Programa de Doctorado en
Ingeniería de la Información y el Conocimiento

UNSUPERVISED INTRUSION DETECTION WITH CROSS-DOMAIN ARTIFICIAL INTELLIGENCE METHODS

Tesis Doctoral presentada por

RAFAEL SAN MIGUEL CARRASCO

Director:
DR. MIGUEL ÁNGEL SICILIA URBÁN

Alcalá de Henares, 2021

Unsupervised Intrusion Detection with Cross-Domain Artificial Intelligence Methods

Rafael San Miguel Carrasco

Index of contents

1. Introduction

1.1. Motivation and objectives

Cybercrime is a major concern for corporations, business owners, governments and citizens. Major incidents are causing significant damage to organizations and consumers, including economic losses, reputational impact and failure to meet compliance requirements. This trend continues to grow in spite of increasing investments in security and fraud prevention, public and private sector initiatives geared towards collaboratively defeating cybercrime, and new regulations pushed from national and international bodies.

On the other hand, security professionals are equipped with tools, techniques and technologies, each designed to solve a given problem. However, these resources must be used in combination with others (layered approach) and effectively orchestrated in order to obtain optimal results. These solutions heavily rely on existing knowledge about attack and fraud patterns. They are built upon past incidents and procedures used by attackers to ensure that these methods (or their variants) won't succeed in the future once they are deployed in the environment to be protected. They also rely on human experts to analyze triggered alerts and discard false positives, focusing on those alerts that are more likely to represent true attacks.

However, as novel attack techniques are developed, with increasing degree of sophistication, existing expert knowledge quickly becomes obsolete. Also, as more devices and business channels are monitored for potential intrusions, the volume of events, and the number of false positives becomes unmanageable for analysts.

The aim of this research work was to target both problems (detecting unknown attacks and reducing false positive ratio) by leveraging artificial intelligence techniques. Artificial Intelligence was chosen for two reasons: it exhibits the ability to learn from both data and human knowledge, and it can potentially generalize.

While the first statement has been extensively demonstrated by the research community, the second statement (ability to generalize) is still a green field. Artificial General Intelligence (AGI) is even nowadays a theoretical concept, not a reality.

However, cross-domain applications of artificial intelligence, which sits between applied artificial intelligence and AGI, looks to be achievable. The goal of this research was to develop means to solve an existing problem in a given domain (intrusion detection) and then leverage the same underlying tools (AI) to solve a related problem in a related domain (fraud).

While different in purpose, data nature and volume, and business implications, fraud detection shares with intrusion detection the aim to detect malicious activity to prevent incidents from occurring. It also shares the need to go beyond current human expert knowledge and reduce the volume of alerts that must be reviewed by human experts. These common factors made fraud detection an ideal field to reuse AI techniques.

Artificial intelligence can learn in a supervised and unsupervised fashion. Depending on the use case, one paradigm might more suitable than the other. The focus of this research has been placed on unsupervised techniques to approach the primary problem (intrusion detection), with supervised techniques becoming helpers for more specific tasks (reducing false positive ratio). Under the aforementioned context, the objectives of this research were the following:

- Reduce the dependence on well-known attack patterns for effective intrusion detection, by leveraging AI techniques that can distinguish malicious activity by modeling legitimate activity with network features.
- Prove that artificial intelligence, while far from AGI, can evolve towards cross-domain applications and use cases, as long as target domains are related and share some principles and needs.
- Demonstrate that unsupervised intrusion detection can be augmented by supervised learning, thus effectively increasing their accuracy.
- Reduce false positive ratio of fraud detection systems by proving that human expertise used to confirm or discard alerts can be (partially) captured and synthetized by AI, in order to reduce the volume of alerts that require manual review.

1.2. Target fields

Cybercrime is a fast-growing area of crime which affects both individuals and organizations, including governments, corporations and non-profit associations (Udo, Bagchi, & Kirs, 2018) (Tsakanyan, 2017). It has increased globally and jumped by 32 per cent in 2016 (*Cybersecurity Challenges in the Middle East*, 2017). It's not only increasing in number of occurrences but also in cost per incident, which doubles each year (Lukasik, 2000). In fact, global cybercrime cost has been estimated (*Net Losses: Estimating the Global Cost of Cybercrime*, 2014) to be $400 billion in 2014. These estimates might be conservative, because companies are hesitant to report security incidents in surveys (Chen, Dong, Chen, & Xu, 2016), and due to the fact that security incidents go undetected for long time (Rudd, Rozsa, Günther, & Boult, 2016). On the other hand, on average, the announcement of a breach has a negative impact of about 2.1% of the market value of the company (Chen, Dong, Chen, & Xu, 2016). These figures suggests that existing security measures, techniques and systems are not effective against most advanced attacks (Ullah & Babar, 2019).

On the other hand, security and privacy concerns in computer networks and systems have significantly increased over the last few years (da Costa, Papa, Lisboa, Munoz, & de Albuquerque, 2019). This is true for computers, mobile devices, IoT devices (Chapaneri & Shah, 2019), and payment networks. Malware is also major area of concern (Idika & Mathur, 2007). In both contexts, machine learning has been subject to extensive research (Akhi, Kanon, Kabir, & Banu, 2019) (Firdausi, Erwin, & Nugroho, 2010), as it can accurately spot differences between legitimate and malicious activity by generalizing and finding patterns in data (Liu & Lang, 2019). Fraud, a growing issue in online payments (Fiore, De Santis, Perla, Zanetti, & Palmieri, 2019), has also been approached with machine learning (Raj & Portia, 2011).

This research work targets a primary field (unsupervised intrusion detection) and applies the underlying techniques (AI) to a related domain (fraud detection).

1.2.1. Intrusion detection

1.2.1.1. Challenges

An IDS (Intrusion Detection Systems) is designed to recognize intrusions and trigger alerts, obtaining high intrusion detection ratios and acceptable (as low as possible) false positive ratios (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017). An IDS can leverage the following types of detection method: signature-based, anomaly-based or specifications-based (Hindy, Brosset, Bayne, Seeam, Tachtatzis, Atkinson, & Bellekens, 2018).

Signature-based systems rely on prior knowledge about attack patterns. In these systems, a signature is created for each known pattern, which results in a high degree of accuracy for known attacks, but a lack of detection capabilities for unknown attacks (Ashraf, Ahmad, & Ashraf, 2018). This limitation also applies to systems based on supervised settings, as they rely on the availability of labelled instances of attacks. In fact, previous research (Zarpelão, Miani, Kawakani, & de Alvarenga, 2017) states that traditional techniques cannot be successfully applied to different environments, including emerging ones as IoT, which uses specific protocols, standards and devices.

Therefore, recognizing unknown attacks is one of most desired features of an intrusion detection technique (Casas, Mazel, & Owezarski, 2012). Unknown attacks don't have a pattern, neither statistical nor signature-based. Anomaly-based systems could detect unknown attack patterns, assuming that malicious behavior is significantly different from legitimate activity. Their main drawbacks (Carlin, Hammoudeh, & Aldabbas, 2015) are: usability dependent on the false alarm rate, effective behavior models are difficult to design, greater implementation complexity, and inability to explain the type of attack that was detected.

On the other hand, an IDS must process significant amounts of data that is received from multiple other systems (Garvey & Lunt, 1991). Moreover, there could be irrelevant data that won't improve accuracy, and perhaps key data or features are missing.

In this scenario, intrusion detection is now a relevant field for researchers. However, there are a number of additional challenges that researchers must face. First, cyber-attacks leverage stealth techniques for intrusion detection evasion in signature-based approaches (used by most antivirus software packages) (Alston, 2017). These techniques are evolving towards evading machine learning anomaly detection techniques as well (Viegas, Santin, Abreu, & Oliveira, 2017). Also, new technology eco-systems like mobile (Zhang, Lee, & Huang, 2003), cloud (Joshi, Joshi, & Rani, 2017) and IoT (Minoli, Sohraby, & Kouns, 2017) are demanding custom security approaches. On the other hand, there is increasing regulatory pressure around data protection (Raab & Szekely, 2017), and a growing amount of business applications of big data (Choi, Chan, & Yue, 2016), which encourages companies to store high volumes of data, hence increasing the attack surface and the available threat vectors.

1.2.1.2. Development paths

1.2.1.2.1. Increase detection rate

Recent surveys (Ahmed, Mahmood, & Hu, 2016) (Keegan, Ji, Chaudhary, Concolato, Yu, & Jeong, 2016) (Buczak & Guven, 2015) and taxonomies (Qayyum, Islam, & Jamil, 2005) show that there are multiple intrusion detection techniques leveraging Machine Learning or Artificial Intelligence. They can be based on anomaly detection (unsupervised) or misuse detection (supervised).

Common attacker activities are learnt and compared with current activity in misuse detection (Cannady, 1998). These malicious activities are labeled as such in the training dataset, which must be available for the learning process to occur. All other observations in this training dataset are labeled as normal activities. Unknown attacks are difficult to detect, because they were not present in the learning phase (Meira, Andrade, Praça, Carneiro, Bolón-Canedo, Alonso-Betanzos, & Marreiros, 2020).

It's also remarkable that datasets with labeled malicious activity are not widely available. There are some challenges specific to the intrusion detection field that make it hard to produce those datasets.

For example, the activities surrounding a given action, that is, its context, heavily influence whether that action is considered an attack or not. In addition to this, recognizing an intrusion might require looking at a number of related events, often produced by more than one system. Therefore, a single event cannot be classified (alone) as an attack or legitimate. Lastly, there are multiple datasets with synthetic attack data, but datasets with realistic intrusions are scarce. Moreover, the research community cannot reliably assess attack statistics and indicators (Kuypers, Maillart, & Paté-Cornell, 2016).

In anomaly detection techniques, the focus is on learning legitimate activity, not malicious actions (Chimphlee, Abdullah, Sap, Srinoy, & Chimphlee, 2006), in order to discover patterns that don't align with that legitimate activity (Chandola, Banerjee, & Kumar, 2009). Attack labels are not needed. The main objective is to recognize actions that can't be fit to most usual activity learnt in the training phase (Chandola, Banerjee, & Kumar, 2009). This approach has proven to be effective to be able to spot intrusions (Meira, Andrade, Praça, Carneiro, Bolón-Canedo, Alonso-Betanzos, & Marreiros, 2020), for a number of reasons.

First, evasion techniques cannot be easily applied. Their impact is much lower than in systems whose focus is to learn attack patterns. Second, because anomaly detection is based on broad and generally-applicable principles, it can be applied to all elements of a technology ecosystem without reducing its effectiveness. Third, the evolution of intrusion vectors doesn't influence these techniques' performance, because knowledge about these vectors is not required at all.

Moreover, anomaly detection exhibits the following advantages: they can detect unknown attacks, they can detect abuse of privileges, and they are not dependent on the underlying operating system (Liao, Lin, Lin, & Tung, 2013).

However, anomaly detection also exhibits several disadvantages: they can take longer to produce an alert when malicious activity happens, detection is sometimes disabled while legitimate activity is being learnt, and the fact that these techniques are vulnerable to concept drift, because legitimate activity patterns can change overtime (Liao, Lin, Lin, & Tung, 2013) (Lane & Brodley, 1998). Besides, creating patterns of legitimate activity can become a hard task to accomplish (Zamani & Movahedi, 2013). Also, the volume of data to be processed by an IDS is quite significant, attacks represent just a small fraction of that data, and discovering effective criteria to separate legitimate and malicious activity is a hard task (Zamani & Movahedi, 2013).

Anomaly detection requires more training data than supervised settings, which translates into higher demand for compute resources (Bashah, Shanmugam, & Ahmed, 2005). That training data can contain attacks, which can impact the quality of models produced (Estevez-Tapiador, Garcia-Teodoro, & Diaz-Verdejo, 2004). Lastly, in some cases the assumption that legitimate behavior is frequent might not hold (Estevez-Tapiador, Garcia-Teodoro, & Diaz-Verdejo, 2004), and infrequent activity might not be related to attacks.

Attackers could also influence anomaly detection algorithms to learn malicious activity as legitimate (Chebrolu, Abraham, & Thomas, 2005). By introducing small (incremental) changes in behavior over a long period of time, malicious activity would go unnoticed and would become part of the normal behavior profile. In any case, this manipulation requires details about how the algorithm was implemented, and the attacker must be able to generate activity while the algorithm is being trained.

Lastly, the actual performance of anomaly detection techniques relies on a list of assumptions (Gates & Taylor, 2006): attacks are anomalous or rare, anomalous activity is malicious, definition of

malicious is universal, attack-free data is available, simulated data (for those experiments in which legitimate activity is simulated) is representative, false alarm rate over 1% is acceptable, and administrators can interpret anomalies.

1.2.1.2.2. Reduce false positive rate

There are multiple approaches available for automated decision-making processes whose goal is to minimize false alarms. Most popular methods are adaptive learning (Pietraszek, 2004), similarity with confirmed alerts (Njogu & Jiawei, 2010), greedy aggregation algorithm (Harang & Guarino, 2012), neuro-fuzzy approach (Alshammari, Sonamthiang, Teimouri, & Riordan, 2007), alert enrichment framework (Bakar, Belaton, & Samsudin, 2005), and outlier detection (Xiao, Jin, & Li, 2010).

Alert reduction and false positive minimization are problems that share the same principles in intrusion detection and fraud detection.

1.2.2. Fraud detection

1.2.2.1. Challenges

Fraud detection shares key objectives with intrusion detection: maximize detection rate and minimize false positive rates, which lead to high performance. It differs from intrusion detection in the type of input features available to achieve that performance (Soh & Yusuf, 2019): Personal Account Number (PAN) and related details, transaction location, date and time, customer details, and the amount associated with the transaction.

The set of indicators used to measure performance in fraud detection are equivalent to those for intrusion detection: precision, recall, accuracy, and false alarms ratio (Kumari & Mishra, 2019). Nevertheless, the lack of a universal dataset to be used as reference for comparing performance across algorithms prevents these metrics to be really useful for benchmarking purposes. However, there are promising proposals (Yang, Zhang, Ye, Li, & Xu, 2019) that leverage federated learning, which rely on data sharing among banks. These initiatives might solve the aforementioned issue.

Fraud detection systems (FDS) are designed to detect fraudulent activity and trigger alerts. These systems implement techniques leveraging advanced detection methods, rule-based detection, artificial intelligence and statistical models. Transactions flagged as fraud must be reviewed by a fraud analyst (human expert) to check whether the FDS made the right decision or not. Significant cost reduction can be achieved by replacing these manual reviews with automated methods.

