispositivos especializados [Gaonkar et al., 2020] como BotMiner [Gu et al., 2008a] . Una estrategia frecuente para optimizar estos detectores es la selección de las mejores características: esto reduce las características a extraer de los datos y también la complejidad del clasificador. En [Chen et al., 2017b] las características se seleccionan utilizando la métrica Gini Importance, y la clasificación se realiza con un clasificador Random Forest (RF). Sobre el conocido conjunto de datos CTU-13 [Garcia et al., 2014] , este enfoque alcanzó una precisión de 0,90 y el RF fue capaz de clasificar 204,711 muestras en 27,1 segundos.

2.2 Detección de bots sociales

Durante la revisión de la literatura sobre bot sociales realizada durante la Tesis, se observó una dificultad cada vez mayor para detectar las redes de bots de Twitter a lo largo de los años, lo que sugiere una evolución en la sofisticación de estos bots. Por ejemplo, el modelo de Pham et al. [Pham et al., 2022] Bot2Vec, que combina el algoritmo word2vec para crear representaciones numéricas a partir de los datos de usuarios de Twitter, y una máquina de vectores de soporte (SVM) para utilizar dichas representaciones, obtuvo un valor F1 de 0,9840 cuando se aplicó a datos de 2015. Schnebly y Sengupta [Schnebly and Sengupta, 2019] obtuvieron una precisión de 0,9025 con un RF en una mezcla de datos de 2015 y 2017. Los datos de 2015 eran mayores (1946 humanos, 3457 bots) que los de 2017 (445 humanos y 3202 bots). Un trabajo posterior de Fonseca et al. [Abreu et al., 2020] solo utilizó datos de 2017, e informó de una precisión media de 0,8549 entre tres clasificadores, RF, SVM y NB.

Trabajos posteriores proponen enfoques con un análisis más profundo de los tweets. Attia et al. [Attia et al., 2022] propusieron un enfoque de doble entrada para detectar bots de Twitter a partir de sus publicaciones. La primera entrada es la representación en N-gramas del contenido de los tweets, y la segunda se basa únicamente en el recuento de palabras en los tweets. Cada entrada se alimenta a un modelo CNN, y la salida de ambos se concatena para alimentar una capa de salida final. Probaron su modelo en el conjunto de datos CLEF 2019 [Kosmajac and Keselj, 2020] logrando una precisión de 0,9325, suponiendo un incremento de 3,22 % sobre el modelo base con el que compararon. Ade-

más, Feng et al. [Feng et al., 2021] combinaron un análisis profundo de los textos de los tweets utilizando BERT con la información de los datos gráficos de la red de Twitter utilizando un GCN de 2 capas obteniendo un valor F1 de 0,87 sobre el conjunto de datos que recopilaron en 2020.

Trabajos recientes publicados en paralelo y después de la experimentación descrita en el capítulo 6 continúan la tendencia de utilizar el contenido de las publicaciones para identificar bots sociales en Twitter. En su estudio, Chawla y Kapoor [Chawla and Kapoor, 2023] proponen un modelo que combina codificaciones de ADN con representaciones de los tweets, hechas con BERT preentrenado, para distinguir entre spambots y cuentas auténticas. Alcanzaron una precisión de 0,7523 en el conjunto de datos DynaSent. Además, Arin y Kutlu [Arin and Kutlu, 2023] propusieron una arquitectura que combinaba representaciones GloVe preentrenadas junto a modelos LSTM para utilizar los tweets y la descripción de la cuenta de Twitter, junto con los metadatos del usuario, para predecir si la cuenta es un bot social. Evaluaron su propuesta en cinco conjuntos de datos alojados en Botometer publicados entre 2011 y 2019. La puntuación F1 alcanzada fue de 0,80, con una precisión de 0,83.

3 Detección de botnets eficiente

Las botnets, redes de dispositivos infectados y controlados remotamente, son una de las mayores amenazas online en la actualidad, capaces de realizar ataques como denegaciones de servicio distribuidas (DDoS por su nombre en inglés) o campañas de spam [Zhao et al., 2013] . Además, el continuo aumento del número de dispositivos conectados a internet hace que la superficie de ataque para infectar equipos para una botnet sea mayor [Koroniotis et al., 2019; Li et al., 2019] .

Por ello, este trabajo se ha enfocado en mejorar el rendimiento de la clasificación de tráfico para detectar actividad botnet. Trabajos recientes han confirmado la ventaja de usar Árboles de Decisión (DT por su nombre en inglés) así como k-Vecinos Cercanos (k-NN por su nombre en inglés) [Stevanovic and Pedersen, 2014; Khan et al., 2019; Lee et al., 2021; Alshamkhany et al., 2020; Jiménez and Goseva-Popstojanova, 2018] . Por ello, en este trabajo se han seleccionado los modelos DT, Bosque Aleatorio (RF por su nombre en inglés) y k-NN como candidatos para obtener un modelo óptimo de detección de botnets.

Este trabajo se ha enfocado en tráfico que usa el protocolo TCP dado que es el utilizado por la mayoría de las botnets conocidas [Wainwright and Kettani, 2019] . En un anterior trabajo [Velasco-Mata et al., 2019] , se utilizaron trece características para describir los flujos TCP, pero en este trabajo se descartaron dos de ellos: la media y la varianza de la velocidad de transmisión de los paquetes (mVel y vVel). Su eliminación es debida a que la información que aportaban ya estaba representada por la media y la varianza del intervalo de tiempo entre paquetes consecutivos (mTime y vTime), como se muestra en la ecuación (1). La tabla 1 muestra las once características restantes que seleccionamos

Tabla 1: Características del tráfico de red usadas en [Velasco-Mata et al., 2019]

Característica(s)	Descripción
sPort, dPort	Puertos fuente y destino en la conexión TCP
mLen, vLen	Media y varianza del tamaño del payload de todos los paquetes intercambiados en la conexión TCP
mTime, vTime	Media y varianza de los intervalos de tiempo entre paquetes consecutivos enviados durante la conexión TCP
mResp, vResp	Media y varianza de los intervalos de tiempo que el dispositivo que inició el flujo TCP tarda en responder a la llegada de un paquete
nBytes	Número de bytes total intercambiado en la conexión TCP
nSYN	Número total de SYN flags intercambiadas durante la conexión TCP
nPackets	Número total de paquetes intercambiados en la conexión TCP

para modelar el tráfico de red.

$$\text{mVel} = \frac{\text{numero de paquetes en un flujo TCP}}{\text{duración del flujo TCP}} = \frac{1}{\text{mTime}} \tag{1}$$

Para llevar a cabo la experimentación, se crearon dos nuevos conjuntos de datos a partir del conocido CTU-13 [Garcia et al., 2014] . El problema que presentaba este dataset cuando se extraen los flujos TCP para utilizarlos como muestras individuales es que hay una gran desproporción entre la cantidad de flujos de unas clases de bots y otras. Por ello el primero de los nuevos subconjuntos, que hemos llamado QB-CTU13, o CTU13 cuasi-balanceado, recoge todas las muestras de las clases minoritarias y solo balancea mediante descarte de muestras a las tres clases más representadas.

El otro de los dos nuevos conjuntos de datos es EQB-CTU13, o QB-CTU13 extendido, que incluye las muestras de QB-CTU13 y añade flujos de tres botnets extra: Bunitu, Miuref y NotPetya, que han sido declarados como difíciles de detectar en la literatura [Abraham et al., 2018; AlAhmadi and Martinovic, 2018] . El número de flujos presente en cada dataset se recoge en la Tabla 2.

El otro aspecto considerado para optimizar los detectores fue la selección de características, la cual se llevó a cabo en dos fases. En la primera fase, realizada sobre QB-CTU13, se determina la importancia de cada característica combinando dos métodos de selección: Importancia de Gini [Nembrini et al., 2018] y Ganancia de Información [Wen et al., 2016] . Ambos métodos ordenan las características según su importancia relativa entre ellas. La Importancia de Gini de acuerdo a cómo de útil es una característica para dividir las muestras en grupos diferenciados, y la Ganancia de Información, como su nombre indica, según la información que aporte cada característica. Combinando estos dos ran-

Tabla 2: Flujos TCP usados como muestras en los conjuntos de datos QB-CTU13 y EQB-CTU13 Datasets.