Alert reduction is especially challenging in the fraud detection field due to the evolving nature of the payments sector. There are new channels and methods to execute payments and new customer segments are constantly found. As a consequence, the number of transactions is always increasing, which also increases the number of manual reviews required to detect fraud. Growing the team of experts effectively allows to handle additional workload, but doesn't scale in terms of cost. A better option is to implement automated methods to quickly discard false alarms with common patterns, so that experts only have to review the less trivial cases.

1.2.2.2. Development paths

1.2.2.2.1. Increase detection rate

To increase detection performance, drivers in the fraud detection field resemble those seen for intrusion detection: feature selection and engineering (Singh & Jain, 2019) (Lucas, Portier, Laporte,

Calabretto, Caelen, He-Guelton, & Granitzer, 2019), ability to leverage ensembles of models (Kim, Lee, Shin, Yang, Cho, Nam, 2019) (Prusti & Rath, 2019a) or hybrid models (Carcillo, Le Borgne, Caelen, Kessaci, Oblé, & Bontempi, 2019) (Prusti, Padmanabhuni, & Rath, 2019), incorporated domain expertise (Rushin, Stancil, Sun, Adams, & Beling, 2017), and deciding between misuse detection and anomaly detection techniques (Niu, Wang, & Yang, 2019).

The underlying assumption in supervised models (misuse detection) is that current fraud can be recognized with a set of patterns that can be discovered and synthesized from past fraud data (Carcillo, Le Borgne, Caelen, Kessaci, Oblé, & Bontempi, 2019). In unsupervised models (anomaly detection), the underlying assumption is that legitimate activity can be baselined, and fraudulent activity is significantly different from that baseline. Lastly, hybrid systems come into play when the hypothesis is that standalone techniques fail to effectively detect fraud, and that using multiple techniques leveraging different approaches simultaneously is a preferred method. Recent research (Mittal & Tyagi, 2019) has evaluated the most innovative methods available today. The key conclusion is that, when it comes to accuracy, unsupervised methods yield better results (Mittal & Tyagi, 2019).

Beyond the chosen algorithm, feature selection and engineering also plays a relevant role in obtained performance. A novel feature engineering framework (Zhang, Han, Xu, & Wang, 2019) has demonstrated to perform better when applied to several methods based on artificial intelligence.

1.2.2.2.2. Reduce false positive rate

This development path hasn't been so extensively explored in the fraud detection field. However, given the similarity of this field with intrusion detection, it seems reasonable that these methods could be re-engineered to make them effective for fraud detection as well.

2. Systematic review

2.1. Motivation

This systematic review focuses on unsupervised techniques based on anomaly detection, that have been applied to the intrusion detection domain. The reason behind setting this scope is that the main goal of this research, as described in the motivation and objectives section, is to reduce the dependence on well-known attack patterns in the intrusion detection field, and unsupervised techniques are best suited for this purpose. It's therefore required to understand existing methods for unsupervised learning before enhanced techniques can be developed and tested.

While supervised settings are also present in this research work, they are leveraged only as helpers or secondary techniques for specific functions. Therefore, and extensive review is not a requirement, and the relevant literature review is included in a separate section.

Several anomaly detection methods have been applied to intrusion detection (Fernandes, Rodrigues, Carvalho, Al-Muhtadi, & Proença, 2019), with varying performance results. Moreover, recent surveys (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017) (Liao, Lin, Lin, & Tung, 2013) (Kwon, Kim, Kim, Suh, Kim, & Kim, 2019) have focused on producing a taxonomy to classify these methods. There are other surveys focused on the usage of anomaly detection to detect cyber-attacks (Chandola, Banerjee, & Kumar, 2009).

This review aims to provide a comprehensive and up-to-date reference of available techniques. It includes a brief description on how each of them operates, as well as their advantages, drawbacks and limitations, and what detection and false positive rates were obtained by researchers along the experimental testing phase, where available.

Surveys and taxonomies on the field of intrusion detection have already been developed and made available to the security research community. Recent surveys include references to latest developments around neural networks (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017) and extend the scope by including other aspects of security, like vulnerability assessments (Debar, Dacier, & Wespi, 2000). A recent taxonomy (Axelsson, 2000) focused on the actual systems used for intrusion detection, rather than their underlying techniques, while another (Liao, Lin, Lin, & Tung, 2013) took a comprehensive approach to review and classify modern intrusion detection systems.

The aforementioned reviews combine supervised and unsupervised techniques, misuse detection and anomaly detection approaches. Moreover, while several surveys focused on anomaly detection techniques for IDS already exist (Gyanchandani, Rana, & Yadav, 2012) (Garcia-Teodoro, Diaz-Verdejo, Maciá-Fernández, & Vázquez, 2009), they include both misuse detection (supervised) and anomaly detection (unsupervised) algorithms.

However, unsupervised techniques based on anomaly detection can only be assessed and compared when isolated from other available paradigms. Benchmarking cannot be accurate if some techniques make use of labeled data and others don't. Also, the ability to detect unknown attacks doesn't apply to misuse detection.

On the other hand, even though hybrid systems represent a promising path in intrusion detection, current implementations weaken the agnostic nature of anomaly detection by incorporating pre-existing patterns of well-known attacks. While the resulting performance is competitive, in most cases that performance is measured against known attacks, hence not measuring the real accuracy against any type of attack.

These arguments have motivated this systematic review, which aims at: collecting research done with most relevant methods that fall into this category; assess the strengths, advantages and constraints of each of those methods; identify most promising areas of research to drive future efforts.

2.2. Review method

2.2.1. Planning the review

The protocol used for this systematic review builds upon the following components: a set of research questions, sources of information, inclusion and exclusion criteria, and guidelines used to summarize results and focus the discussion and conclusions.

2.2.2. Research questions

The goal of this review was to collect, assess, classify and discuss available research on unsupervised techniques based on anomaly detection for intrusion detection.

To plan the review, a list of research questions was identified. They are shown in Table 1.

Research question	Motivation
(1) What is the current status of unsupervised methods based on anomaly detection in the IDS domain? (1.1) What algorithms and implementation approaches are used? (1.2) What detection and false positive rates do these methods achieve? (1.3) How applicable these methods would be to a real-life environment?	It helps in understanding what methods have been researched and are available in the target domain. For this purpose, most popular techniques and algorithms are referenced and compared. Studies which compare and benchmark related methods are also included.
(2) What criteria and processes are followed for data selection, feature extraction, and performance evaluation? (2.1) What datasets are leveraged to train, test, optimize and benchmark algorithms? (2.2) What feature extraction techniques lead to better results, and what features are typically extracted? (2.3) How is performance measured, and what metrics are used for that purpose?	It helps in understanding whether quality data is available to enable the development of effective methods, and whether obtained results are comparable. Multiple feature extraction strategies and their influence on performance are reviewed. Common criteria applied to measure performance of unsupervised methods for IDS are identified and assessed.
(3) Key sub areas (3.1) What are the main advantages and drawbacks of each method?	It helps in understanding what methods should be used under different scenarios.

(3.2) Which techniques are more likely to detect complex attacks which differ significantly from well-known attacks? (3.3) Which techniques would remain accurate when applied to other intrusion datasets, and can therefore generalize well?	Pros and cons of each technique are assessed. Techniques that are more resilient to advanced attackers are identified. Lastly, potential for generalization is evaluated.

Table 1. Research questions.

2.2.3. Information sources

For extensive coverage of the available literature in this research field, databases and information resources were researched. To the best of my knowledge, these resources are relevant for the target domain.

2.2.3.1. Research databases

Table 2 enumerates selected research databases.

Database name	URL
archiv.org - Cornell University Library	http://arxiv.org/
IEEE Xplore Digital Library	http://ieeexplore.ieee.org/
ResearchGate	https://www.researchgate.net/
SemanticScholar	https://www.semanticscholar.org/
ScienceDirect	http://www.sciencedirect.com/
CiteSeerX - The Pennsylvania State University	http://citeseerx.ist.psu.edu/index
SpringerLink	https://link.springer.com/
ACM Digital Library - Association for Computing Machinery	http://dl.acm.org/

Table 2. Research databases.

2.2.3.2. Additional sources

Table 3 enumerates selected additional sources.

Database name	URL
Google Patents	https://www.google.es/patents/
The Definitive Security Data Science and Machine Learning Guide	http://www.covert.io/the-definitive-security-datascience-and-machinelearning-guide/
Packet Storm	https://packetstormsecurity.com/
Black Hat	https://www.blackhat.com/html/archives.html
Google Scholar	https://scholar.google.com/

Table 3. Additional sources.

2.2.4. Search criteria

Keywords such “anomaly detection”, "intrusion detection" and "security" or "network security" were included in most abstract-related searches. They returned a significant number of results with varying levels of relevance.

Table 4 enumerates the search strings for selected sources.

Search no.	Source	Search string	#
1	archiv.org	Subject: Computer science Abstract: anomaly detection intrusion detection	76
2	IEEE Xplore Digital Library	(("Abstract":"anomaly detection") AND "Abstract":"intrusion detection")	502
3	ResearchGate	Publications: anomaly detection intrusion security	1.000+
4	SemanticScholar	"anomaly detection" intrusion security	10.757
5	ScienceDirect	pub-date > 1994 and TITLE-ABSTR-KEY("anomaly detection") and TITLE-ABSTR-KEY("intrusion detection")[All Sources(Computer Science)]	126
6	CiteSeerX	abstract:("anomaly detection" "intrusion detection" "network security") AND year:[1995 TO 2017]	600.997
7	SpringerLink	anomaly detection intrusion security	4.706
8	ACM Digital Library	anomaly detection intrusion security	52.609
9	Google Patents	"anomaly detection", "intrusion detection", "security"	4.296
10	The Definitive Security Data Science and Machine Learning Guide	Machine Learning and Security Papers Deep Learning and Security Papers Abstract: "anomaly detection"	
11	Packet Storm	Files: anomaly detection pdf	19
12	Black Hat	"anomaly detection", "deep learning"	7
13	Google Scholar	"anomaly detection" "intrusion detection" security	18.400

Table 4. Search strings.

2.2.5. Inclusion and exclusion criteria

The first filter applied to papers returned by each source was based on title. Intrusion detection papers not leveraging anomaly detection techniques, or vice versa, were tagged as irrelevant. Next, same criteria was applied to abstract. Given that an abstract describes in greater detail the actual

research work performed, it allowed to further distinguish whether papers were or not under scope. Several research papers included "anomaly detection" or similar terms in their title but their abstract included references to supervised learning techniques, hence rendering them not relevant.

Next, full text was reviewed. Again, implementations combining anomaly detection with supervised learning or leveraging labeled security data were discarded. Exceptions were applied to anomaly detection techniques used in a hybrid context that were worth mentioning for some reason.

Additional relevant papers were found manually, using references of other papers, or using more *ad hoc* keyword searches. Particularly:

- Papers describing a general-purpose technique that is then used for intrusion detection in other paper. They provide context and insights to understand how other researchers applied that technique to the domain under scope.
- Papers related to feature extraction in an intrusion detection context. They are helpful to understand how relevant this process is to achieve acceptable performance levels.
- Papers documenting network, system or application intrusions datasets. They are useful to understand what sources researchers leverage in the IDS field.
- Papers related to topics included in the discussion. While these are not part of the core review, they provide direction and guidance on the most promising areas that must be considered in future research.

This review is based on both qualitative and quantitative studies, published from 1997 to date. Prior security research is based on data that might not be representative of current network and systems configurations, or account for the complexity of services and applications commonly found in computer networks today.

All included studies were written in English language.

Figure 1 depicts the exclusion process. Search results pointed to 5.120 total papers, downsized to 1.070 based on title, and 576 based on abstracts. These 576 papers were analyzed in detail to select a list of 240 papers.

Figure 1. Article assessment procedure.

2.2.6. Quality assessment

The studies have been assessed for internal and external validity of results, as well as bias.

2.2.7. Data extraction

The goal of the data extraction process was to answer questions in Table 1. Although it was not possible to answer all questions in all papers, the outcome, quality and relevance of the review was not negatively impacted. Each paper became highly relevant to answer certain questions, and so it made a relevant contribution to the overall review.

The procedure applied can be described as follows: one author reviewed all the studies and gathered data from them. For data extraction consistency, a second researcher took a random sample of papers and extracted data from them. Results were then checked for discrepancies. In those cases, consensus conversations among authors were used to resolve them.

2.3. Results

2.3.1. Proposed taxonomy

The techniques were classified into their underlying algorithms, in order to group together those that share the same foundational approach. Related techniques were compared as to assess what were their strengths and weaknesses, and whether results shared patterns or not (which would suggest that the algorithm was not key to obtain the results).

The resulting classification is shown in Figure 2.