Clase	QB-CTU13	EQB-CTU13
Normal	3890	3890
Neris	3890	3890
Rbot	3890	3890
Virut	3890	3890
Murlo	2036	2036
NSIS	355	355
Donbot	233	233
Sogou	36	36
Bunitu	-	3890
Miuref	-	3890
NotPetya	-	111

kings, se propusieron tres subconjuntos de características:

1. Subconjunto con cinco características: [dPort, nPackets, nBytes, vLen, mLen]
2. Subconjunto con seis características: [dPort, nPackets, nBytes, vLen, mLen, mTime]
3. Subconjunto con siete características: [dPort, nPackets, nBytes, vLen, mLen, mTime, vTime]

En la segunda fase, realizada sobre EQB-CTU13, se compararon las combinaciones entre los tres subconjuntos de características y los tres modelos optimizados para maximizar su velocidad: DT, RF ($m = 10$) y k-NN ($k = 1$). Se usaron tres medidas para evaluar estas combinaciones: (1) el valor F1, (2) el tiempo computacional medio que tarda un modelo ya entrenado en clasificar una muestra y (3) la división del primero por el segundo, para medir el rendimiento del modelo, que será mejor cuanto mayor sea su F1 y menos tarde en clasificar muestras. Estas se realizaron mediante una validación cruzada de 10 particiones.

La Figura 1 representa los valores F1 medios, y muestra una clara ventaja de DT y RF sobre k-NN. Además, k-NN también mostró un mayor coste computacional para la clasificación que los otros modelos (véase la Figura 2). Por otro lado, aunque RF obtuvo valores F1 más altos que DT (véase la Figura 1), el coste de utilizar más de un DT supuso una reducción significativa del rendimiento, como se muestra en la Figura 3. El modelo que obtuvo el mejor rendimiento fue el DT que utilizaba el subconjunto de 5 características, mientras que el DT que utilizaba un subconjunto de 6 características obtuvo un resultado similar. Sin embargo, el uso de una característica adicional también implica un coste

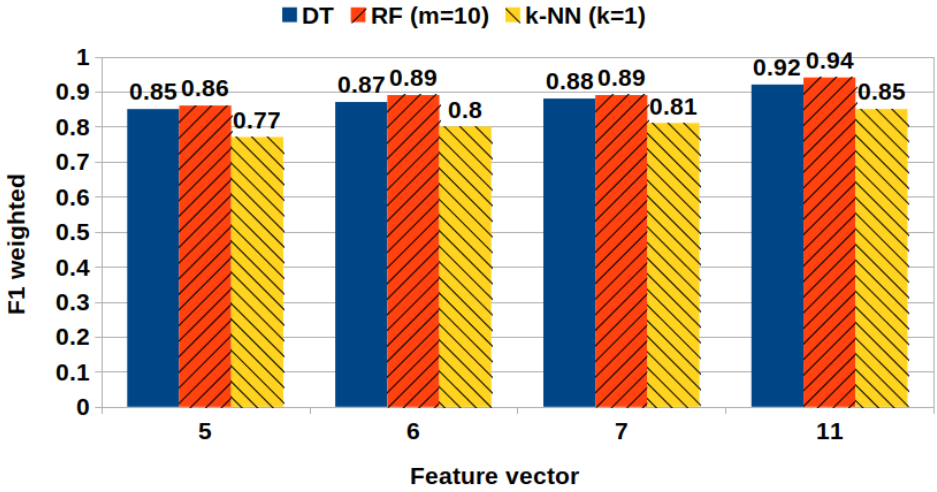

Figura 1: Valores F1 obtenidos por DT, RF ($m = 10$) y k-NN ($k = 1$) usando los conjuntos de 5, 6 y 7 características, así como el conjunto completo de 11 características. El valor F1 se ha calculado como la media ponderada entre las clases de tráfico en EQB-CTU13.

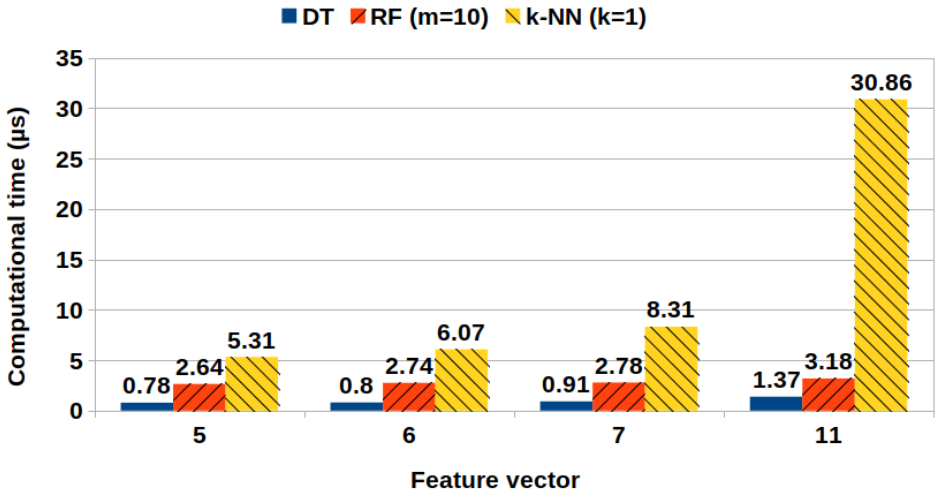

Figura 2: Tiempo computacional medio para clasificar una muestra necesitado por DT, RF ($m = 10$) y k-NN ($k = 1$) usando los conjuntos de 5, 6 y 7 características, y el conjunto completo de 11 características sobre el conjunto de datos EQB-CTU13.

computacional adicional para calcularla. Así pues, desde el punto de vista del rendimiento, el DT con el subconjunto de 5 características es más recomendable.

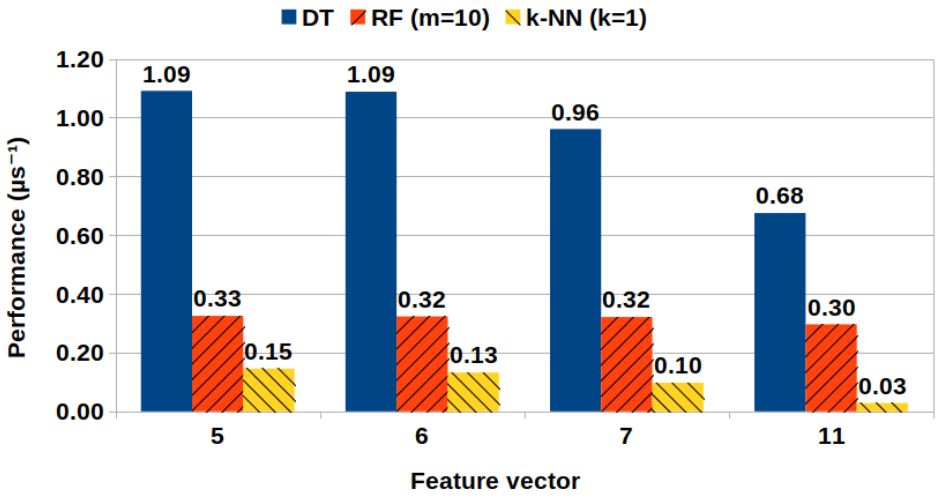

Figura 3: Rendimiento conseguido por DT, RF ($m = 10$) y k-NN ($k = 1$) usando los conjuntos de 5, 6 y 7 características, y el conjunto completo de 11 características sobre el conjunto de datos EQB-CTU13.

En la Tabla 3 se muestran los valores de F1, recall, precisión y rendimiento por cada clase del conjunto de datos EQB-CTU13 obtenidos en una validación cruzada de 10 iteraciones con dos versiones del DT: utilizando el subconjunto de 5 características optimizado para el rendimiento, y utilizando las once características. El rendimiento se definió como la relación entre el valor F1 y el tiempo medio necesario para clasificar una muestra. También se comparó la propuesta de este trabajo con las propuestas de Joshi et al. [Joshi et al., 2020] , es decir, una k-NN que utiliza dos características (knn-2f), la IP origen y el protocolo usado; e Ismail et al. [Ismail et al., 2021] , una SVM que utiliza tres características (svm-3f), el par de IPs y el protocolo usado. Según la experimentación de estos dos trabajos, k-NN y SVM obtuvieron mejores resultados que DT utilizando sus propuesta de conjuntos de dos y tres características, respectivamente.