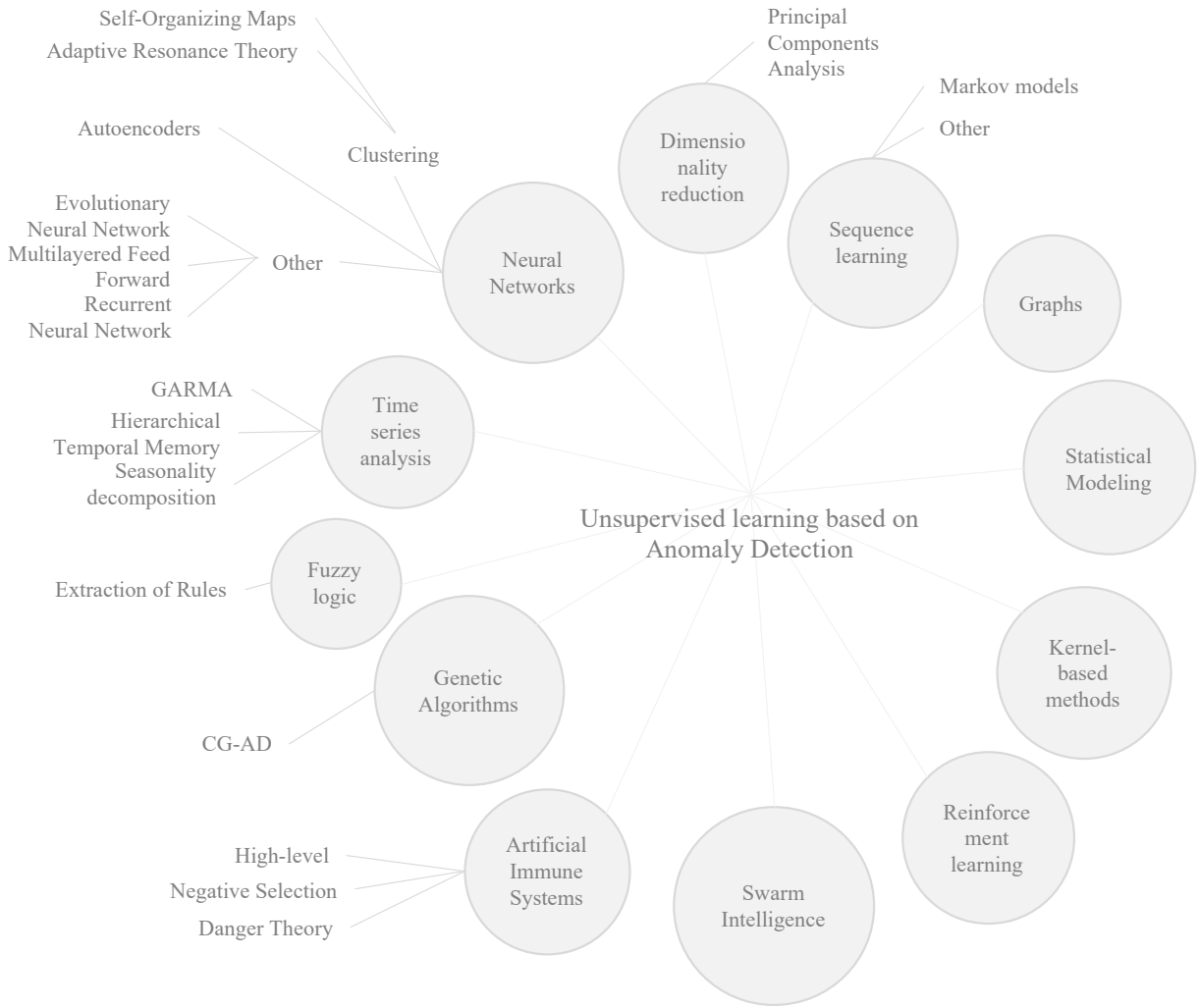

Figure 2. Proposed taxonomy.

2.3.2. Datasets, feature selection and performance evaluation

2.3.2.1. Datasets

From a general perspective, intrusion detection data is commonly collected from network behavior, user commands, system calls, log files, system error logs, or hardware consumption (Wu & Banzhaf, 2010). In practice, however, most intrusion detection implementations measured performance on KDDCup99 or NSL-KDD datasets (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017). These datasets are considered to be very large (Wu & Banzhaf, 2010), so 10% of it is frequently used. However, using sampled datasets conflicts with the big data nature of intrusion detection data.

On the other hand, artificially generated datasets might provide biased rates. Particularly, the false alarm rate could significantly differ depending on whether real or synthetic data is used (McHugh, 2000), and data structure, which varies for real and synthetic intrusion data, influences detection performance. An anomaly-detection algorithm was applied to 165 datasets with different structure which were calibrated and to which anomalies were injected. The conclusion was that performance differed as much as an order of magnitude (Maxion & Tan, 2000).

Also, several features in synthetic data (IP, time to live, TCP options and window size) have fixed and tight range, but a large range in real data (Mahoney & Chan, 2003b). The synthetic data doesn't include rare network packets, like those with bad checksums, garbage information in TCP fields that are not used, or IP fragmentation. Synthetic command-line data doesn't include malformed calls to applications, or wrong arguments.

Time range is also a key feature in intrusion data. Small ranges can lead to inaccurate behavior modeling, which can impact performance. Even with a high volume of traffic in data, the timespan can be a few minutes (Balthrop, Forrest, & Glickman, 2002), which is insufficient to characterize legitimate behavior.

Removing existing anomalies in training data when characterizing normal behavior also remains a challenge. With regards to this issue, a robust approach to removing anomalies from training data when building SOMs has been proposed (Rhodes, Mahaffey, & Cannady, 2000), in which several training sets train a map on each set, and then each set is filtered through other maps. This technique has not been leveraged by other researchers to account for the issue.

Several attempts have been made to create synthetic anomalous data to train classifiers in a pseudo-supervised fashion. In this approach, boundaries drawn to separate normal and anomalous data lead to more accurate detection of unknown attacks. Particularly, empirical studies (Fan, Miller, Stolfo, Lee, & Chan, 2004) showed that algorithms leveraging synthetic attacks detected more than 77% of the attacks not present in the training phase, and exhibited an accuracy beyond 50% per intrusion category.

Lastly, it’s remarkable that only recent studies used real network data.

2.3.2.2. Feature selection

The method used to select relevant features can heavily influence the performance of an IDS leveraging artificial intelligence (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017).

This is especially true for highly dimensional datasets, which are common in the intrusion detection space. The curse of dimensionality problem (Erfani, Rajasegarar, Karunasekera, & Leckie, 2016),

that arises when handling a high number of features, can pose a challenge to model legitimate behavior. Moreover, techniques based on clustering methods are more likely to be impacted by this issue, since they rely on identifying clusters of related feature values, which will not likely occur in multidimensional spaces.

2.3.2.3. Performance metrics and evaluation

Performance is typically measured by calculating detection accuracy and the ratio of false positives (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017).

The effectiveness of a technique is measured by the confusion matrix (Wu & Banzhaf, 2010), which contains figures for the following indicators:

- True Negative: normal traffic tagged as normal.
- True Positive: attack traffic tagged as attack.
- False Negative: attack traffic tagged as normal.
- False Positive: normal traffic tagged as attack.

The following metrics are built on top of the previous indicators to provide more accurate performance measures:

- True Negative Rate (TNR)
- True Positive Rate (TPR)
- False Negative Rate (FNR)
- False Positive Rate (FPR)
- Accuracy
- Precision

ROC (Receiver Operating Characteristic) curve can further assess IDS performance and compare the results obtained by different settings or systems (Wu & Banzhaf, 2010). It plots TPS on *y* axis and FPR on *x* axis, at single or multiple parameter settings. However, displaying true positives versus false positives doesn't allow researchers to understand the reasons for the obtained performance (McHugh, 2000).

2.3.3. Detection techniques

2.3.3.1. Neural networks

Neural networks are interconnected group of nodes which map inputs (observations) to an output, which is typically a class label in a predictive modeling exercise.

Neural networks are becoming popular in IDS research (Choksi, Shah, & Kale, 2014).

It has been demonstrated that they can operate on raw signals (data), hence automating feature extraction and making this process dynamic (Saxe & Berlin, 2017), which tackles the need in intrusion detection to fine tune features as attacks evolve. In fact, it was able to outperform manual feature extraction on the benchmark tests, leading to a 5%-10% increase in detection rate, keeping a low false alarm rate of 0.1%.

2.3.3.1.1. Clustering

Clustering techniques are unsupervised methods that discover groups of related data observations. They are typically used for intrusion detection by attempting to form two clusters of normal and

malicious data, or by forming multiple clusters of normal data and spotting data instances not fitting into any cluster, which are typically malicious.

Most popular clustering methods based on neural networks are self-organizing maps and Adaptive Resonance Theory. Generally speaking, SOM shows less overhead and ability to handle coordinated intrusion (Ahmad, Abdullah, & Alghamdi, 2010). ART has demonstrated to deliver a higher detection rate, low false negative rate, higher maturity, and more cost effectiveness (Ahmad, Abdullah, & Alghamdi, 2010).

2.3.3.1.1.1. Self-organizing maps

SOM implement clustering through a neural network. They can be used to spot unknown attacks. There is extensive research in the intrusion detection field that leverages this technique under several varying settings (Kumar & Radhakrishnan, 2014) (Sarasamma, Zhu, & Huff, 2005) (Rhodes, Mahaffey, & Cannady, 2000) (Ramadas, Ostermann, & Tjaden, 2003) (Amini, Jalili, & Shahriari, 2006).

One key advantage of SOMs is that they can easily filter out existing intrusions in training data used to profile normal behavior. With large datasets and the strength of the neighborhood function, a given SOM limits how much diversity can exist. As intrusions represent a small fraction of the training set, the map will be dominated by legitimate activity. In fact, an intrusion-detection system grounded on SOM competitive network achieved better detection and false positive rates than other approaches based on neural network (Kumar & Radhakrishnan, 2014). However, they can exhibit a low detection rate for certain attacks, like U2R and R2L (Choksi, Shah, & Kale, 2014).

Hierarchical or multi-layer SOMs are common for intrusion detection. In this setting, each layer uses only a few features from the set of available features. They can outperform single-layer SOMs in several attack types, and also increase overall detection rate (Sarasamma, Zhu, & Huff, 2005). In another research work (Kayacik, Zincir-Heywood, & Heywood, 2007), two-layer SOM obtained a 1.38% false positives rate and a 90.4% detection rate with all the features (41) available in the KDD dataset.

Moreover, when assigning a different network protocol to each layer, the ratio by which legitimate and malicious traffic differed was greater than an order of magnitude (Rhodes, Mahaffey, & Cannady, 2000). A similar setting (Ramadas, Ostermann, & Tjaden, 2003) assigning network services (HTTP, DNS and SMTP) to each layer obtained accurate results in three attack scenarios: buffer overflow in Sendmail, encapsulated HTTP traffic, and a buffer overflow in BIND. Extracted feature values for each connection were the following: interactivity (volume of requests per second), average size of questions (in bytes), average size of responses, idle question-answer time, idle time, and volume of connections.

Hyperellipsoidal SOM with Gaussian radial basis function as transfer function and winner-take-all approach (Sarasamma & Zhu, 2006) achieved detection rates from 91.55% to 91.71%, with false positive rate between 2.68% and 4.84%, when tested with KDD Cup 99. Parameters for the transfer function were not precalculated using the training dataset, but an accretion approach updates them as each data observation is processed. This introduces a slight overhead in execution time which is compensated by detection performance levels.

Another SOM implementation based on anomaly detection (Depren, Topallar, Anarim, & Ciliz, 2005) tested against KDD Cup 99 obtained a detection rate of 98.96% and a FPR of 1.01%. It

focused on the following features: service, duration, protocol, status flag, bytes sent and received. Three different SOMs were trained for each of the protocols (TCP, UDP, ICMP).

SOM has also been implemented in combination with other techniques, particularly hypothesis testing (Hoglund, Hatonen, & Sorvari, 2000) and Decision-Making systems (Miller & Inoue, 2003). In the first case, the Anomaly P-value of a data instance was the number of BMU (Best Mapping Unit) distances calculated in the training phase that were greater than the BMU distance for that data instance, divided by the total amount of training data instances. When that *p*-value exceeded a threshold, the data instance was flagged as attack. In the second case, the Decision-Making system made the final decision based on the reputation level of each SOM, based on past performance and implemented through Reinforcement Learning.

They have also been used to build a visualization tool (Girardin, 1999) for security analysts to discover network traffic anomalies, including intrusions (IP spoofing, FTP DoS, network scan and network hoping attacks), through topological classification of network events.

Lastly, SOMs can be used to preprocess data when building profiles of normal network packets in a hybrid system (Shon & Moon, 2007), leveraging GA for feature selection and Enhanced SVM for classification. Highly dimensional datasets are translated into bidimensional spaces, making it easier for Enhanced SVM to identify anomalous instances. This hybrid method outperformed signature-based systems Snort and Bro, achieving a detection rate of 99,99% and a FPR of 0,01% for one of the datasets used for testing.

2.3.3.1.1.2. ART

Adaptive Resonance Theory (ART) is similar to SOM, as it implements clustering through a neural network.

In Adaptive Resonance Theory technique, an input vector is transferred to its best matching neuron, which outputs a negative signal to each of the other neurons to inhibit their output (Carpenter & Grossberg, 1987a). Each neuron then represents a category or cluster to which input vectors are classified. Several variants were created, including one to handle continuous inputs (Carpenter & Grossberg, 1987b), another to support fuzzy logic (Carpenter, Grossberg, & Rosen, 1991).

They have also been applied to build an IDS (Chauhan, Pratap, & Dixit, 2015) (Amini, Jalili, & Shahriari, 2006). Compared to SOMs, with the same feature set (27 features generated from IP, TCP, UDP and ICMP headers), ART obtained a 97% accuracy, while SOM was slightly less accurate (95%) (Amini, Jalili, & Shahriari, 2006). When compared with FuzzyART (Durgin & Zhang, 2005), it was noted that the sensitivity of FuzzyART was much higher than that of SOM.

2.3.3.1.2. Autoencoder or replicator neural networks

An autoencoder is neural network with multiple layers which compresses and decompresses data in order to learn frequent patterns. As a result, legitimate activity is properly recognized but outliers (infrequent data) is not (Veeramachaneni, Arnaldo, Korrapati, Bassias, & Li, 2016). For example, a novel method for anomaly detection with deep autoencoders (Lyudchik, 2016) was used to recognize outliers (instances of digit '7') in the MNIST dataset of handwritten digits. 39 different models were tested. The model having the best performance, that is, having a high reconstruction error for an outlier and a low error for non-outliers, was the one with three hidden layers and six latent features.

Autoencoders have been successfully leveraged for anomaly detection in previous research (Zhou & Paffenroth, 2017) (Fan, Wen, Li, Qiu, Levine, & Xiao, 2020) (Borghesi, Bartolini, Lombardi,

Milano, & Benini, 2019) (Chen, Sathe, Aggarwal, & Turaga, 2017). They can even detect anomalies which linear techniques like PCA fail to spot (Sakurada & Yairi, 2014). For example, Kitsune (Mirsky, Doitshman, Elovici, & Shabtai, 2018) is an online IDS based on an ensemble of autoencoders which achieved a FPR of 0.1% with an average recall of 36,53%. The dataset used for testing was generated by launching attacks against a video surveillance network.

Autoencoders can also be applied to feature selection processes. The obtained feature set is supplied as input to a classifier. Using a self-taught learning technique (Javaid, Niyaz, Sun, & Alam, 2016) led to 88.39% accuracy, 85.44% precision, 95.95% recall, and 90.4% F-measure against the NSL-KDD dataset. A similar technique (Yu, Long & Cai, 2017) implementing stacked dilated convolutional autoencoders for pre-training stage, and a softmax classifier for fine-tuning led to 90.65% precision, 88.80% recall, and 88.64% F-measure. In similar setting (Yan & Han, 2018) with stacked sparse autoencoders and SVM, authors reported 99.35% accuracy, 99.01% detection rate and 0.13% FPR.

Another research (Ludwig, 2017) combined unsupervised and supervised settings under an ensemble of the following algorithms: deep belief neural network, autoencoder, deep neural network, and extreme learning machine. It achieved 93% precision, 93% recall and 92% F-score against the NSL-KDD dataset. A similar technique (Yousefi-Azar, Varadharajan, Hamey, & Tupakula, 2017), tested against KDD dataset, achieved 83.84% accuracy. The Taguchi method has also been integrated to stacked sparse autoencoders working as feature extractors (Karim, Güzel, Tolun, Kaya, & Çelebi, 2018). Tested against the UNSW-NB15 dataset, it achieved 99.7% precision and 99.7% accuracy.

Non-symmetric deep autoencoders have been leveraged for intrusion detection in combination with Random Forest. The autoencoder helped reduce the dimensionality of inputs without impacting detection performance (Shone, Ngoc, Phai, & Shi, 2018). Obtained results led to 97.8% accuracy, 99.99% precision, 97.85% recall, 98.15% F-score and 2.15% False Alarm Rate.

Autoencoders (Alam et al, 2019), in this case implemented using memristors, were tested against the NSL-KDD dataset, leading to an accuracy of 92.91%. Similar research (Khan, Gumaei, Derhab, & Hussain, 2019), implementing a novel deep learning model with two stages, obtained 89.13% accuracy. Yet another research using autoencoders (Choi, Kim, Lee, & Kim, 2019) achieved 91.70% accuracy.

Lastly, autoencoders' reconstruction error has been leveraged for intrusion detection (Sun, Wang, Xiong, & Shao, 2018 (Meidan, Bohadana, Mathov, Mirsky, Shabtai, Breitenbacher, & Elovici, 2018). Tested against KDD Cup 99 (Sun, Wang, Xiong, & Shao, 2018, it achieved an F-score of 81.20%. When applied to botnet detection in IoT devices (Meidan, Bohadana, Mathov, Mirsky, Shabtai, Breitenbacher, & Elovici, 2018), it achieved a TPR of 100% and a FPR of 0.007±0.01.

2.3.3.1.3. Other neural network approaches

Evolutionary neural network (ENN) (Han & Cho, 2006) produced 0.0011% false positive rate at 100% detection rate when tested system-call sequences in the BSM audit data of the IDEVAL dataset. ENNs have the advantage that both the weights and the network topology are learnt automatically, with no human intervention.

Multilayered Feed Forward (MLFF) neural network (Tan, 1995) successfully identified abnormal commands, connection times, connections to hosts, services and programs, when trained with a set of UNIX users' commands, designed to capture their behavior. In a similar setting (Ryan, Lin, &

Miikkulainen, 1998), a detection rate of 96% with a FPR of 7% was obtained. However, anomalous behavior was generated randomly, and so might not be representative of a real, more subtle, change in behavior to perform malicious actions.

Recurrent Neural Network (RNN) were used to learn sequences of legitimate user actions, from login to logout, while excluding anomalies with a rule-based system (Debar, Becker, & Siboni, 1992). It then predicted most probable next commands, allowing to recognize captured behavior through successful predictions, and anomalous behavior otherwise.

Lastly, a semisupervised approach (Ghosh, Wanken, & Charron, 1998) trained a neural network with legitimate, random, and well-known malicious patterns for the *lpr* UNIX command. Highest detection rates were achieved with random inputs, which suggest that the use of this kind of inputs can potentially help in detecting unknown attacks.

2.3.3.2. Fuzzy logic

In this variant of traditional logic, the truth values of features are real numbers, instead of binary values. It's applied to use cases where there is vagueness and uncertainty, as it's the case for intrusion detection.

When used in the form of fuzzy association rules mined during the training phase with normal network traffic (Bridges & Vaughn, 2000), it was able to recognize IP spoofing and port scanning attacks. These rules are then compared with those obtained when mining test data that contains anomalies. The similarity between both sets of rules quantifies the anomaly score. In this setting, extracted features were the amount of SYN, FIN, RST flags in TCP header, and the number of distinct destination ports during last 2 seconds.

In a similar setting (Linda, Manic, Vollmer, & Wright, 2011), fuzzy rules were extracted from a previous clustering exercise that identified clusters of normal network traffic (learning directly from the stream of network packets). Fuzzy rules extraction was done through a fuzzy membership function (nonsymmetrical Gaussian) with different standard deviations for right and left sides. Classification was made based on maximum t-conorm to rules output, which is related to the degree of membership to a normal behavior cluster. Tests were performed against a dataset of 200.000 packets containing anomalous network activity, achieving 99.36% correct classification rate with 0.0% FPR and 0.9% FNR.

2.3.3.3. Genetic algorithms

This type of algorithm is a metaheuristic that resembles a natural selection process. They are used to discover high-quality solutions in search problems by leveraging operators such as mutation, crossover and selection.

When combined with an unsupervised clustering approach (Aissa & Guerroumi, 2015), a GA achieved detection rates from 75% to 98%, and FPR from 1.3% to 0.12% against different subsets of KDD99 dataset. This technique created a cluster of rejected instances, including those not meeting the confidence interval set for the fitness function. This approach, called CG-AD, turned out to be more efficient than K-means.

In another setting (Shon, Kim, Lee, & Moon, 2005), a GA was used to perform optimal feature selection from 24 attributes representing TCP and IP header fields. The fitness function leveraged anomaly and communication scores defined in MIT Lincoln Lab datasets (Shon, Kim, Lee, & Moon,

2005). The feature sets obtained in the final generations (91-100, 15 out of 24 features selected) led to best detection rate (97,56%) and the lowest FPR (0%) when feeding an SVM-based IDS.

2.3.3.4. Artificial Immune Systems

2.3.3.4.1. Introduction

Artificial Immune Systems (AIS) abstract human immune system and apply this abstraction to solving problems with computer systems. AIS design is suitable for intrusion detection due to these features (Kim, Bentley, Aickelin, Greensmith, Tedesco, & Twycross, 2007): distributed, disposable and multi-layered, as well as diverse, all of which increases robustness; self-organized, which provides adaptability; lightweight, which increases efficiency.

The first implementation of an AIS-based intrusion detection system (Kephart, 1994) used six decoy programs to entice the potential virus to infect them. If that occurred, an algorithmic analysis of the infected decoy extracted relevant signatures by executing it with a debugger. These signatures were tested against a set of legitimate programs to avoid false positives. If tests were passed, the newly created signatures were added to the database of well-known viruses.

On the other hand, a conceptual AIS-based multi-agent system (Dasgupta, 1999) made use of agents with different monitoring levels (user, system, process, packet) and Action/Decision agents to take action when an anomaly was detected, based on previously learnt normal behavior. While useful as a general framework, it doesn't provide any implementation insights. Another AIS-based high-level IDS (de Paula, de Castro, & de Geus, 2004) proposed several detection mechanisms resembling Human Immune System (HIS): evidence-based detector (policy violation), behavioral-based detector, signature extractor and knowledge base detector (misuse detection).

2.3.3.4.2. Negative selection

In negative selection (Forrest, Perelson, Allen, & Cherukuri, 1994), detectors are set up for a set of randomly generated non-self strings, representing patterns. If current behavior of one of the monitored entities matches one of those patterns, then that behavior is flagged as an anomaly. One of the most appealing features of negative selection, from an anomaly detection standpoint, is that no prior knowledge of intrusions is required, and so unknown attacks can be spotted (D'haeseleer, 1996). However, the following factors influence performance: number of detectors, non-self space covered by each detector, and storage needed for the detector. Also, holes (non-self space that can't be matched by any detector), constitute the biggest risk of this approach for intrusion detection.

An IDS based on Negative Selection (Hofmeyr, 1999) showed high performance when tested against probing a port scanning exercises, and random connections. It generated self and non-self strings from TCP data paths (source IP, destination IP, and service). It implemented several features to make it more suitable for intrusion detection: more than a single representation, which helps to avoid holes; reducing false alarms with sensitivity levels and activation thresholds; co-stimulation by a human, which allows to avoid autoreactive detectors; generation of detections in a distributed fashion, which allows to adapt to self sets that change; detectors of dynamic nature, which helps to increment detection rates; and memory, so that detection based on signatures is also available.

When tested with 28 fields of TCP headers and 16 fields of UDP/ICMP headers, high detection rates and low error rates were achieved (Kim, Bentley, Aickelin, Greensmith, Tedesco, & Twycross, 2007). However, these results might be influenced by the use of simulated (and not real) intrusions.

Negative Selection poses two additional issues: scalability and coverage. To obtain a detection rate of 80%, more than 643 million of detectors are required (Kim & Bentley, 2001a), and 1.000 detectors are generated in 20 hours (these figures might be now obsolete, considering that more computational power is available nowadays).

Dynamic Clonal Selection was added to the Negative Selection approach (Kim & Bentley, 2002) to mitigate the scalability issue, by capturing normal behavior using just a subset of self-antigens. Clonal Selection also increased detection rates of well-known attacks when used in a supervised approach (Kim & Bentley, 1999). In this setting, the primary IDS is helped by a secondary IDS that clones best performing detector sets, which makes it more specialized on most probable non-self antigens. The drawback of this approach is that it reduces the chance to detect novel attacks. Lastly, in another Clonal Selection implementation (Kim & Bentley, 2001b), detectors evolved towards the non-self patterns hidden in the collected non-self data, which essentially transforms the anomaly detection approach into a misuse detector.

As for coverage, the permutation mask technique (Hofmeyr & Forrest, 1999) can significantly reduce holes. It consists of adding detectors with multiple representations (through permutation) but identical non-self space. The union of coverages of all detectors would likely reduce the resulting number of holes.

Also, a Negative Selection variant for continuous data (Randomized Real-Valued Negative Selection) (Gonzalez, Dasgupta, & Niño, 2003) attempted to maximize coverage. However, when benchmarked (Stibor, Timmis, & Eckert, 2005) against three other algorithms (Real-Valued Positive Selection, OneClassSVM, and Parzen-Window) with KDD Cup 99 data, it was discovered that it was not appropriate for high-dimensional datasets as those found in the intrusion detection field. In another setting (González & Dasgupta, 2003), abnormal samples representing the non-self space were produced, while normal samples defined the self space. Both sets of samples were supplied as input to train a MLP neural network classifier. Tested with MIT-DARPA 98 and MIT-DARPA 99 datasets and compared with a SOM-based implementation, this method showed equivalent detection rate but with a high FPR. Lastly, the effectiveness of this technique has been questioned (Stibor, Mohr, Timmis, & Eckert, 2005) in unsupervised settings, as its performance heavily depends on the self-radius parameter, which can only be accurately calculated when instances from the anomalous class are available in the training dataset.

When combined with fuzzy rules (Gómez, González, & Dasgupta, 2003), the results obtained were a 98,22% detection rate with a 1.9% FPR against KDD Cup 99 dataset, and a 94,63% detection rate for the DARPA 99 dataset, outperforming two other similar algorithms: Evolving Rule Detectors and Parallel Hill Climbing of Fuzzy Rules Detectors.

Lastly, when combining Real-Valued Negative Selection with a MLP (Gonzalez & Dasgupta, 2002), results were comparable to those obtained with a SOM. Moreover, a higher detection rate could be achieved if a higher false positive rate was allowed.

2.3.3.4.3. Danger theory

Danger Theory is an evolution of the Negative Selection algorithm in which danger signals (unusual death of self-cells in the HIS context) are the triggers for immune response. The concept of danger signals is subject to interpretation when applied to network intrusion.

An implementation combining elements of Danger Theory as artificial tissue, dentritic cells algorithm and T-cell algorithm was able to detect malicious code execution (Kim, Greensmith, Twycross, & Aickelin, 2010). This system required PAMPS (Pathogen Associated Molecular Patterns), that is, known security policy violations from *systrace* program. This requirement suggests that this technique is highly dependent upon existing knowledge about intrusion and hence couldn't be classified as unsupervised. Another setting (Greensmith, Twycross, & Aickelin, 2006) leveraging Dentritic Cells successfully detected ICMP scans through the high concentration of PAMPS (destination unreachable errors) and danger signals.

An implementation based on lymphotic laws (Hashim, Munasinghe, & Jamalipour, 2010) effectively detected DoS, DDoS and worm attacks. Detection is based on three consecutive danger signals: initiation process, recognition process and co-stimulation process. First signal is triggered when network traffic deviating from normal profile is found. Second signal detects malicious anomalies present in the deviated traffic through a flow-level spectral analysis. Third signal estimates the significance level of the anomaly (to be considered an attack) through the likelihood-ratio of the traffic deviation.

2.3.3.5. Swarm Intelligence

Swarm Intelligence techniques are inspired by collective behavior observed in decentralized systems like those found in social insects or swarms. The aim of these techniques is to implement simple solutions for complex problems.

A clustering algorithm based on ant colonies approach to self-organization (Ramos & Abraham, 2005) was compared with Decision Trees, SVM and Linear Genetic Programming (LGP) leveraging KDD Cup 99 data and obtained notable results. In this approach, a set of agents move around the classification habitat using available information about pheromone concentrations. Also, a similar approach (Feng, Wu, Wu, Xiong, & Zhou, 2005) (Feng, Zhong, Xiong, Ye, & Wu, 2007) (Feng, Zhong, Ye, & Wu, 2006) based on advanced distance-based metrics (different from those used by most clustering techniques) claimed to successfully detect unknown intrusions.

2.3.3.6. Graphs

Graphs are data structures with finite set of nodes and pairs (edges) of these nodes. Graphs are used in multiple domains and use cases and are suitable for intrusion detection given the relational nature of their features (Veeramachaneni, Arnaldo, Korrapati, Bassias, & Li, 2016).

Link prediction is a graph-based approach in which the anomaly score of a link between two nodes can be obtained through algorithms like preferential attachment, spreading activation, or generative models (Huang & Zeng, 2006). It successfully assigned the highest anomaly score to five synthetically generated fake e-mails inserted into the Enron e-mail corpus.

A graph-based behavior modeling approach for use with system call sequences of a program (Gao, Reiter, & Song, 2004) was tested against common UNIX programs like *proftpd, wu-ftpd*, *httpd*, and *httpd* with *chroot*, obtaining promising results. The system built an execution graph that only accepted sequences aligned with the program's control flow graph. This graph contained all possible executions, therefore representing legitimate behavior.