Con respecto a a los dos enfoques basados en DT (dt-5fs y dt-11fs), se observa que en tres clases de botnet, Donbot, Murlo y NotPetya, el valor F1 logrado con cinco características es mejor que el obtenida utilizando las once características. La puntuación recall obtenida para NotPetya también es mayor cuando se utiliza el subconjunto de 5 características. Aunque el vector de once características logra mejores resultados de detección, la métrica de rendimiento es significativamente mayor con el vector de cinco características, excepto para la clase Sogou, lo que se explica por la escasa representación de esta clase (es decir, sólo contiene 36 flujos TCP). Esto también provocó que la botnet Sogou obtuviera los valores más bajos de puntuación F1 y recall. Otros trabajos [Le et al., 2019] obtuvieron una mayor puntuación para esta botnet en el conjunto de datos CTU-13 ya

Tabla 3: Métricas F1, Recall, Precision y Rendimiento de la propuesta, un DT con el conjunto de 5 características (dt-5fs). Para comparar, se incluyen las métricas de un DT que usa el conjunto completo de características (dt-11fs), la SVM propuesta por Joshi et al. [Joshi et al., 2020] que usa tres características (svm-3fs), y la propuesta de Ismail et al. basada en k-NN [Ismail et al., 2021] que usa dos características (knn-2fs).

	F1				Recall				Precision				Rend. (ms^{-1})			
Clase	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f	dt-5fs	dt-11fs	svm-3f	knn-2f
Normal	0.78	0.93	0.99	0.99	0.83	0.93	0.99	0.99	0.74	0.93	1.00	1.00	**1007.49**	68.32	2.16	48.51
Bunitu	0.84	0.94	0.01	0.99	0.87	0.94	0.07	0.99	0.81	0.94	0.01	0.99	**1079.65**	682.32	0.03	48.46
Donbot	0.88	0.85	0.64	0.98	0.85	0.87	0.52	0.98	0.91	0.83	0.84	0.99	**1139.20**	620.69	1.38	47.97
Miuref	0.99	1.00	0.37	1.00	0.99	1.00	0.23	1.00	0.99	1.00	1.00	1.00	**1278.51**	728.21	0.81	48.76
Murlo	0.98	0.97	0.00	1.00	0.97	0.97	0.00	1.00	0.99	0.97	0.00	1.00	**1259.58**	709.21	0.00	48.70
NSIS	0.59	0.85	0.00	0.01	0.59	0.86	0.00	0.11	0.59	0.84	0.00	0.10	**761.88**	623.66	0.00	0.29
Neris	0.69	0.84	0.00	1.00	0.71	0.83	0.00	1.00	0.67	0.85	0.00	1.00	**891.02**	610.92	0.00	48.63
NotPet.	0.96	0.95	0.00	0.99	0.97	0.95	0.00	1.00	0.95	0.94	0.00	0.98	**1241.69**	693.49	0.00	48.28
Rbot	0.98	0.99	0.50	0.96	0.98	0.99	0.58	0.92	0.98	0.99	0.45	0.99	**1262.78**	722.94	1.09	46.62
Sogou	0.22	0.55	0.00	1.00	0.38	0.48	0.00	1.00	0.15	0.64	0.00	1.00	285.76	**400.61**	0.00	48.78
Virut	0.75	0.84	0.32	1.00	0.78	0.85	0.56	1.00	0.72	0.83	0.22	1.00	**967.83**	614.89	0.69	47.71

que trabajaron con cada escenario por separado.

Como se indicó anteriormente, para crear el conjunto de datos EQB-CTU13 a partir del QB-CTU13, se añadieron muestras de tráfico de las botnets Bunitu, NotPetya y Miuref porque la revisión bibliográfica indicó que son botnets relativamente difíciles de detectar. En Abraham et al. [Abraham et al., 2018] , el mejor valor F1 alcanzado para la botnet Bunitu fue de 0,90, mientras que con el enfoque propuesto utilizando once características, se obtiene un F1 de 0,94 mientras que con el subconjunto más eficiente de cinco características, el valor F1 alcanzado fue de 0,84.

En cuanto a las redes de bots NotPetya y Miuref, AlAhmadi y Martinovic [AlAhmadi and Martinovic, 2018] obtuvieron un recall para NotPetya de 0,60, donde el 25 % de las muestras de NotPetya se clasificaron erróneamente como muestras Miuref. En nuestra propuesta, el DT optimizado (es decir, dt-5fs) obtuvo un recall de 0,97 para la clase NotPetya, y las puntuaciones F1 y recall de la clase Miuref fueron ambas de 0,99.

Por último, por un lado se observa que el k-NN que utiliza dos características obtuvo los mejores resultados en todas las clases. Estas dos características eran la IP de origen y el protocolo utilizado. Por otro lado, la SVM también utilizó estas dos características y añadió la IP de destino, y obtuvo peores resultados. Se observa que la SVM que utiliza tres características consigue un buen rendimiento distinguiendo entre tráfico normal y de botnets, pero sufre en la tarea de diferenciar botnets. Sin embargo, el rendimiento de estos dos clasificadores fue del orden de 10^{-2} inferior al rendimiento del DT, lo que indica que incluso utilizando más características y considerando también el tiempo extra para calcularlas, el DT es significativamente más rápido que SVM y k-NN.

4 Detección de botnets en tiempo real

Como ya se indicó en el anterior trabajo, las botnets son una de los principales preocupaciones en cuanto a amenazas online por su capacidad de escalar ataques online, como DDoS [Shafi and Basit, 2019] , campañas de spam [Besel et al., 2018] , espionaje [Bederna

and Szadeczky, 2019] , phishing [Martínez Torres et al., 2019] y ransomware [Wainwright and Kettani, 2019] . Las pérdidas económicas debidas al cibercrimen aumentan cada año, estimándose en 6,9 miles de millones de dólares americanos en el año que se hizo esta investigación [FBI, 2022] ,

En la Industria 4,0 [Moustafa et al., 2018] , es necesario llevar a cabo una rápida detección de potenciales amenazas, idealmente en tiempo real. Recientes estudios sobre redes definidas por software (SDN por su nombre en inglés) midieron los requisitos de hardware para detectar botnets [Maimó et al., 2017; Gil Pérez et al., 2017; Maimó et al., 2018] , pero se limitaron a las redes móviles 5G. En este trabajo se adopta una visión más amplia acotando los requisitos de hardware necesarios para detectar botnets en tiempo real sobre los protocolos estándar TCP y UDP en cualquier tipo de redes de tráfico que los utilice, con anchos de banda de hasta 10 Gbps.

En este trabajo el modelo propuesto trabaja clasificando flujos de paquetes, pero usando una definición más genérica a los flujos TCP del trabajo anterior. En particular, se define un flujo como la cadena de paquetes intercambiada por dos máquinas durante el intervalo de tiempo de un segundo. Los paquetes que pasan por el ancho de banda de la red en dicho intervalo se identifican como pertenecientes a un flujo u otro a través de las direcciones IP y de los puertos usados. Esta definición permite usar también tráfico que use el protocolo UDP.

El esquema propuesto se compone de dos módulos: uno que, por cada segundo de tráfico, separa los flujos y extrae las características de estos; y un modelo que clasifica dichos flujos. En particular, se han usado cuatro características de las once del anterior trabajo, seleccionadas por su facilidad de cálculo para hacer más rápido el proceso:

- El puerto fuente.
- El puerto destino.
- El número de paquetes del flujo.
- El número total de bytes transmitidos en el flujo.

El modelo seleccionado para clasificar tráfico fue DT siguiendo los resultados del trabajo anterior, que mostraron que era el más rápido de los considerados. El conjunto de datos usado fue el EQB-CTU13 también presentado en el anterior trabajo.

Para asegurar la robustez del modelo propuesto, se estudió como varía su rendimiento en redes saturadas, es decir, aquellas donde la carga sobre el ancho de banda es tal que se empiezan a perder paquetes. Puede considerarse que una red sigue siendo operativa mientras la perdida de paquetes no supere un 10 % [Kurose and Ross, 2010; Vyopta, 2018] . Por ello, en la Figura 4 se compara el rendimiento del modelo –siempre entrenado con tráfico no saturado– al clasificar tráfico donde no se han perdido paquetes (situación ideal) frente a tráfico donde se ha simulado una pérdida aleatoria del 10 % de los paquetes

(saturación). Estos resultados son los valores medios de una validación cruzada estratificada de 10 iteraciones. Puede verse que los valores F1 no sufrieron significativamente por las pérdidas de paquetes. En cuanto al bajo valor F1 en la botnet Donbot, aproximadamente el 96 % del tráfico de Donbot y el 30 % del tráfico de Neris se dirigían al puerto 25, lo que causó que el clasificador identificara erróneamente la mayor parte del tráfico de Donbot como tráfico de Neris.