Anomalous Substructure Detection and Anomalous Subgraph Detection (Cook & Holder, 2000) techniques were tested against 37 attack types of KDD Cup 99 data (Noble & Cook, 2003). The first one examined the entire graph and reported unusual substructures, while the latter partitioned the

graph into subgraphs and determined how anomalous they were when compared to the rest. They both assigned a 'normality score' below a reasonable threshold to most attacks. In both cases, *snmpgetattack* and *snmpguess* attacks couldn't be detected, which suggests that further testing is required.

2.3.3.7. Sequence learning

Most sequence learning approaches rely on Markov models or Hidden Markov Models. However, other researchers have also been successful in this task by leveraging neural networks, instance-based learning or even custom algorithms.

2.3.3.7.1. Markov models

Markov models have been used to profile legitimate system behavior from training data (Ye, 2000). Current system behavior is then analyzed to measure the probability that the model supports observed behavior.

HMMs (Hidden Markov Models), a Markov model variant, has shown high discrimination power for system call sequences data and low discrimination power for shell command sequences (Yeung & Ding, 2003). This can be due to the fact that temporal relationships are key in the first scenario, but harmful in the latter. When tested against user shell activity (UNIX) (Lane & Brodley, 2003), similar conclusions were reached. Also, it was discovered that sequence length and number of states heavily influenced performance: longer sequences and more states led to better detection rate. However, results were not uniform across programs (ps, login, named and sendmail), which suggests that the underlying structure of the program must also be considered in the modeling process. The technique was further improved (Kruegel, Kirda, Mutz, Robertson, & Vigna, 2005) and obtained satisfactory results when tested against three dummy vulnerable programs and three real-world applications (apache2, ftpd, imapd).

HMMs performance has been subject to improvement (Khreich, Granger, Sabourin, & Miri, 2009) by using a multi-HMM system in which each model was built with varying number of hidden states. The resulting models were combined with the Maximum Realizable ROC (MRROC). When applied to synthetic HIDS data, obtained performance was higher than the highest obtained with a single HMM or STIDE (sequence matching).

When combined with a Naïve Bayes Classifier (that can properly handle the skewness in network traffic) (Karthick, Hattiwale, & Ravindran, 2012), an HMM with more than five states had 100% accuracy classifying all IPs from CAIDA (Hick, Aben, Claffy, & Polterock, 2007) as attack and IPs from DARPA (“MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation”, n.d.) as clean.

With HsMM (Hidden Semi Markov Models) (Xie & Yu, 2008), the optimal configuration achieved a detection rate of 97% with 1,8% FPR when detecting application-level DDoS attacks in HTTP session data. HsMM is an extension of HMM with explicit state duration, which makes it suitable to model user HTTP sessions. Deviation from the mean entropy of training data measured the abnormality of an observed request sequence made by a given user.

Lastly, combined with the Urn and Ball model (Rabiner, 1989), promising results were obtained in detecting whether a TCP session was anomalous or not (Joshi & Phoha, 2005). Urns were 5 features (from 41 features) of KDD Cup 99: *duration*, *is_host_login, src_bytes*, *dst_bytes*, and *is_guest_login*.

2.3.3.7.2. Other sequence learning approaches

Recurrent Neural Networks (RNNs) are neural networks which can learn sequences. When applied to intrusion detection (Al-Subaie & Zulkernine, 2007), they detected novel attacks and recognized new normal activity. They also reduce FPR and FNR, and generalize better than other IDS.

By combining sequence learning with dimensionality reduction (Oka, Oyama, Abe, & Kato, 2004), a 72.3% detection rate with a 2.5% FPR was obtained for anomalous sequences of user commands. In this approach, a user profiles were built by modeling the sequence' dynamic features, extracting principal features of the model, and producing a layered network from them.

On the other hand, using a novel similarity measure (Lane & Brodley, 1997b) that counted adjacent matching tokens (strings), 66.177 tokens from 4 different users were successfully classified: users were similar to themselves and dissimilar to the other users. The set of sequences extracted in the training phase were compared against those obtained in the detection phase. The similarity measures of the set of sequences for a given user were aggregated through a smoothing filter, and the result was leveraged to classify the overall stream as normal or unusual.

Using Instance-based learning (IBL) (Lane & Brodley, 1999) (Lane & Brodley, 1997a), a sequence learning approach in which temporal sequences of categorical observations are translated into a metric space through a similarity measure, users with a different behavior were clearly distinguished from profiled users, at an early stage. Tests were performed with UNIX user command-line data. These researchers later assessed the impact of concept drift (that occurs when statistical properties of features being modeled change over time) in user behavior modeling (Lane & Brodley, 1998). They compared three techniques (DAIP, *truncate*, U-ins). DAIP matched or outperformed *truncate* in 52% of true accept tests but performed worst on true detect nearly 85% of the time. The disparity was greater for U-ins, which was superior to *truncate* 60% of the time on true accept but worst 90% of the time on true detect tests.

An Extended Finite State Automata (EFSA) that captured normal behavior of hosts and routers in a network (Sekar, Gupta, Frullo, Shanbhag, Tiwari, Yang, & Zhou, 2002) was tested against the DARPA 99 dataset, detecting all probing and DoS attacks with less than 10 false alarms per day. The mapping process modeled how often a transition was taken, and common values of state variables on a transition. These statistics were then recalculated in the detection phase. Anomalies were flagged when a significant deviation was found.

Incremental Probabilistic Action Modeling (IPAM) (Davison & Hirsh, 1998) predicted the next command in a sequence of UNIX commands entered by a user for which a normal profile was built in advance. Compared to C4.5 classifier over 77 users, it outperformed C4.5 sixty times, tied once, and lost sixteen times. This algorithm worked under the assumption that recent activity influences future activity more than old activity, much in the same way as Markov models do.

Lastly, Bayesian parameter estimation (Singer, 1999) detected 91% of all the Whisker (web vulnerability scanner) scans and previously unseen sequences of web pages requested by users (Cho & Cha, 2004). Attacks were detected by comparing real-time web logs with expected frequencies of the resulting sequences, with the underlying assumption was that sequences of pages requested by users will exhibit patterns.

2.3.3.8. Time-series analysis

Time series analysis has been consistently applied to domains other than intrusion detection for forecasting purposes. In particular, ARMA and its variants have been the preferred statistical models to perform time series analysis.

GARMA models, which include generalized responses in the exponential family (Poisson counts, binary responses), have been successfully used to forecast cyberattacks (Pillai, Palaniappan, Abdullah, & Imran, 2015), with acceptable accuracy and sufficient early-warning time to defeat these attacks.

Hierarchical Temporal Memory (HTM) has been used to detect anomalies in traffic received from real-time sensors (Ahmad, Lavin, Purdy, & Agha, 2017), outperforming other anomaly detection algorithms like CAD OSE, KNN CAD, Relative Entropy, Twitter AdVec, Skyline and Sliding Threshold on some but not all of the tests. HTM is based on an online sequence memory algorithm. In this approach, the prediction error or anomaly score is the gap between predicted sequence and current input.

2.3.3.9. Reinforcement learning

In this type of learning, agents operate in an environment and take actions to maximize the cumulative reward. This method has also been applied to the intrusion detection field.

An adaptive neural network (Cerebellar Model Articulation Controller or CMAC) using RL for DoS (denial of service) detection (Cannady, 2000) achieved a response error of 1.94-07% for Ping Flood attacks and a response error of 8.53-14% for a type of attack for which hadn't been trained (UDP Packet Storm). The reinforcement learning component was implemented by providing feedback to the neural network from the protected system, from a number of system state-related indicators (CPU load, available memory, network load, etc.).

A log correlation and association rule learning IDS used RL to increase (reward) or decrease (penalty) the importance of the rules that correctly pointed to log files containing attack patterns (Deokar & Hazarnis, 2012). Performance was not made available.

Temporal-difference (TD) learning has been applied to sequential anomaly detection and tested against system-call sequences to spot multi-stage intrusions like buffer overflows, symlink attacks and Trojan programs (Xu, 2010). It outperformed HMM and supervised techniques like SVM and Naïve Bayes. This approach was based on a novel Markov reward model in which the learning prediction of value functions was equivalent to estimating the anomaly probabilities of the data sequences. TD learning algorithm was also tested (Xu, 2006) against KDD99 data and the MIT *lpr* system call data, confirming the suitability of this approach for intrusion detection.

RL has also been used to minimize the information exchange volumes of IDS agents in a distributed architecture, and to improve the detection rates (Xu, Sun, & Huang, 2007). In this approach, observed sequences of source IP addresses were only shared with other agents if related to a DDoS attack.

2.3.3.10. Dimensionality reduction

In Principal Components Analysis (PCA), features are projected to a space with principal components. This projection is then reversed to recover the features from those principal components (Veeramachaneni, Arnaldo, Korrapati, Bassias, & Li, 2016). If only the first principal components

(those explaining most variance in data) are used for projection and reconstruction, reconstruction error will be low for most observations and high for outliers.

When tested against system calls in a dataset made available by the University of New Mexico and a UNIX command-line dataset produced by AT&T Research Lab (Wang, Guan, & Zhang, 2004), it obtained a 100% detection rate with a FPR of 2.75% for system calls, and 100% detection rate with 0% FPR for UNIX command data. The algorithm measured the distance between the original vector and its projection, with a long distance representing unusual user or program behavior. Another similar approach (using the distance between major and minor principal components to the mean of the sample as anomaly score) (Shyu, Chen, Sarinnapakorn, & Chang, 2003) obtained 98.94% recall and a precision of 97.89%, with FPR of 0.92% against KDD Cup 99.

Abnormal traffic subspace, another PCA-based technique, is characterized by the (*n*-*k*) remaining Principal Components (PC). In this setting, *n* is the total number of PCs and the k first Principal Components represent the normal traffic subspace. This approach was useful to identify volume-related anomalies (Huang, Nguyen, Garofalakis, Jordan, Joseph, & Taft, 2006), which constitute a subset of the network intrusion categories.

Lastly, multiscale streaming PCA and hierarchical streaming PCA added real-time anomaly detection (Kiran, 2017), a very convenient feature for a network intrusion detection system.

2.3.3.11. Clustering

This section includes clustering techniques not based on neural networks.

A variant of the single-linkage clustering technique (Portnoy, 2000) obtained a maximum detection rate of 88% with 8,14% FPR, when tested with KDD Cup 99. The algorithm was trained with normal distances and identified clusters of normal and anomalous instances, with no feature selection, transformation or extraction process.

By leveraging the first phase of BIRCH hierarchical clustering algorithm (Burbeck & Nadjm-Tehrani, 2004), a detection rate of 95% and a FPR of 2.8% were achieved against KDD Cup 99 dataset. The clusters hierarchy was built through incremental training. The detection process consisted of searching from the root the closest cluster feature. The distance from the centroid to the new data point was computed, with a high distance revealing an anomaly.

Frequent baskets concept (Leung & Leckie, 2005) achieved a ROC curve of 0.867, comparable to other algorithms like KNN or SVM, when tested against KDD Cup 99. Data instances were transformed into sequences of frequent baskets, used to generate the FP-tree with the FP-growth algorithm, and clusters were then extracted using the count back method.

A novel clustering algorithm (Last, Shapira, Elovici, Zaafrany, & Kandel, 2003) compactly represented and classified the content of web pages browsed by users in order to identify when a user performs abnormal activity. Tested against 1,000 vectors representing normal accesses and 100 vectors representing abnormal accesses, TPR was 0.7 and FPR was 0.008.

Transductive Confidence Machines for KNN (TCM-KNN) (Li, Fang, Guo, & Chen, 2007) achieved a detection rate of 99.48% with 1,74% FPR against KDD Cup 99 with no feature selection, and 99,32% and 2,81%, respectively, with a subset of the features. In this approach, p-value associated with a strangeness ratio was calculated to identify anomalous observations.

Lastly, the performance of three additional clustering methods applied to intrusion detection were compared (Syarif, Prugel-Bennett, & Wills, 2012): distance-based outlier detection approach led to 80.15% accuracy, while EM clustering led to 78.06%, and K-Medoids led to 76.71%.

2.3.3.12. Multivariate outliers detection

OneClassSVM is a variant of the most recent SVM implementation (Cortes & Vapnik, 1995). It discovers a region where most training data is located, and labels those observations as one class. Observations outside the region are labeled as another class.

Tested against 1998 DARPA Intrusion Detection Evaluation Data (Lazarevic, Ertoz, Kumar, Ozgur, & Srivastava, 2003), it achieved the best detection rate (84,2%) but also a high FPR (4%), compared to Density-Based Local Outliers (LOF), Nearest Neighbors (NN), and Mahalanobis Distance Outliers Detection. In this approach, the following types of features were produced from *tcpdump* files: connection-based, content-based, and time-based, most of them being simple counts and sums.

Tested against KDD Cup 99 dataset (Eskin, Arnold, Prerau, Portnoy, & Stolfo, 2002), it achieved 98% detection rate with a 10% FPR, and 100% detection rate when evaluated against system-call sequences, which suggests that this method might outperform HMM and other sequence learning methods.

Lastly, two variants attempting to make OneClassSVM more robust to outliers (Amer, Goldstein, & Abdennadher, 2013) outperformed other standard unsupervised anomaly detection techniques in two of four datasets of UCI Machine Learning Repository. Both of them aimed at reducing the weight of outliers in computing the decision boundary.

On the other hand, a kernel-based method that projected raw network traffic feature space into a highly dimensional feature space (Ahmed, Coates, & Lakhina, 2007) obtained results comparable to those of PCA-based methods. In this approach, the kernel trick (inner product of feature vectors) was used to produce a dictionary representing the region of normality through two different thresholds.

2.3.3.13. Association Rule Learning

Association rule learning (ARL) finds patterns and relations in data. It's a machine learning technique. As opposed to sequence learning, it does not account for the order of items.

When implemented to isolate rare events in nominal time-series data (Mahoney & Chan, 2003a), it achieved a 64% detection rate at a threshold of 100 false positives against the IDEVAL dataset. The approach consisted of finding conditional rules to model legitimate patterns in a time series of tuples of nominal attributes (packet fields values, elements in an HTTP request or strings in a TCP session). An anomaly score was then produced based on the time since the rule was last triggered, the rule's support, and the number of allowed values for the attribute.