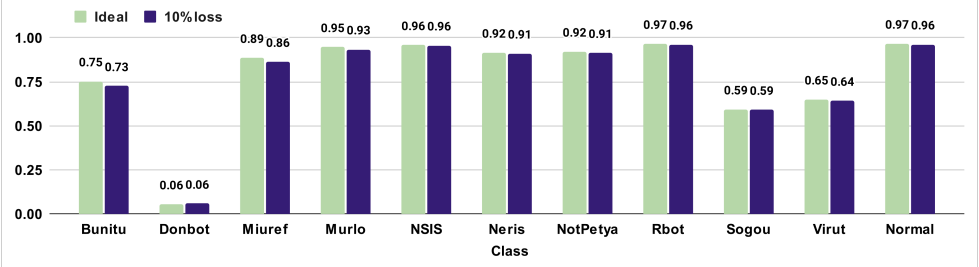

Figura 4: Valores F1 por cada clase de tráfico en condiciones ideales y en condiciones saturadas donde el 10 % de paquetes se pierde.

A continuación, se evaluaron los requisitos de hardware de la propuesta para distintos anchos de banda. Se simularon tres anchos de banda para estimar el coste computacional de un DT: 100 Mbps, 1 Gbps y 10 Gbps. Esta simulación se realizó juntando flujos reales generados en diferentes momentos en la misma ventana de tiempo, de forma que el sistema recibe para analizar 100 Mb, 1 Gb o 10 Gb de datos cada segundo respectivamente. El módulo se construyó utilizando Python 3 y se ejecutó en un Intel(R) Xeon(R) E5-2630v3 (CPU de 2,4 GHz, 3,5 GHz en modo Turbo).

La Figura 5 muestra el tiempo máximo, medio y mínimo requerido por el modelo para calcular las características, clasificar las muestras, y todo el proceso completo en los tres anchos de banda. Puede observarse que calcular características tomó menos tiempo que solo la clasificación, gracias al conjunto de características seleccionado. Tanto el ancho de banda de 100 Mbps como el de 1 Gbps requieren un máximo de aproximadamente 3,8 segundos por cada segundo de tráfico. Esto significa que si el proceso se paralelizara, se necesitarían al menos 4 núcleos CPU de la misma frecuencia (o equivalente) para tardar un solo segundo en clasificar un segundo de tráfico.

En el caso de un ancho de banda de 10 Gbps, el tiempo máximo necesario para clasificar un segundo de tráfico fue de 15,9 s, pero teniendo en cuenta que en el peor de los casos el cálculo de características necesitó 8,0 s y la clasificación 10,3 s, la opción más segura con paralelización requeriría al menos 19 núcleos CPU a la misma frecuencia o equivalente.

Finalmente, se comparó el modelo propuesto con el presentado por Gahelot y Dayal [Gahelot and Dayal, 2020] , también basado en DT, pero que usa un conjunto de características más extenso:

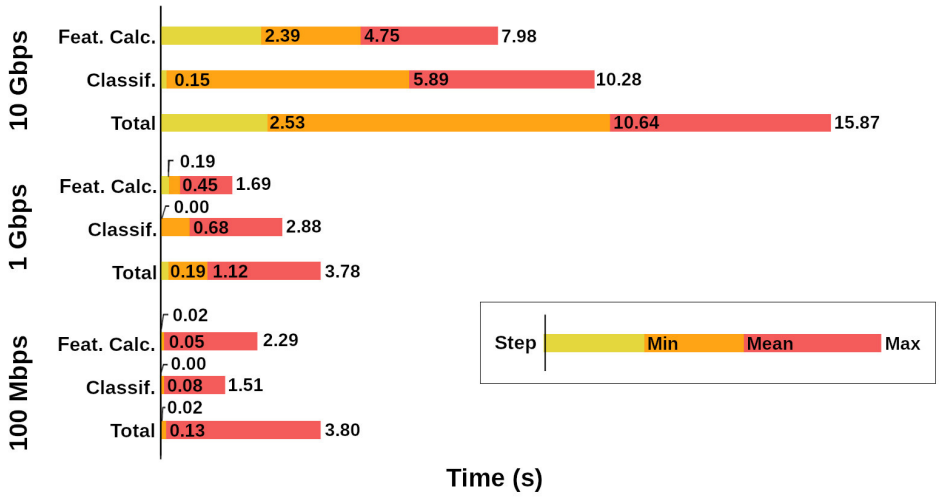

Figura 5: Tiempo máximo, medio y mínimo (s) requeridos para calcular las características, clasificar muestras, y todo el procedimiento junto si el sistema usa ventanas de tiempo de un segundo en anchos de banda de 100 Mbps, 1 Gbps y 10 Gbps.

1. IP origen.
2. IP destino.
3. Puerto destino.
4. Tamaño medio de los paquetes.
5. Número de paquetes sin payload.
6. Tamaño del primer paquete.
7. Protocolo.
8. Duración.
9. Total de bytes transferidos.
10. Número total de paquetes transferidos.

Como puede observarse, la propuesta de Gahelot and Dayal usa como características las IPs de las comunicaciones. Sin embargo, se consideró que esto puede ser contraproducente, dado que si bien las IPs pueden usarse con el propósito de agregar paquetes para calcular otras características, no deberían usarse como característica por sí mismas dado que (1) no fueron asignadas por el programa bot malicioso, (2) pueden ser enmascaradas

mediante NATs o tecnologías similares y (3) pueden ser usadas para identificar la máquina infectada en el entrenamiento de la misma manera a como si se usara la etiqueta como característica. Esto significa que el algoritmo podría aprender que "dispositivos con la IP A estarán infectados, y dispositivos con la IP B no", de manera que mostrarían un buen rendimiento en el mismo conjunto de datos, pero un rendimiento pobre en otros conjuntos o en un escenario real. Para probar esto, se evaluó también una versión modificada del modelo de Gahelot y Dayal que simplemente no usa las IPs como características. Además, en este experimento se dividió EQB-CTU13 en un conjunto de entrenamiento con el 70 % de las muestras, preservando el porcentaje de muestras de cada clase, y en un conjunto de test con el resto de muestras a la vez que se aseguraba que las IPs de las muestras del conjunto de test no estuviesen presentes en el conjunto de entrenamiento. Los resultados pueden verse en la Tabla 4

Tabla 4: Valores F1 ponderados y tiempo medio para procesar una muestra. El tráfico se dividió en flujos de un segundo. El modelo *modificado* de Gahelot y Dayal se refiere a aquel en el que se excluyeron las IPs como característica. El modelo DT optimizado se refiere al propuesto en este trabajo, de cuatro características.

Modelo	**F1**	**Tiempo medio/muestra (ms)**
Gahelot y Dayal [Gahelot and Dayal, 2020] (Original)	0.095	0.048
Gahelot y Dayal [Gahelot and Dayal, 2020] (Modificado)	0.911	0.046
DT optimizado	**0.926**	**0.007**

Como era de esperar, la versión original de Gahelot y Dayal [Gahelot and Dayal, 2020] (es decir, la que contenía las IPs como características) obtuvo un mal resultado, con un valor F1 de 0,095, mientras que la versión modificada (es decir, sin las IPs) no sólo obtuvo una puntuación F1 mucho mayor (0,911), sino que también fue ligeramente más rápida. La mejor puntuación F1 y el mejor tiempo la obtuvo la propuesta de este trabajo, el DT que usa cuatro características, siendo capaz de analizar cada flujo en una media de 0,007 ms.

5 Detección de falsos usuarios en redes sociales usando datos de tipo grafo

Twitter (actualmente conocido como X) es una de las plataformas más populares donde la gente comparte sus pensamientos sobre cualquier tema. Con 206 millones de usuarios activos diarios, es considerada por muchos la forma más rápida de informarse sobre los acontecimientos mundiales [Dean, 2022] . Sin embargo, su popularidad atrae a spammers, estafadores y botnets [Rodríguez-Ruiz et al., 2020] . Este trabajo se ha centrado en la detección de cuentas automatizadas, también conocidas como bots sociales, y en particular en Twitter. Si bien es cierto que existen cuentas automatizadas legítimas, como cuentas que reportan el tiempo atmosférico, los conjuntos de datos utilizados que proce-

den de Botometer – un proyecto centrado en la detección de bots en Twitter – no hacen dicha distinción. Además, estas cuentas legítimas siempre pueden ser añadidas a listas blancas por empleados de Twitter.

Se utilizaron un total de ocho conjuntos de datos procedentes de Botometer, y para incluir los datos sobre quién-sigue-a-quién, es decir, datos de grafo, se usó una cuenta de desarrollador y la API de Twitter. Este información se extrajo entre diciembre de 2020 y enero de 2021. La Tabla 5 muestra la cantidad de cuentas bot y humanas de cada conjunto de datos además de si dichas cuentas tenían conexiones con otras del datase.

Tabla 5: Información sobre las cuentas aisladas y conectadas (tanto humanas como bots) en cada dataset utilizado.