When applied to modeling legitimate user command-line calls (Lee, Stolfo, & Mok, 1999), it was able to accurately detect anomalies in a subset of DARPA datasets. These anomalies had very low similarity scores (outside the range of legitimate behavior), which represented the likelihood that the current user behavior is aligned with recorded profile.

More complex settings (Vaccaro, 1988) combined rule extraction and an evaluation process to model normal user/system behavior, given by a set of a attributes. A rule forest was generated with no a priori templates. Each rule was assigned a grade, based on its historical accuracy. An anomaly score was then calculated in the detection phase for each new transaction. Particularly, a score was

obtained for each attribute, aggregated to obtain a transaction (set of attributes) score, and further aggregated to obtain a thread score (set of transactions for a given user/system). Each individual score was based on the number of rules fired and violated, and their grades.

The Frequent Episodes Rules (FER) (Qin & Hwang, 2004a) was leveraged to enhance Association Rule Learning detection rates (Qin & Hwang, 2004b), with tests against NetAttack dataset achieving a detection rate of 47% for the following types of attack: DoS, R2L and portscan. This represents an average of 51% gain versus association rules. As opposed to association rules, which identifies relationships among attributes in a single connection, FER identify relationships among multiple connection records in a sequence. The underlying assumption justifying this approach is that sequence-related information is relevant to uncover attacks.

Association Rule Learning has also been leveraged to extract rules from network traffic (Chan, Mahoney, & Arshad, 2003), which detected 117 attacks out of 201 in the DARPA 99 dataset (with 50 to 75 rules and 10 false alarms per day). In this approach, the goal was to identify rules with high support along the training phase, and violations of those rules along the detection phase. A novel clustering algorithm was also implemented to discover outliers along the training phase, so that the data used in this phase could be considered attack-free. A similar approach (Chen, Lu, & Teng, 1990) extracted rules from user activity, through Time-based Inductive Machine (TIM) algorithm (Chen, 1988), which accounts for the sequential order of the events in the antecedent of a rule.

2.3.3.14. Other statistical modeling methods

PAYL (Wang & Stolfo, 2004) is a technique that profiles payload byte frequency distribution and leverages Mahalanobis distance to measure similarity with expected distribution. Nearly 100% accuracy was achieved with 0.1% FPR for http, when tested against the 1999 DARPA IDS and data made available by Columbia CS department network. However, PAYL can be defeated with *polymorphic blending attacks*, that successfully evaded byte frequency-based detection systems by mimicking the statistical profile of normal behavior (Fogla, Sharif, Perdisci, Kolesnikov, & Lee, 2006). An enhanced version developed later (Perdisci, Gu, & Lee, 2006) was less vulnerable to these attacks. This new version generated descriptions of the payload in multiple feature spaces with related features, whose relationships were identified through a clustering algorithm designed for text classification (Dhillon, Mallela, & Kumar, 2003).

A variant of PAYL (Bolzoni, Etalle, & Hartel, 2006) with a two-tier architecture achieved a higher detection rate and a lower FPR than PAYL, and PHAD (Mahoney & Chan, 2002) (Mahoney & Chan, 2001), another payload-based anomaly detector. The first layer implemented a SOM to pre-process data, and the second layer implemented PAYL to perform classification. SOM was found to be a useful method to pre-classifying packets. The assigned cluster then became an input to PAYL.

K-means was leveraged to produce clusters of normal and anomalous traffic (Münz, Li, & Carle, 2007), which led to successful detection of port scans and DoS attacks against both synthetic and real data. The algorithm was independently run for each of the services observed in the traffic, assuming that different protocols would exhibit different normal behavior profiles. Extracted features included number of packets, bytes and source-destination pairs per flow.

Conditional probabilities were used (Stolfo, Hershkop, Bui, Ferster, & Wang, 2005) to detect anomalous file system accesses through the following features: user ID, working directory, command-line invoking the process, parent directory, filename, the three previously accessed files, and file access frequency. One order consistency check calculated the probability of a feature value,

and another calculated the probability of a feature value conditioned on another feature. This system achieved 95% detection rate and a 2% FPR.

Histograms have also been leveraged (Kind, Stoecklin, & Dimitropoulos, 2009) to represent flows for a set of features (source and target IP and port, TCP flags, protocol, packet size, and duration). When recalculated in the detection phase, those exceeding a distance to the clusters found in the training phase represented anomalies. When tested against 15 types of attacks of the IDEVAL dataset, 13 alarms were raised (86.7% accuracy).

Another statistical approach (Ye, & Chen, 2001) was based on a statistic called X^2, equivalent to Hotelling's T^2 statistic, but less computationally expensive. A sample of audit data for normal events (with 284 event types and a stream of 3019 events) from MIT Lincoln Laboratory was used for testing. The detection rate by audit event was 75% and reached 100% for intrusion session (password guessing, symlinks attacks, and unauthorized remote access). 71% of the intrusion sessions are detected at the first audit event. In a similar approach (Zhang, Li, Manikopoulos, Jorgenson, & Ucles, 2001), the resulting statistical model was used as input to a neural network-based classifier. Yet another equivalent approach (Krügel, Toth, & Kirda, 2002) produced an anomaly score from three metrics (probability distribution of request length, character distribution of payload data in the request, and likelihood of a given request type). It achieved 100% detection rate against DNS and HTTP data with six different attack types.

Markov models and statistical anomalies were combined (Kruegel & Vigna, 2003) to detect malicious HTTP requests, using the following features: length, character distribution, attribute presence or absence, attribute order, and structural inference (for which the Markov model was used). Three datasets with webserver logs were used for testing. Collectively, the models were able to detect all intrusion scenarios (buffer overflow, directory traversal, cross-site scripting, input validation and Code Red).

Another technique successfully detected SYN flood attacks using Latent Dirichlet Allocation (Newton, 2012) in TCP/IP header data. The dataset used was made available by the University of North Carolina at Chapel Hill (UNC). IP address sessions were translated into documents, with words being target IP address/port number combinations. The resulting data was run through LDA to create two models in the training and detection phases, which were then compared using variational inference (log likelihood of documents in new data). A likelihood below the selected threshold was associated with an anomaly.

Random forests were implemented under an unsupervised setting to build a classifier that was able to find service-related patterns and match each data instance with the right service (Zhang & Zulkernine, 2006). Data instances for which a class couldn't be matched were considered anomalies. The method was tested against the KDD Cup 99 and achieved a 95% detection rate of with a 1% FPR. However, if false alarm rate was adjusted to 0.1%, detection rate dropped to 60%.

Lastly, entropy measurement has been used for worm detection (Wagner & Plattner, 2005), by assuming that worm traffic is more uniform than normal traffic, and target IP addresses seen in flows are more random than those in normal traffic. Entropy was obtained through data compression. The method was successfully tested against Blaster and Witty worms. Authors warn, though, that this method is not suitable for slow worms, since other techniques might provide shorter time to detection.

2.3.4. Comparison of techniques

2.3.4.1. Ability to generalize

From the vast array of techniques analyzed in this systematic review, it turns out that testing is commonly performed against a single dataset. These datasets belong to one of the following categories: network traffic, intrusion data, system calls, user command-line data, or audit events. While testing in any research experiment is limited by nature, the fact that most of the research work is based on a particular dataset or two datasets of the same category must be accounted for when assessing generalizability. If the results obtained with one dataset won't export to data of different nature, then the proposed implementation will be solving the problem partially, as the underlying algorithm will not be suitable for intrusion detection in general, but for intrusions given particular input information.

On the other hand, several experiments present performance results on a network protocol or service basis (Krügel, Toth, & Kirda, 2002) (Kruegel, Kirda, Mutz, Robertson, & Vigna, 2005), or are directly tested on a single service (Cho & Cha, 2004) (Kruegel & Vigna, 2003). This approach also represents a strong limitation of the proposed systems, prevent them from achieving comprehensive intrusion detection, even if the underlying algorithm is multipurpose and has been successfully applied to different use cases.

Lastly, several experiments showed acceptable results on a given attack type (Huang, Nguyen, Garofalakis, Jordan, Joseph, & Taft, 2006) (Greensmith, Twycross, & Aickelin, 2006) (Xu, Sun, & Huang, 2007) and not the others present on the datasets used for testing. This is another sign of a general lack of generalization capabilities in the current research literature.

The aforementioned lack of generalization doesn't seem to be tied, however, to the particular algorithm or modeling technique used, but to the conceptual design of the experiment. In other words, the underlying techniques in each category might be suitable for a wider range of services, protocols, datasets or attack types, but clearly that fact hasn't been the goal of any of the references reviewed. Having said that, the only promising approach (in the references included in this systematic review), from a generalization perspective, seem to be those systems based on neural networks (Saxe & Berlin, 2017) (Zanero & Savaresi, 2004), which can automatically extract features from supplied data or work with packet payload data directly. This feature effectively means that they don't rely on a particular data structure to be effective, and so they could potentially generalize well.

2.3.4.2. Detection of complex attacks

In general, an intrusion detection technique based on an unsupervised setting is expected to find attack types that can't be easily spot with more traditional approaches, like rules or signatures. In fact, the goal of anomaly detection is to identify suspicious activity by comparison with legitimate activity, not requiring any attack pattern knowledge in advance for effective detection. On the other hand, the use of general-purpose algorithms must lead to more sophisticated detections, that is, those that would go unnoticed to an experienced security analyst, or to a properly configured detection tool that relies on well-known patterns alone.

When looking at the technique categories presented in this review, two categories seem to be theoretically best positioned for this task: clustering based on neural networks (SOM and ART), and Sequence Learning.

In the first case, this is supported by the actual tests performed against datasets of very different nature, combined with their multi-layer capabilities, which allows them to produce specialized detection models across data inputs, protocols, services and attack types. This is a requirement for detection of complex attacks, whose traces can span multiple data sources.

In the case of Sequence Learning, the ability to detect complex attacks is not so much supported by the diverse or heterogeneous scenarios in which these techniques have been evaluated, but rather by the nature of those complex attacks, which can be interpreted as a sequence of malicious steps. Any technique which relies on a single data point to detect attacks will be able to detect simple attacks, whereas a technique that can correlate (and even detect in order) several circumstances associated with an intrusion will be able to spot complex, multi-stage attacks.

Regarding AIS, and particularly systems leveraging Negative Selection (NS) and Danger Theory (DT), their reliance on the non-self concept and the danger signals can make them less appropriate for complex attacks. The reason behind is that these attacks don't show a uniform pattern of activity or leave consistent traces that could be easily captured by a set of detectors, but can rather adopt many different shapes with no similarity among them, thus requiring an infeasible number of detectors to be created.

Fuzzy logic, graph, and Swarm Intelligence, on the other hand, are very general-purpose techniques. Therefore, their ability to detect complex attacks is linked to other components of the system, like data preprocessing or feature extraction. In the case of Reinforcement Learning, it seems to be a very experimental approach with very limited research work available. Lastly, statistical methods that rely on probability distribution of input data could be weak in detecting attacks which might not impact the observed probability distribution, like it's the case of many stealthy attacks that generate almost no noise compared to legitimate activity.

It's also worth noting that several experiments included in this review (Bridges & Vaughn, 2000) (Hofmeyr, 1999) (Sekar, Gupta, Frullo, Shanbhag, Tiwari, Yang, & Zhou, 2002) (Münz, Li, & Carle, 2007) can only detect attack types like DoS, probing and port scanning, which can be easily detected with static thresholds. These techniques, or the way they have been implemented, are likely far from being able to detect complex attacks that require no significant volumes of traffic or logs for success.

2.3.4.3. Scalability

Scalability refers to the ability to move from a laboratory exercise with a data sample to a real-life environment with significantly higher volumes of data.

From the techniques reviewed, Negative Selection is the one for which scalability issues have already been identified (Kim & Bentley, 2001a). However, scalability might pose a serious risk for other detection categories: statistical methods, because capturing probability distribution is an intense computational task as data volume grows; Association Rule Learning, because managing a very large ruleset can become infeasible for timely intrusion detection; graphs, because memory consumption is linked to graph size, which in turn depends on the volume of data to be represented; Markov models, because this statistic model was designed to handle a small set of states, well below the potential requirements of real intrusion data; and Reinforcement Learning, for the same reason. In the case of Reinforcement Learning, though, more recent developments using neural networks suggest that scalability might no longer be an issue, but it's yet to be demonstrated that these models can be successfully implemented for intrusion detection.

2.3.5. Discussion

Even though research which applies neural networks to intrusion detection is still under heavy development, the references included in this review suggest that techniques based on this approach are closer to offer generalizability, ability to detect complex attacks and scalability than any of the included categories. Their main disadvantage is that they require big datasets in order to produce accurate models; however, this requirement is very likely to be satisfied in real-life environments.

Fuzzy logic accounts for uncertainty, which is a desirable feature in intrusion detection. However, the developments around this tool seem to focus on producing rules to model legitimate behavior. The resulting rulesets show acceptable detection rates, but the scalability of those implementations is at least questionable. The same applies to techniques based on Association Rule Learning.

Genetic algorithms have proven to enhance another technique like clustering, and to optimize feature selection for accurate intrusion detection. However, from the references included in this review, they can't still be used as standalone methods.

Artificial Immune Systems have shown comparable performance to other methods; however, they suffer from scalability and coverage issues, and they are not suitable for detecting complex attacks, given their dependence on detectors, which are likely to represent known attack patterns.

Swarm Intelligence has had a very limited focus so far. The results obtained by available research are comparable to other methods, and researchers claim that this technique can detect unknown attacks. However, there is little support from the research community to validate those claims and confirm whether they can be repeated under different testing scenarios. The sample applies to graphs, which also suffers from potential scalability issues, as discussed before.

Sequence learning algorithms are by definition well suited to detect more complex attacks involving several steps that would take place in order for the intrusion to succeed. However, Markov models were designed to handle a small set of states, and therefore they can't scale efficiently. On the other hand, RNNs, which are sequence learning neural networks, can both generalize and scale, but there is not enough literature to strongly support these claims.

Reinforcement Learning is still a very experimental approach, and so pros and cons can be backed yet with enough experiments.

Dimensionality reduction has obtained promising results against public and commonly used datasets. It's a simple technique that combines feature extraction with detection. However, its simplicity might become an issue with real data where the underlying assumptions (for example, linearity in PCA) might not hold. Moreover, recent dimensionality reduction techniques based on deep neural networks, like autoencoders, could replace more traditional techniques in this category.

Clustering techniques not based on neural networks have also obtained promising performance and are by definition well suited to detect unknown attacks. Their effectiveness, though, heavily relies on the feature selection process, which will make anomalous clusters arise more or less clearly. The same applies to multivariate outlier detection, where the right choice of kernel function is the key to properly separate normal and anomalous instances.

Lastly, the vast array of statistical methods implemented for intrusion detection share their lack of generalization, since they have been tested against very specific datasets or attack types.

2.3.6. Conclusions

Depending on the particular intrusion detection use case, certain techniques could offer more advantages than the others. While there are techniques that are in theory better prepared to support a wide range of scenarios, like neural networks, each of the techniques included in this systematic review could successfully address a given intrusion detection need. In fact, future development paths are focusing on combining several techniques and even paradigms (supervised and unsupervised) in order to build upon the advantages of more than a single algorithm (Tsai, Hsu, Lin, & Lin, 2009). Another approach would be to reengineer methods that have been successfully applied in a supervised setting, so that they can also fit into unsupervised approaches. In fact, CNN has only been recently applied to building an IDS in an unsupervised setting. It has been though leveraged in supervised approaches (Upadhyay & Pantiukhin, 2017), which built a CNN from the DeepLearn ToolBox by Mathworks Matlab and obtained a classification error rate below 2% for the KDD99 dataset.

On the other hand, complex attacks are likely to span multiple datasets, which suggests that data fusion could be another development path. This engineering discipline makes inferences from activities, events and situations whose data is gathered from multiple sources.

This approach would allow to achieve a higher degree of inference, including identity of intruder, behavior of intruder, situation assessment and threat analysis. Previous research (Bass, 2000) has already proposed to apply multisensor data fusion to intrusion detection systems.

Lastly, all techniques presented in this review rely on trusted training data to successfully detect intrusions. However, an attacker could inject noise into the data used for training and trick the anomaly detector to model an attack as normal behavior. This adversarial noise issue has been analyzed by previous research (Kloft & Laskov, 2010). When using a simple learning algorithm like online centroid anomaly detection, using HTTP traffic and a set of well-known web application exploits, an attacker would need to control over 5-15% of the traffic used for training for a poisoning attack to succeed. In a similar setting, a *poisoning attack* led to same conclusions (Kloft & Laskov, 2012). Both experiments suggest that the resilience of an algorithm to these attacks is a relevant feature for which extensive research is needed.

2.3.7. Limitations of this review

This systematic review follows a procedure commonly used in the research community that aims at providing extensive coverage of the topic under discussion and answer key questions related to the motivation and objectives of this research.

However, this review has a number of inherent limitations. First, the extensive literature on the topic makes it hard to deliver full coverage and include every single research work that might be relevant to the discussion. Therefore, some references that could have also been included in this review might be missing. For the same reason, this review has focused on popular techniques with several research experiments associated with them. Niche methods or very custom approaches that are not supported by different researchers have been discarded in the process.

Lastly, obtained conclusions are based on relevant findings of previous research and how they have been interpreted in the context of this research work's motivation and objectives. Other researchers describing those findings from another perspective or evaluate them under a different context might obtain different conclusions.

In any case, to the best of my knowledge, this systematic review constitutes an in-depth assessment and provides a solid foundation to the rest of this research work.

2.3.8. Supporting background research

This section includes research literature that doesn't fit into the scope of the systematic review but had to be reviewed prior to starting the developments associated with this research work.

2.3.8.1. Intrusion detection

In this section, relevant research literature in which supervised learning has been applied to intrusion detection is presented and discussed. These references were included for comparison purposes: they allow to measure the relative degree of success obtained by this research work's developments, that is, to what extent these developments outperform any of the performance results obtained with techniques that have prior knowledge of attack patterns.

Most common classifiers (Ahmed, Mahmood, & Hu, 2016) used to build an IDS are based on SVM, Bayesian networks, neural networks, and rule mining methods.

SVM, used in combination with Deep Belief Networks for feature extraction, achieved (Marir, Wang, Feng, Li, & Jia, 2018) 90.47% precision, 97.21% recall and 93.72% F-measure when tested against UNSW-NB15. An IDS based on Bayesian networks (Kruegel, Mutz, Robertson, & Valeur, 2003) leveraging OS calls triggered by the DARPA 1999 dataset (Mahoney & Chan, 2003b) achieved accuracies ranging from 75% to 100%. Deep learning has reported enhanced performance (Kwon, Kim, Kim, Suh, Kim, & Kim, 2019). These include (Zhao, Yan, Chen, Mao, Wang, & Gao, 2019): autoencoders, Deep Belief Networks, Deep Boltzmann Machines, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN).

Deep Belief Networks, used in combination with Modified Density Peak Clustering Algorithm (MDPCA-DBN), obtained 90.21% accuracy against UNSW-NB15 dataset. A variant of Boltzmann Machines (Aldwairi, Perera, & Novotny, 2018), Restricted Boltzmann Machines, was trained with the ISCX dataset (Shiravi, Shiravi, Tavallaee, & Ghorbani, 2012) using Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD) algorithms. It achieved 88.6% (CD) and 89.0% (PCD) accuracies, 88.4% (CD) and 84.2% (PCD) true positive rates, and 88.8% (CD) and 93.8% (PCD) true negative rates.

RNNs have been leveraged to model sequences of basic network parameters and predict legitimate future behavior (Radford, Apolonio, Trias, & Simpson, 2018). When focusing on protocol-bytes sequences, Area Under the Curve (AUC) ranged from 0.78 to 0.84. On the other hand, a CNN (Saxe & Berlin, 2017) was able to detect attacks using OS and application-level resources as input, including URLs, files and registry keys. The system achieved a detection rate ranging from 97.8% to 99.3%.

Natural Language Processing (NLP) implemented through an artificial neural network has been used to build an IDS (Mimura & Tanaka, 2017). Particularly, Paragraph Vector algorithm was used, which creates embeddings for documents. An SVM classifier was also leveraged to classify legitimate and malicious traffic. The aim of this research was to detect command-and-control activity generated by malware. Precision ranged from 77% and 99%, and recalled ranged from 69% and 100%. With another NLP technique, based on skip-gram models (Carrasco & Sicilia, 2018), 99.2% precision and 82.07% recall were reported.

CNNs are useful to detect malicious URLs, file paths and registry keys. The obtained detection rate ranged from 97.8% to 99.3% (Saxe & Berlin, 2017). Another method based on Cerebellar Model Articulation Controller and RL was able to successfully detect DoS attacks (Cannady, 2000). The obtained error was 3.28E-05%.

Previous research (Kumar, Glisson, & Cho, 2020) compared the performance of MeanShift, K-means, and Fuzzy C-Means. Detection rate was 79.1%, 82.3%, and 84.6%, respectively, and accuracy was 81.2%, 77.2%, and 82.1%. The dataset used was KDD 99. Another research (Nawir, Amir, Yaakob, & Lynn, 2019) tested five classification algorithms against the UNSW-NB15 dataset: Naïve Bayes, Averaged One Dependence Estimator (AODE), RBFN, MLP, and J48 trees. Accuracies ranged from 76.12% to 98.71% (J48 trees).

Deep learning was tested (Fernández & Xu, 2019) in both supervised and unsupervised settings to detect intrusions in the CIC IDS 2017 dataset. 96.77% TPR and 0.52% FPR were obtained with the supervised approach, and 0.013% FPR and 76.70% FNR with the unsupervised (autoencoder) setting.

Another research (Meftah, Rachidi, & Assem, 2019) performed feature selection, using Recursive Feature Elimination and Random Forest, to find top performing features and input them to several binary classifiers. Best results were obtained by Support Vector Machines (SVM), with an accuracy of 82.11% when tested against the UNSW-NB15 dataset. Also, a variation of SVM with a new scaling method obtained an accuracy of 85.99% against UNSW-NB15 (Jing & Chen, 2019).

Previous research discovered that feature selection optimization (Parker, Yoo, Asyhari, Chermak, Jhi, & Taha, 2019) led to an accuracy of 98.04% in the Aegean Wi-Fi Impersonation Attack Detection dataset. Another research (Blaise, Bouet, Conan, & Secci, 2020) developed a technique that detects bots (malware) through clustering, by classifying new hosts either as bots or benign ones based on distances to labeled clusters. Tested against CTU-13 dataset, the obtained F1-Score ranged from 80% and 93%.

Lastly, Convolutional Neural Networks were used by another research (Van Huong & Hung, 2019) to detect intrusions in IoT systems. The obtained average accuracy was 98.9%.

2.3.8.2. Fraud detection

In this section, research literature focused on increasing fraud detection rate are presented and discussed. Their obtained results are useful to determine to what extent this research work's development provide comparable performance or even outperform existing approaches. Supervised and unsupervised learning techniques are included.

Recent research states that credit card fraud detection can be best implemented by using artificial neural networks (Rushin, Stancil, Sun, Adams, & Beling, 2017), after evaluating the performance obtained by both neural networks and other techniques like SVM, AIS, KNN, Dempster Shafer Theory, decision trees, logistic regression, association rule learning, active learning, Cardwatch (Aleskerov, Freisleben, & Rao, 1997) and several others.

From a set of classifiers that included random forest, KNN, decision trees, and logistic regression, the latter achieved best accuracy (Soh & Yusuf, 2019).

In terms of skewness, techniques like C5.0 classifier, SVM, neural networks and logistic regression obtained better results than KNN, AIS, Naïve Bayes or Bayesian Belief Network (Makki, Assaghir, Taher, Haque, Hacid, & Zeineddine, 2019). On the other hand, imbalanced data is better managed by

neural networks, compared with SVM, decision trees and random forest (Parthasarathy, Ramanathan, JustinDhas, Saravanakumar, & Darwin, 2019).

Extreme Gradient Boosting (GB) and Random Forest (RF), both supervised methods, obtained better results (in terms of AUROC metric) than Restricted Boltzmann Machine (RBM) and Generative Adversarial Networks, which had obtained top performance among unsupervised methods. The vales ranged from 98.8% to 98.9%. The chosen feature selection method is also a driver performance. In particular, filter and wrapper methods obtained better accuracy for PART and J48 decision trees (Singh & Jain, 2019). It also obtained better sensitivity and precision levels for random forest, J48 decision tree and AdaBoost.

A novel method leveraging a non-parametric approach exploited data reduction to extract relevant transactions (Vardhani, Priyadarshini, & Narasimhulu, 2019), while another leveraging DeepWalk and decision trees with Gradient Boosting obtained an F1 score which ranged from 61.43% to 71.84%.

The hypersphere model (Chen, Zhang, Liu, Yang, Meng, & Wang, 2019) has been used in an unsupervised setting to model normal user activity, obtaining better performance than other methods. On the other hand, Local Outlier Factor (LoF) and Isolation Forest have outperformed other artificial intelligence techniques (Sharmila, Kumar, Sundaram, Samyuktha, & Harish, 2019).

A neural network trained to recognize sequences has proven to achieve notable performance (Yang & Xu, 2019). This method also makes it easier to interpret model results, and is better prepared to tackle concept drift. In fact, higher performance is observed in malicious credit card transactions when concept drift is accounted for (Lucas, Portier, Laporte, Calabretto, He-Guelton, Oblé, & Granitzer, 2019). Hidden Markov Models have delivered better precision and recall compared to other fraud detection systems (Lucas, Portier, Laporte, Calabretto, Caelen, He-Guelton, & Granitzer, 2019).

When skewness is accounted for, unsupervised methods show higher performance than supervised techniques (Mittal & Tyagi, 2019). On the other hand, when data is oversampled (using a method like SMOTE), accuracy increases in supervised techniques (Varmedja, Karanovic, Sladojevic, Arsenovic, & Anderla, 2019). Lastly, accuracy might decrease if several scores are combined to come up with a final score (Carcillo, Le Borgne, Caelen, Kessaci, Oblé, & Bontempi, 2019).

Better accuracy is obtained when SVM, KNN and MLP classifiers are implemented together. The comparison was done against standalone classifiers, including these three and several others (Naïve Bayes, Extreme Learning Machine) (Prusti & Rath, 2019a) (Prusti, Padmanabhuni, & Rath, 2019). A variant of AdaBoost and stacking ensemble technique obtained better accuracy (94.5%) than other methods as stacking, AdaBoost, RF, logistic regression, and DT (Prabhakara, Kumarb, Ponnarb, Sureshb, & Jayandhiranb, 2019).

Semantic fusion (Darwish, 2020), with artificial bee colony algorithm (ABC) and k-means clustering obtained a high accuracy. Indeed, higher accuracy is obtained if customers are classified into groups before transactions are processed (Kasa, Dahbura, Ravoori, & Adams, 2019).

Previous fraud detection research (Dornadula & Geetha, 2019) compared the accuracy of five different algorithms with a novel technique implementing oversampling, behavioral pattern extraction, and feedback management. Authors obtained the following results: Local Outlier Factor (45.82%), Isolation Forest (58.83%), Logistic Regression (97,18%), Decision Trees (97.08%), and

Random Forest (99.98%). A similar research (Varmedja, Karanovic, Sladojevic, Arsenovic, & Anderla, 2019) concluded that Random Forest leads to best results, with 99.96% accuracy. Yet another study (Kang, 2019) that compared Random Forest and Boosted Trees led to accuracies of 98.98% and 99.24%, respectively. Lastly, another research (Kumar, Soundarya, Kavitha, Keerthika, & Aswini, 2019) leveraging Random Forest obtained 90% accuracy.

Another research (Misra, Thakur, Ghosh, & Saha, 2020) proposed a two-stage model in which an autoencoder was used for feature engineering and several algorithms (Multi Layered Perceptron, K-nearest Neighbor, logistic regression) for classification. The resulting F-score ranged from 77.15% to 82.65%. On the other hand, SVM was the algorithm of choice of another recent research (Gupta, 2019), that used a dataset with ten years of credit card transactions. It achieved an accuracy on 90.00% in the classification task. Similarly, another research (Latif, Kulkarni, Molkire, & Nangare, 2020) leveraging Naïve Bayes theorem achieved 94.57% accuracy.

Paysim1 dataset was leveraged to for performance comparison of DT, neural networks, and SVM(Shahed, Ibrahim, & Akter, 2019). The accuracies obtained were 94.04%, 96.14%, and 97.83%, respectively. Undercomplete autoencoders (Misra, Thakur, Ghosh, & Saha, 2020) have also been tested against the Paysim1 dataset, achieving 85.34% precision and 80,15% recall.