Conjunto de datos	**Humanos aislados**	**Humanos conectados**	**Bots aislados**	**Bots conectados**
Botwiki	0	0	234	464
Verified	1042	945	0	0
Pronbots	0	0	16317	1565
Botometer-feedback	284	95	122	17
Cresci-rtbust	204	136	101	252
Political-bots	0	0	49	13
Vendor-purchased	0	0	676	411
Celebrity	799	5119	0	0

Como modelo base del que partió la experimentación, se utilizó el modelo de Redes Convolucionales de Grafo (GCN por su nombre en inglés), propuesta por T. N. Kipf and M. Welling. [Kipf and Welling, 2017] y utilizada por Alhosseini et al. en el problema de detección de botnets en Twitter [Ali Alhosseini et al., 2019] . Sin embargo, este modelo presenta una desventaja: Dado que está diseñado como "propagador de etiquetas", es decir, que predice la clase de un nodo (del conjunto de test) según la clase conocida de nodos cercanos – del conjunto de entrenamiento – (de ahí el nombre de convolución de grafo), si el modelo trata de predecir la clase de nodos que no están apenas conectados con nodos de entrenamiento, o no conectados en absoluto con estos, su precisión cae significativamente [Li et al., 2018b; Zhang et al., 2019] .

La contribución que se propuso para solucionarlo es una estructura de pipeline (tubería en inglés) entre dos modelos, un modelo de apoyo que no use datos de grafo, es decir, que solo prediga el tipo de usuario según sus características sin atender a quién sigue en el conjunto de datos; y el modelo GCN que usa ambos tipos de datos. Este esquema es genérico, es decir, puede usarse en otros problemas de clasificación de nodos distintos a la detección de bots en Twitter.

Inicialmente, ambos modelos se entrenan sobre el conjunto de entrenamiento. Después, en la fase de inferencia, actúa primero el modelo de apoyo para predecir la probabilidad de que los nodos del conjunto de test sean de una clase u otra. Si la probabilidad predicha supera un cierto umbral prefijado t, entonces se asigna esa etiqueta al nodo, y si no se deja sin etiquetar.

Posteriormente, el modelo GCN se re-entrena con las predicciones del primer mode-

lo (no con las etiquetas reales) y se usa para predecir el resto de nodos que no fueron etiquetados por el primer modelo.

Para realizar la experimentación, se usó el framework GraphSAGE (SAmple and aggreGatE) desarrollado por Hamilton et al. [Hamilton et al., 2017] , que permite utilizar GCNs. Además, este proyecto permite usar el GCN original, y GCNs como generador de embeedings o representaciones numéricas de los datos y que pueden usarse en otro modelo como características para predecir la clase del nodo. Por ello se usaron tres versiones de la red neuronal: el GCN original denotado como tal, y el GCN como generador de embeddings y utilizado por un Bosque Aleatorio (RF por su nombre en inglés) y denotado por GCN+RF, o utilizado por una Máquina de Vectores de Soporte (SVM pro su nombre en inglés Naïve Bayes) y denotado por GCN+SVM. Así pues, se testearon nueve modelos:

- Tres modelos de Aprendizaje automático: RF, SVM y clasificador Bayesiano Ingenuo (NB por su nombre en inglés).
- Las tres versiones de GCNs: GCN, GCN+RF, GCN+SVM.
- Tres pipelines obtenidos de usar un RF como modelo de apoyo, y cada una de las tres versiones de GCNs: RF-GCN, RF-GCN+RF, RF-GCN+SVM.

Las 33 características que se consideraron para describir a los usuarios se recogen en la Tabla 6. En particular, se consideró el conjunto de seis características utilizado por Alhosseini et al. [Ali Alhosseini et al., 2019] al se llamó “detectme” por el título de su manuscrito, y el conjunto de 23 características de Kouvela et al. [Kouvela et al., 2020] al que se llamó “botdetective” por la misma razón. Por último, también se probó un conjunto de 33 características denominado “complete” que se obtuvo combinando los dos conjuntos anteriores, y nueve características adicionales, producto de calcular ratios a partir de otras características, y contando los emojis utilizados en los campos que permiten su uso. La intuición para considerar los emojis provino de la observación directa de los datos.

De los ocho conjuntos de datos de la Tabla 5, se consideró a Cresci-rtbust como la que supone el mayor reto, dado que contiene muestras tanto de cuentas humanas como de bots, tanto conectadas como aisladas; y tiene un mejor balance de clases que Botometer-feedback. Por esta razón se decidió usar Cresci-rtbust como el conjunto de test, y utilizar las otras siete para primero hacer una validación cruzada y luego entrenar los modelos y testearlos en Cresci-rtbust.

Un problema que se dio a la hora de realizar la validación cruzada es que no se encontró en la literatura ningún método para crear particiones adaptadas a grafos y para predecir nodos. Normalmente se asignan los nodos aleatoriamente a cada partición sin considerar los enlaces entre estos. Por ello se desarrolló un nuevo método para crear particiones que preservase en la mayor medida de lo posible estas estructuras de grafo. La idea clave es seleccionar nodos semilla aleatoriamente, y por cada uno de ellos crear un cluster añadiendo un número m de sus vecinos ($k = 1$), vecinos de vecinos ($k = 2$), y así

Tabla 6: ID, descripción, y tipo (Booleano (B), Entero (I), or Real (R)) de las 33 características usadas en este trabajo. Las últimas tres columnas muestras si la característica está incluida en el conjunto correspondiente. El identificador de una cuenta es el que se acompaña de un "@".

ID	Descripción	Tipo	detectme	bot detective	complete
1	La cuenta usa el perfil de defecto	B	-	✓	✓
2	La cuenta usa la descripción de defecto	B	-	✓	✓
3	La cuenta usa la imagen de perfil de defecto	B	-	✓	✓
4	Cuenta verificada	B	-	✓	✓
5	El perfil contiene una URL	B	-	✓	✓
6	Se especifica una localización	B	-	✓	✓
7	El identificador contiene la palabra "bot"	B	-	✓	✓
8	El nombre de usuario contiene la palabra "bot"	B	-	✓	✓
9	La descripción contiene la palabra "bot"	B	-	✓	✓
10	Antigüedad de la cuenta en días	E	✓	-	✓
11	# publicaciones	E	✓	✓	✓
12	# tweets marcados me-gusta	E	✓	✓	✓
13	# seguidores	E	✓	✓	✓
14	# amigos (usuarios seguidos)	E	✓	✓	✓
15	Longitud del identificador	E	✓	✓	✓
16	Longitud del nombre de usuario	E	-	✓	✓
17	Longitud de la descripción	E	-	✓	✓
18	Longitud de la localización	E	-	-	✓
19	# veces que la cuenta fue incluida en una lista	E	-	✓	✓
20	# URLs en la descripción	E	-	✓	✓
21	# caracteres numéricos en el identificador	E	-	✓	✓
22	# caracteres numéricos en el nombre de usuario	E	-	✓	✓
23	# hashtags en el nombre de usuario	E	-	✓	✓
25	# hashtags en la descripción	E	-	✓	✓
25	# emojis en el nombre de usuario	E	-	-	✓
26	# emojis en la descripción	E	-	-	✓
27	Ratio seguidores/seguidos	R	-	✓	✓
28	Ratio publicaciones/antigüedad de la cuenta	R	-	-	✓
29	Ratio me-gusta/antigüedad de la cuenta	R	-	-	✓
30	Ratio de caracteres numéricos en el identificador	R	-	-	✓
31	Ratio de caracteres numéricos en el nombre de usuario	R	-	-	✓
32	Ratio de caracteres alfabéticos en el identificador	R	-	-	✓
33	Ratio de caracteres alfabéticos en el nombre de usuario	R	-	-	✓

hasta un valor prefijado K. Son estos clusters los que luego se seleccionan para agregarse a una partición u otra con la que realizar la validación cruzada.

Se usó esta propuesta de creación de particiones para realizar una validación cruzada de cinco iteraciones con los nueve modelos seleccionados. La Tabla 7 presenta los valores F1 medios y sus desviaciones estándar alcanzadas por cada uno de los seis modelos que no siguen el esquema pipeline, utilizando los tres conjuntos de características descritos en la tabla 6. Los resultados son la media de las cinco validaciones que se obtuvieron con cinco particiones. Además, en el caso de los clasificadores que utilizan un GCN el experimento se ejecutó cinco veces, para atenuar el posible efecto causado por la inicialización aleatoria de los pesos de las capas neuronales de GraphSAGE. Por lo tanto, en esos casos, los resultados son la media de 25 experimentos. Esta aleatoriedad de la red no es un pro-

blema para los algoritmos de Aprendizaje Automático, cuyos resultados medios proceden únicamente de la validación cruzada de cinco particiones.

Tabla 7: Medias y desviaciones estándar de los valores F1 obtenidos por los seis modelos que no son pipelines, en una validación cruzada de cinco iteraciones. Por cada modelo, se destaca el mejor resultado entre los tres conjuntos de características usados. El mejor resultado general está subrayado.

	conjunto de características		
Clasificador	**detectme**	**botdetective**	**complete**
RF	$0{,}982 \pm 0{,}003$	$0{,}983 \pm 0{,}003$	$\underline{\mathbf{0{,}984 \pm 0{,}002}}$
SVM	$0{,}897 \pm 0{,}005$	$0{,}896 \pm 0{,}005$	$\mathbf{0{,}898 \pm 0{,}005}$
NB	$\mathbf{0{,}842 \pm 0{,}005}$	$0{,}829 \pm 0{,}008$	$0{,}830 \pm 0{,}008$
GCN	$0{,}925 \pm 0{,}020$	$0{,}956 \pm 0{,}009$	$\mathbf{0{,}960 \pm 0{,}009}$
GCN+RF	$\mathbf{0{,}925 \pm 0{,}033}$	$0{,}920 \pm 0{,}043$	$0{,}917 \pm 0{,}045$
GCN+SVM	$0{,}860 \pm 0{,}079$	$\mathbf{0{,}900 \pm 0{,}070}$	$0{,}899 \pm 0{,}069$

El efecto observado por usar un conjunto de características u otro en los clasificadores de Aprendizaje Automático es mínimo, es decir, inferior al 1% del valor F1, y está dentro de la desviación estándar. Lo mismo puede observarse en los modelos GCN+RF y GCN+SVM, pero con una desviación estándar mayor, que surge de la aleatoriedad inicial del modelo de red neuronal antes de ser entrenado. Sin embargo, el efecto del conjunto de características es más notable en GCN: el uso de un conjunto con más características aumenta el valor F1 medio de $0{,}925$ con "detectme" –que estaba formado por 6 características– a $0{,}960$ con "complete" (33 características). Además, la desviación estándar también se reduce aproximadamente a la mitad al utilizar la información adicional del conjunto de características "complete".

Los resultados de los pipelines son demasiado extensos para esta Sección, por lo que se incluyeron en el Anexo A. En la Tabla 8 se muestra el mejor resultado de cada modelo pipeline, así como sus configuraciones, es decir, los umbrales y conjuntos de características utilizados por el modelo de soporte y por el modelo GCN, dado que pueden usar conjuntos de características distintos. En general, los efectos de elegir un conjunto de características u otro no son significativos, como se observa en el caso de los modelos no pipeline (véase la Tabla 7). Además, se observa que la tendencia general es que el valor F1 medio disminuye ligeramente a medida que aumenta el umbral en todas las tablas del Anexo. Esto es coherente con la explicación de las limitaciones de GCN anterior: el rendimiento de GCN depende en gran medida del número de etiquetas del conjunto utilizado para el entrenamiento, que es menor cuando el umbral es más alto, ya que el modelo de soporte predice menos usuarios para que GCN los utilice.

Tras realizar la validación cruzada, se testearon los nueve modelos en el conjunto de datos Cresci-rtbust. Los resultados F1 de los seis modelos que no son pipelines se muestran en la Tabla 9, y los resultados de la mejor configuración de cada uno de los tres pipelines se muestran en la Tabla 10. La información completa de los resultados con todas las

Tabla 8: Medias y desviaciones estándar de los valores F1 de la mejor configuración de cada uno de los tres pipelines, indicada en la segunda columna como el umbral t usado, el conjunto de características usado por el modelo de apoyo y el conjunto de características usado por el modelo GCN. Los resultados proceden de la validación cruzada de cinco iteraciones.

Pipeline	**Mejor configuración**	**F1**
RF-GCN	$t = 0{,}8$, detectme, detectme	$0{,}970 \pm 0{,}003$
RF-GCN+RF	$t = 0{,}8$, detectme, complete	$0{,}979 \pm 0{,}004$
RF-GCN+SVM	$t = 0{,}8$, detectme, complete	$0{,}978 \pm 0{,}005$

configuraciones para los pipelines se encuentra en el Anexo A.

Tabla 9: Medias y desviaciones estándar de los valores F1 obtenidos por los seis modelos que no son pipelines, testeándolos sobre el conjunto de datos Cresci-rtbust. La desviación estándar de los modelos de Aprendizaje Automático es cero porque son deterministas y solo hay un conjunto de test. Las variantes de GCN presentan desviación estándar repetirse el experimento cinco veces, y en cada entrenamiento la red neuronal no queda necesariamente en el mismo estado. Por cada modelo, se destaca el mejor resultado entre los tres conjuntos de características usados. El mejor resultado general está subrayado.

	conjunto de características		
Clasificador	**detectme**	**botdetective**	**complete**
RF	$0{,}694 \pm 0{,}000$	$\underline{\mathbf{0{,}708 \pm 0{,}000}}$	$0{,}705 \pm 0{,}000$
SVM	$\mathbf{0{,}678 \pm 0{,}000}$	$0{,}677 \pm 0{,}000$	$0{,}677 \pm 0{,}000$
NB	$0{,}685 \pm 0{,}000$	$\mathbf{0{,}687 \pm 0{,}000}$	$0{,}684 \pm 0{,}000$
GCN	$\mathbf{0{,}632 \pm 0{,}009}$	$0{,}599 \pm 0{,}017$	$0{,}600 \pm 0{,}011$
GCN+RF	$0{,}651 \pm 0{,}022$	$\mathbf{0{,}671 \pm 0{,}040}$	$0{,}643 \pm 0{,}061$
GCN+SVM	$\mathbf{0{,}707 \pm 0{,}020}$	$0{,}601 \pm 0{,}115$	$0{,}656 \pm 0{,}036$

Tabla 10: Medias y desviaciones estándar de los valores F1 de la mejor configuración de cada uno de los tres pipelines, indicada en la segunda columna como el umbral t usado, el conjunto de características usado por el modelo de apoyo y el conjunto de características usado por el modelo GCN. Los resultados proceden del test sobre el conjunto de datos Cresci-rtbust.

Pipeline	**Mejor configuración**	**F1**
RF-GCN	$t = 0{,}8$, botdetective, detectme	$0{,}697 \pm 0{,}015$
RF-GCN+RF	$t = 0{,}8$, botdetective, detectme	$0{,}784 \pm 0{,}013$
RF-GCN+SVM	$t = 0{,}85$, detectme, detectme	$0{,}732 \pm 0{,}006$

La dificultad de Cresci-rtbust puede apreciarse comparando los resultados del test con los de la validación cruzada. El rendimiento del mejor clasificador de Aprendizaje Automático, RF, cayó de $\sim 98\,\%$ a $\sim 70\,\%$. El valor F1 del modelo de Alhosseini et al. [Ali Alhosseini et al., 2019] , GCN utilizando el conjunto de características "detectme", cayó de $0,925$ a $0,632$; y entre las variantes GCN, sólo GCN-SVM alcanzó un resultado superior

a 0,700. Los modelos pipeline también sufrieron una caída en su valor F1 con respecto a la validación cruzada, aunque una caída significativamente menor para las versiones que utilizaron GCN+RF o GCN+SVM.

En esta prueba, la ventaja de los modelos pipeline es notable: todas las variantes de pipeline que utilizan GCN+RF o GCN+SVM obtienen resultados significativamente mejores que los tres modelos de Aprendizaje Automático y las tres variantes de GCN. En particular, el mejor pipeline RF-GCN+RF logró un valor F1 medio de 0,784, que es un 24 % superior a la propuesta original de Alhosseini et al. [Ali Alhosseini et al., 2019] , que obtuvo un F1 medio de 0,632 en las mismas condiciones.

En cuanto a los efectos de utilizar un conjunto de características mayor, estos fueron insignificantes al igual que ocurrió con la validación cruzada. Si se consideran las seis características de "detectme", pueden separarse en tres categorías en función del tipo de información que ofrecen:

- Sobre el mantenimiento de la cuenta: la antigüedad de la cuenta y la longitud del identificador de la cuenta. Este último se incluye aquí porque Twitter primero autogenera uno, y luego es el usuario quien puede modificarlo posteriormente.
- Sobre la actividad de la cuenta: el recuento de me-gustas, tweets publicados y retweets; y el recuento de amigos.
- Acerca de cómo la cuenta es percibida por otros usuarios: el recuento de seguidores.

El resto de las 33 características o bien combinan esos campos, como la proporción entre los tweets publicados y la antigüedad de la cuenta; o bien se refieren a los demás campos que pueden ser modificados por el usuario, excepto "el número de veces que la cuenta fue incluida en una lista". En otras palabras, todas estas características adicionales del conjunto "detectme" añaden más información sobre cómo se personalizó la cuenta. Sin embargo, si la cuenta falsa se crea manualmente, el diseñador también puede personalizar estos campos. Y, si el propietario del bot prefiere generar automáticamente un lote de cuentas falsas, entonces cada bot podría simplemente configurar un perfil personalizado a partir de una plantilla, o simplemente copiar la personalización del perfil de un usuario aleatorio [Unhackthevote, 2019] . Por tanto, una posible explicación de los resultados es que la información extra de las funciones adicionales puede ser menos útil para detectar cuentas de bots que la proporcionada por el conjunto "detectme".

6 Detección de bots sociales basada en el comportamiento del usuario

La red social antes conocida como Twitter, ahora rebautizada como X [Corse et al., 2023] , ha sido objeto de gran atención tras la decisión de Elon Musk de comprarla en 2022

[Conger and Hirsh, 2022] . La supuesta motivación de Musk para comprar la empresa era contrarrestar los bots maliciosos presentes en la red social [Keenan, 2022] . Ciertamente, las cuentas automatizadas en las redes sociales son capaces de iniciar comportamientos destructivos, desde la distribución de propaganda política hasta el ciberacoso y la distribución de spam y archivos infectados [Tardelli et al., 2022; TS and Sreeja, 2022; Srinivas et al., 2022; Gupta et al., 2023; Breuer et al., 2023] . Además, los nuevos bots sociales disponen de técnicas para eludir los mecanismos de seguridad implementadas para evitar que dichos bots puedan registrarse, lo que facilita la creación de una gran cantidad de cuentas nuevas [Gao et al., 2022] .

Con el objetivo de identificar bots dentro de la red de Twitter, se investigó y desarrolló un detector automatizado basado en el comportamiento de las cuentas [Shevtsov et al., 2022; Arin and Kutlu, 2023] . El clasificador propuesto funciona de la siguiente manera:

1. Primero se usa BERTopic, un framework basado en BERT, para determinar el tópico o tema del que trata cada uno de los tweets de los usuarios a analizar. En particular, los tweets se clasifican en 102 categorías, correspondientes a 100 posibles tópicos mas una categoría de “otros tópicos” y otra de “sin texto”, correspondiente a, por ejemplo, publicar un solo emoticono.

2. Usando la información anterior, se construye una representación de la actividad de cada usuario en ventanas de tiempo de 15 minutos. Cada ventana de tiempo recoge los dos tópicos más tweeteados y los dos tópicos más retweeteados. Finalmente se añade una pseudo-ventana de tiempo final del mismo tamaño que el resto de ventanas, pero conteniendo otros metadatos del usuario.

3. Finalmente, un clasificador basado en LSTM predice si el usuario es bot o humano.

Para llevar a cabo la experimentación, se construyó un nuevo conjunto de datos sobre bots de Twitter recogiendo muestras directamente de la red social y etiquetándolas a mano, que resultó en un total de 17,945 muestras de las cuales había un 85,27 % de cuentas humanas y un 14,73 % de cuentas automatizadas, sin distinguir si dichas cuentas son benignas o no. Por ello el clasificador entrenado en este conjunto de datos detecta si las cuentas están automatizadas, pero no si dicha actividad es maliciosa. La idea es que las cuentas automatizadas benignas, como puede ser un bot que informe del clima, pueden pedir ser añadidas a una lista blanca por parte de los empleados del red social.

Este nuevo conjunto de datos es el más extenso que se conoce sobre bots en Twitter de acuerdo a la revisión bibliográfica realizada en cuanto a cuentas etiquetadas manualmente. Además, el conjunto de datos incluye la siguiente información por cada cuenta: (1) Los metadatos públicos de cada usuario, (2) sus últimas 200 publicaciones –tweets y retweets– (3) sus últimos 100 me-gusta (4) las conexiones a otros usuarios del conjunto de datos.

Debido al desequilibrio entre las dos clases, donde el número de cuentas humanas es mucho mayor que el número de cuentas bot, se dividió el conjunto de datos en las siguientes particiones:

- Dos particiones para test, una con cuentas humanas y otra con bots, cada una de 555 muestras.
- Siete particiones de cuentas humanas para entrenamiento, con 2088 muestras cada una.
- Una partición de entrenamiento con 2088 bots.

Las particiones de entrenamiento se usaron de la siguiente manera. Se entrenaron siete modelos clasificadores basados en LSTM usando cada una de las siete particiones de entrenamiento de cuentas humanas, mas siempre la misma partición de entrenamiento de cuentas bot. Después se construyó un ensamblado de los siete modelos, donde cada uno predice la probabilidad de que una cuenta sea bot o humana y se toma la media de las siete probabilidades para dar la predicción final. En la Tabla 11 se recogen los resultados de cada uno de los siete LSTMs y del ensamblado de todos ellos sobre el conjunto de test.

Tabla 11: Resultados sobre las particiones de test –555 bots y 555 cuentas humanas – de los siete clasificadores LSTM y del ensamblado resultante de combinarlos. Los datos resaltados corresponden al resultado más alto de cada métrica.

Modelo	**Accuracy**	**Precision**	**Recall**	**F1**
1	0.7081	0.6657	0.8360	0.7412
2	0.7189	0.6763	0.8396	0.7491
3	0.7072	0.6592	**0.8576**	0.7454
4	0.7342	0.7031	0.8108	0.7531
5	0.7405	0.7050	0.8270	**0.7611**
6	**0.7675**	**0.7940**	0.7225	0.7566
7	0.7621	0.7709	0.7459	0.7582
Ensamblado	0.7550	0.7134	0.8523	0.7767

Si se observan los resultados de la Tabla 11, se aprecia un equilibrio entre la precisión y el recall de los modelos individuales: una precisión relativamente alta va acompañada de una recuperación relativamente baja y viceversa. Sin embargo, el ensamblado mantiene un recall relativamente alto y una precisión relativamente superior a la media, con lo que obtiene una puntuación F1 más alta que cualquiera de los modelos individuales. Su accuracy también está a la par con los resultados de mayor precisión de los modelos individuales, lo que demuestra la solidez del ensamblado.

7 Conclusiones y perspectiva

7.1 Resumen del trabajo

Esta Tesis Doctoral se centra en la identificación de diferentes tipos de bots, que son entidades automatizadas que operan siguiendo las órdenes remotas de un botmaster, a menudo con fines maliciosos, de ahí la motivación para su detección. Los cuatro trabajos realizados durante el programa de doctorado pueden clasificarse en dos líneas de investigación principales: detección de botnets malware e identificación de botnets sociales.

La primera línea de investigación se refiere a la detección de botnets mediante malware, es decir, que son redes de dispositivos infectados por virus informáticos llamados bots. Engloba los dos primeros trabajos. El primero se centra en la optimización de clasificadores de Aprendizaje Automático mediante la selección de características adecuadas para crear un detector eficiente del tráfico de redes bot. Este trabajo se detalla en el Capítulo 3. El procedimiento para seleccionar el mejor subconjunto de características implicó la combinación de dos métodos de clasificación de características, Importancia de Gini y Ganancia de Información. El resultado indicó que el subconjunto óptimo incluiría entre cinco y siete características. Con esta información, analizamos el rendimiento de tres clasificadores: el árbol de decisión, el bosque aleatorio y k-Vecinos Cercanos, con cada uno de los tres subconjuntos de características. El Árbol de Decisión que utilizó un subconjunto de cinco características fue el que mejor funcionó entre los candidatos, obteniendo un valor F1 media de $0,85$ y clasificando cada muestra en una media de $0,78$ microsegundos.

El segundo trabajo, desarrollado en el Capítulo 4, se basa en la investigación anterior sobre la detección de malware bot, cambiando su enfoque para lograr la detección en tiempo real. En concreto, el objetivo era desarrollar un clasificador de Aprendizaje Automático que pudiera extraer las características de un segundo de tráfico de red como máximo un segundo después, e identificar conexiones bot en ese tráfico como máximo un segundo adicional después, es decir, que necesitara como máximo dos segundos para dar el informe. También se puso como objetivo que el modelo funcionara en redes de gran ancho de banda (simulando hasta 10 Gbps) y que fuera lo suficientemente robusto como para detectar rastros de bots incluso en condiciones de saturación, es decir, cuando el ancho de banda se utilizara hasta un 110% de su capacidad máxima y, por tanto, la red dejara caer hasta un 10% de los paquetes. El modelo propuesto cumplió con éxito estos requisitos, alcanzando un valor F1 de $0,926$ en el conjunto de datos EQB-CTU13 que se elaboró en la primera investigación de este doctorado. Los requisitos de hardware estimados para que este modelo funcione en redes de 10 Gbps son 19 núcleos de CPU funcionando a $2,4$ GHz.

La segunda línea de investigación, que comprende los trabajos realizados sobre la detección de cuentas falsas y automatizadas en medios sociales, habitualmente denominadas bots sociales, incluye los dos últimos de los cuatro trabajos.

El objetivo inicial del tercer trabajo era desarrollar un detector de cuentas de bots pa-

ra Twitter, mejorando el modelo base que empleaba la Red Convolucional de Grafo (GCN por su nombre en inglés) de Kipf y Welling [Kipf and Welling, 2017] . Sin embargo, tras una investigación más profunda, se identificaron dos problemas. En primer lugar, las GCN tienen limitaciones para predecir la clase de nodos en regiones del grafo con escasas o inexistentes conexiones con los nodos de entrenamiento [Zhang et al., 2019] . Y en segundo lugar, se encontró una falta de métodos para dividir los datos del grafo en particiones que garanticen una adecuada preservación de la estructura del grafo. Se propusieron soluciones para estos dos problemas que pueden aplicarse a cualquier tipo de grafos, no solo los de redes sociales. Con estas contribuciones, se desarrolló un modelo que superaba el valor F1 de referencia en un 24 %.

El cuarto y último trabajo se centra en la detección de cuentas falsas en Twitter mediante el análisis del comportamiento de los usuarios. El detector propuesto tiene en cuenta el tema y la frecuencia de las publicaciones de cuentas, analizados mediante modelos BERT y LSTM, respectivamente. En el momento de dicha experimentación, los conjuntos de datos existentes presentaban inconvenientes que no permitían realizar un análisis que combinara tanto las relaciones entre las muestras (es decir, las conexiones existentes entre ellas), como sus publicaciones. O bien contenían poca información, es decir, en su mayoría sólo metadatos de cuentas de Twitter, o si contenían información sobre las publicaciones el etiquetado de los datos era incompleto o no se hizo manualmente, sino automáticamente con un modelo entrenado. Por ello, para asegurarse de que el modelo propuesto en esta Tesis se evaluaba en un conjunto de datos que contuvieran toda la información necesaria, como un registro de las publicaciones más recientes de los usuarios, se creó un nuevo conjunto de datos con muestras de Twitter etiquetadas manualmente. Hasta se sabe por la revisión de la literatura realizada, este conjunto de datos es el mayor de los disponibles públicamente sobre bots de Twitter en términos de datos etiquetados manualmente. El modelo propuesto alcanzó una precisión de 0,7550 y un valor F1 de 0,7767 en este conjunto de datos.

7.2 Resumen de las contribuciones

Las principales contribuciones presentadas en esta Tesis son:

1. Se construyeron dos nuevos conjuntos de datos de tráfico botnet y legítimo basados en el conocido conjunto de datos CTU-13 [Garcia et al., 2014] . El primer conjunto de datos, QB-CTU13, pretende abordar el problema del desequilibrio de clases cuando el tráfico se divide en flujos de red. Abarca tráfico procedente del uso legítimo de ordenadores, además de trazas seleccionadas manualmente de siete redes de bots. El EQB-CTU13 amplía el QB-CTU13 incorporando trazas de tres botnets adicionales, que se seleccionaron en función de su dificultad para ser detectadas por detectores automáticos, según la revisión bibliográfica.

2. Se propone un detector de tráfico de botnets eficiente basado en el algoritmo Árbol

de Decisión que puede clasificar flujos de tráfico en un tiempo medio de 0,78 microsegundos cada uno, alcanzando un valor F1 de 0,85 sobre el conjunto de datos EQB-CTU13.

3. Se presenta una arquitectura bimodular basada en Árbol de Decisión capaz de analizar el tráfico en tiempo real para identificar la actividad de las botnets. Puede escalarse para operar en grandes anchos de banda (es decir, hasta 10 Gbps), requiriendo 19 núcleos de CPU operando a una frecuencia de 2,4 GHz.

4. Se propone un novedoso enfoque basado en el modelo GCN de Kipf y Welling [Kipf and Welling, 2017] para resolver sus limitaciones en la predicción precisa de la clase de nodo en regiones del grafo donde las conexiones a los nodos de entrenamiento son escasas o inexistentes [Zhang et al., 2019] . Aplicado al problema de la detección de bots sociales, este detector mejora el valor F1 del modelo GCN base [Ali Alhosseini et al., 2019] en un 24 % sobre el conjunto de datos Cresci-rtbust [Mazza et al., 2019] .

5. Se ha desarrollado un nuevo algoritmo para repartir los nodos de un grafo (o un conjunto de grafos) en particiones para validación cruzada. Este enfoque mejora la preservación de los datos de las conexiones entre nodos a la vez que garantiza la aleatoriedad en la selección de nodos. Además, se presenta un estudio para demostrar la robustez de este algoritmo.

6. Se construyó un nuevo conjunto de muestras de Twitter de de 2022 que incluye los metadatos públicos de los usuarios, las relaciones entre seguidores en todo el conjunto de datos – lo que garantiza una estructura de grafos relativamente densa – y la actividad de publicación reciente (hasta 200 publicaciones y 100 me-gusta). Este conjunto de datos contiene 17,945 muestras que se etiquetaron manualmente como bots o humanos lo que lo convierte, hasta donde se sabe por la revisión bibliográfica realizada, en el más completo, reciente y amplio en términos de datos etiquetados manualmente de todos los conjuntos de datos de Twitter conocidos públicamente.

7. Se ha desarrollado un método novedoso para identificar bots sociales en Twitter a través de su comportamiento. Este sistema utiliza un predictor basado en BERT para identificar los temas de los tweets y un conjunto de siete clasificadores basados en LSTM para analizar la frecuencia de publicación. Este detector alcanza una precisión de 0,7550 y un valor F1 de 0,7767 en el nuevo conjunto de datos mencionado en el punto anterior.

7.3 Problemas abiertos y trabajo futuro

La primera parte de esta Tesis, sobre detección de bots de malware usando Aprendizaje Automático, concluyó con dos propuestas, un detector eficiente y un detector en

tiempo real. Los experimentos y la comparación con otros modelos se han realizado utilizando Python 3. Aunque estos resultados son suficientes para una prueba de concepto, los modelos podrían ser aún más eficientes en la dimensión temporal si se construyeran utilizando un lenguaje compilado como C/C++ o Rust.

Además, una ventaja de los clasificadores de Aprendizaje Automático es que se pueden reentrenar nuevas instancias con nuevos datos y sustituir a las instancias antiguas. Esto permite a este tipo de detectores adaptarse rápidamente a nuevas variaciones de las redes bots conocidas, o incluso detectar nuevos tipos de bots. Una nueva línea de investigación consistiría en medir la evolución de la eficacia de una sistema de este tipo – que re-entrena las instancias, pero el algoritmo sigue siendo el mismo– a lo largo del tiempo.

El trabajo sobre la detección de bots malware se ha centrado en el análisis del tráfico a nivel de flujo, es decir, analizando las comunicaciones entre dos dispositivos. Sin embargo, los bots operan en redes y, por tanto, hay información relevante que puede obtenerse observando toda la red. Este tipo de análisis no es especialmente adecuado para la detección en tiempo real, ya que requeriría monitorizar la red durante periodos de tiempo más largos y extraer correlaciones. Pero podría ser complementario a los detectores basados en flujos, de modo que los administradores de sistemas podrían obtener dos tipos de informes sobre este aspecto de la red: informes en tiempo real para una detección temprana, e informes más detallados más adelante.

En la segunda parte de esta Tesis, so propusieron dos identificadores de bots sociales, especialmente diseñados para Twitter. En futuros trabajos, estos modelos podrían adaptarse a otras redes sociales.

La segunda propuesta de detector de bots sociales se centra en el comportamiento de las cuentas, en particular en su historial de publicaciones y los temas tratados. Sin embargo, esta propuesta no diferencia entre bots maliciosos y bots benignos o legítimos, por ejemplo cuentas para informar del tiempo, eventos u otra información de interés. En futuros trabajos se pueden diseñar nuevas propuestas que hagan esta distinción.

Además, hay más dimensiones que analizar en términos del comportamiento, como las conversaciones que mantienen los usuarios. Esto abre un amplio campo de investigación para explorar nuevas formas de caracterizar el comportamiento y emplearlo en la detección de bots.