Yet another research (Shukur & Kurnaz, 2019) geared towards algorithm comparison concluded that Isolation Forest was the best-performing technique, with 99.7% accuracy. Another study (Safa & Ganga, 2019) that compared Naïve Bayes, logistic regression and K-Nearest Neighbors obtained accuracies of 83.00%, 97.69% and 54.86% respectively. Lastly, other approaches which have proven their effectiveness in improving classification results for fraud detection include Generative Adversarial Networks (GAN) (Fiore, De Santis, Perla, Zanetti, & Palmieri, 2019), deep learning and hybrid ensemble (Kim, Lee, Shin, Yang, Cho, Nam, 2019), a hybrid model (Prusti & Rath, 2019b) combining decision tress, SVM, and KNN models, and an optimized SVM model combined with the Random Under Sampling (RUS) technique (Pambudi, Hidayah, & Fauziati, 2019).

Lastly, the combination (Eshghi & Kargari, 2019) of decision trees, transaction sequences and user activity (semisupervised setting) improved detection rate by 7%.

2.3.8.3. Alert reduction

In this section, research literature related to this research work's development around fraud detection optimization is presented and discussed.

2.3.8.3.1. Intrusion detection

ALAC (Adaptive Learner for Alert Classification) prototype leveraged adaptive learning for classification of alerts in real time. The system (Pietraszek, 2004) learnt decisions patterns from human expertise, and leveraged a RIPPER classifier (Cohen, 1995). RIPPER (Lee, 1999) delivers generalization accuracy and concise conditions. False alarm rate dropped by 30% (Lippmann, Haines, Fried, Korba, & Das, 2000) when tested with DARPA 1999 dataset.

Clustering also allows to reduce alert volume (Njogu & Jiawei, 2010). Data was enriched with evidence and vulnerability assessment data. The reduction rate was 78%, using the DARPA 1999 data for testing.

Greedy aggregation, also a clustering algorithm, obtained an 83.2% reduction rate (Harang & Guarino, 2012) for Snort (open-source IDS) alerts. Alerts were classified into groups of meta-alerts, with common details.

Snort IDS alerts included in DARPA 1999 data were used to test a method (Nauck, Nauck, & Kruse, 1999) using Jrip and NEFCLASS. Jrip is a RIPPER implementation in Weka. Detection rate was 88% with the first (Jrip), and 84.63% with the latter (NEFCLASS).

A reduction rate of 35.04% was obtained (Bakar, Belaton, & Samsudin, 2005) with data enrichment through an alert quality framework. The quality level was calculated based on criteria like reliability, sensitivity, accuracy and correctness. Additional details like vulnerability details were included. The vulnerability information applied to OS, applications, and network services.

Lastly, frequent pattern-based outlier detection (He, Xu, Huang, & Deng, 2005) has been applied to alert reduction, with rates ranging from 86% to 92% (Xiao, Jin, & Li, 2010). The dataset used was extracted from traffic in a network with 10 systems.

2.3.8.3.2. Deep learning for decision-making automation

Deep Learning (DL) can automate decision-making activities for intrusion detection (McElwee, Heaton, Fraley, & Cannady, 2017), loan application processing (Srivastava, 1992), managerial decision making (Hill & Remus, 1994), medical diagnosis and treatment prescription (Liang, Zhang, Huang, & Hu, 2014), and clinical imaging classification (Ciritsis, Rossi, Eberhard, Marcon, Becker, & Boss, 2019). The combination of rules and artificial neural networks (Tan, Quah, & Teh, 1996) can also support the automation of these processes.

A classifier based on a deep neural network (McElwee, Heaton, Fraley, & Cannady, 2017) was able to triage IDS alerts. The groups into which alerts were classified had meaning for human experts, which confirmed its effectiveness.

Business judgement can also be automated with artificial neural networks. For example, they were applied (Srivastava, 1992) to loan applications, using data about companies with a label indicating whether they defaulted or not. Decisions were taken using the same criteria as humans.

On the other hand, Deep Learning (DL) outperformed rules and shallow neural networks for medical diagnosis (Liang, Zhang, Huang, & Hu, 2014). It also resembled criteria used by human experts in this field better than other methods.

Lastly, a Deep CNN outperformed humans in tasks like classification of clinical images (Ciritsis, Rossi, Eberhard, Marcon, Becker, & Boss, 2019). Data used for testing included 101 images from an internal dataset (accuracy from 87.1% to 93.1% for dCNN and from 79.2% to 91.6% for humans) and 43 from an external dataset (AUC 96.7% for dCNN and 90.9% for humans). Both were extracted from the Breast Imaging Reporting and Data System (BI-RADS).

3. Materials and methods

3.1. Research overview

In this section, an overview of the research work done is presented. The work is split into four different developments, each of them addressing one or more of the objectives shown in the introductory section of this document.

3.1.1. Unsupervised intrusion detection

In order to reduce the dependence on well-known attack patterns for effective intrusion detection in the cybersecurity field, an unsupervised technique was developed that can accurately distinguish malicious network activity by comparing it with legitimate behavior (Carrasco & Sicilia, 2018).

A variant of word2vec (skip-gram) was applied. The technique captures the relationship among nodes (entities) in a network, as well as its legitimate activity. In the detection stage, attacks are detected by measuring how far is observed activity from expected activity for the entities (Carrasco & Sicilia, 2018).

The technique aims to increase artificial intelligence's contribution to intrusion detection, by improving in one or more aspects the existing methods. It leverages UNSW-NB15 (Moustafa & Slay, 2015) public intrusion dataset, which allows proposed approach to be validated, replicated and improved in the future by the research community.

3.1.2. Supervised fraud detection optimization

Multiple settings leveraging deep neural networks were tested to assess whether they could capture the criteria used by humans to discard false positives in anomalous credit card transactions (Carrasco & Sicilia, 2020). The underlying neural networks had already demonstrated to deliver notable performance in other fields (Liu, Wang, Liu, Zeng, Liu, & Alsaadi, 2017).

This development also aims to increase artificial intelligence's contribution to fraud detection, by improving in one or more aspects the state-of-the-art methods found in research literature.

3.1.3. Semisupervised intrusion detection

In order to demonstrate that unsupervised intrusion detection can be augmented by supervised learning, a deep autoencoder leveraging xgboost algorithm for feature selection was designed and tested. In fact, three approaches were implemented and compared: excluding source and target IP addresses from the feature set, including all available features, and selectively including features, based on the feature importance metric supplied by xgboost algorithm. In addition to this, One-Hot Encoding and logarithmic scale discretization were used for feature engineering purposes.

This development leverages UNSW-NB15 (Moustafa & Slay, 2015) public intrusion dataset, which allows proposed approach to be validated, replicated and improved in the future by the research community.

3.1.4. Unsupervised cross-domain malicious behavior detection

In order to prove that artificial intelligence can evolve towards cross-domain applications, an unsupervised technique leveraging topic modeling was designed, implemented and tested against three datasets of network traffic, Internet of Things (IoT) malware traffic, and payment transactions.

In an attempt to build upon this previous research on NLP and topic modeling, LDA was selected as the baseline algorithm of proposed technique. On top of the algorithm, it was developed: a feature engineering technique that converts both categorical and numerical feature values into words, and a tailored scoring mechanism for anomaly detection. Also, the concept of entity was shifted from the subject performing the action to the action being performed.

Publicly available datasets in each field were used. UNSW-NB15 (Moustafa & Slay, 2015) was selected for intrusion detection, Paysim1 ("Synthetic Financial Datasets For Fraud Detection", 2017) for payments fraud detection, and IoT-23 ("IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic", n.d.) for malware detection in IoT devices. The datasets all share the trait of capturing forms of malicious behavior, while being diverse in the environments from which data is extracted. They all have been used in previous research, hence allowing to compare results with those obtained by other authors.

3.2. Datasets

3.2.1. UNSW-NB15

Synthetic datasets support most intrusion detection research (Gogoi, Bhuyan, Bhattacharyya, & Kalita, 2012). The reason behind this is that real traffic data have only become available very recently. In this research, UNSW-NB15 (Moustafa & Slay, 2015) was used. It contains realistic and recent legitimate activity, and recent attack types belonging to 9 categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

UNSW-NB15 was made available by the Australian Centre for Cyber Security (ACCS), and created with PerfectStorm ("PerfectStorm", n.d.). With *tcpdump*, 100 GB of traffic was sniffed. 49 features were produced with Argus and Bro-IDS. Label classified observations into normal or attack. Data is delivered as 4 CSV files. For this research, they have been combined into a single file with 2,218,763 observations of normal activity and 321,283 attacks.

The file was used for training and testing in the following developments: unsupervised intrusion detection, semisupervised intrusion detection, and unsupervised cross-domain malicious behavior detection.

Other common datasets used in the IDS field include KDD Cup 99 and NSL KDD, an enhanced version of KDD Cup 99. KDD Cup 99 suffers from two statistical issues (Tavallaee, Bagheri, Lu, & Ghorbani, 2009): a significant number of redundant records, which leads to a bias towards more frequent observations, and artificially high accuracy measures, which prevents from proper performance comparison among algorithms. While NSL KDD partially solves these issues, it doesn't represent modern attack environments, as it is still based on the same malicious traffic of KDD Cup 99. IDEVAL, also a very popular dataset for intrusion detection, also suffers from several issues (Mahoney & Chan, 2003b).

UNSW-NB15 solves some of the key issues with datasets as KDD Cup 99 and NSL-KDD (Moustafa & Slay, 2016), including: lack of recent attack types, lack of recent normal activity, and different distribution of training and testing sets.

3.2.2. Paysim1

Paysim1 ("Synthetic Financial Datasets For Fraud Detection", 2017) is a synthetic dataset of payment transactions. It was produced from 1 month of real transactions recorded by a mobile money service in Africa. It was produced using Paysim (Lopez-Rojas, Elmir, & Axelsson, 2016), a financial mobile money simulator designed to support fraud detection research. The dataset contains 6,354,407 observations of legitimate transactions, and 8,213 of fraudulent transactions.

Dataset features are enumerated in Appendix 8.2.

This dataset was used for the unsupervised cross-domain malicious behavior detection development.

3.2.3. IoT-23

IoT-23 ("IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic", n.d.) is a dataset of malware traffic from IoT devices. It contains 20 malware traces executed in IoT devices, and three samples of legitimate traffic. Traffic was recorded in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. It was published in January 2020. Its purpose is to allow researchers to design algorithms suitable for IoT-related malware detection. From all the available captures, CTU-IoT-Malware-Capture-1-1 was selected for this research. It contains 469,275 observations of legitimate traffic, 539,465 observations of port-scanning activity, and 8 observations of C&C (Command & Control) malicious traffic.

Dataset features are enumerated in Appendix 8.3.

This dataset was used for the unsupervised cross-domain malicious behavior detection development.

3.2.4. Fraud dataset

446,076 real alerts issued during 6 months of anomalous credit card transactions from a payment processing organization in Spain were also used in this research (Carrasco & Sicilia, 2020).

The percentage of confirmed and discarded alerts is included in Table 5.

Label	# Records	Percentage (%)
Confirmed	195,265	43.77%
Discarded	250,811	56.22%

Table 5. Dataset label distribution.

The feature set is described in Table 6 (Carrasco & Sicilia, 2020).

Feature	Description	Type	Cardinality	Range
Amount	Amount in local currency.	Numerical	N/A	[0, 231.014,48]
Day of month	Day of month.	Numerical	N/A	[1, 31]
Hour	Hour.	Numerical	N/A	[0, 23]

Data Input	PoS identification method for credit card.	Categorical	21	N/A
Authentication method	PoS customer authentication method.	Categorical	12	N/A
Response code	Code returned by the payment processing platform after processing.	Categorical	59	N/A
Merchant type (international)	Type of business, based on international codes.	Categorical	507	N/A
Merchant type (domestic)	Type of business, based on Spanish codes.	Categorical	325	N/A
City	-	Categorical	54	N/A
Country	-	Categorical	187	N/A
Issuing bank	Bank that issued the credit card.	Categorical	107	N/A
Score	Risk level assigned by the fraud detection system.	Numerical	100	N/A
Label	Human expert decision: confirm or discard.	Binary	2	N/A

Table 6. Input columns.

This dataset was used for the supervised fraud detection optimization development.

3.3. Algorithms

3.3.1. Skip-gram modeling

The word2vec algorithm transforms words into embeddings (vectors). The CBOW variant predicts a word from context, while the skip-gram variant predicts context from a given target word (Mikolov, Chen, Corrado, & Dean, 2013).

In skip-gram, the goal is to obtain high probability (cosine similarity) for words that belong to the context of the target word. This variant is more suitable for data with many records, like it is the case for intrusion detection. The reason for this is that it manages context-word pair as a single observation. Training speed and quality of embeddings was improved by algorithm extensions (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

The skip-gram algorithm was modified in this research (Carrasco & Sicilia, 2018), so that it can accurately detect intrusions. The underlying assumption was that both nodes in a network and the connections they initiate (behavior) can be represented as words.

word2vec and skip-gram have been leveraged in multiple domains. The skip-gram variant can uncover relationships among cancer patents (Whitehead & Johnson, 2017). It can also uncover semantic information in medical words (Zhou, Fu, Qiu, Zhang, & Liu, 2017), perform protein classification (Islam, Heil, Kearney, & Baker, 2017), predict consumer acceptance (Kim, Ha, Choi, & Moon, 2018), extract sentiment information against movie reviews (Chakraborty, Bhattacharyya, Bag, & Hassanien, 2018), model harmony (Sears, Arzt, Frostel, Sonnleitner, & Widmer, 2017), evaluate human judgment (Hollis, Westbury, & Lefsrud, 2017), and infer psycholinguistic properties (dos Santos, Duran, Hartmann, Candido, Paetzold, & Aluisio, 2017).

In anomaly detection, skip-gram can detect deviations in log files (Bertero, Roy, Sauvanaud, & Trédan, 2017). In cybersecurity, word2vec can be applied to feature engineering (Zhuo, Zhang, & Son, 2017). It can also find learn semantics of attack types, to be used as input to a CNN classifier (Barot, Zhang, & Son, 2016).

3.3.2. Topic modeling

Topic modeling is a NLP method used for discovering latent topics in a corpus of documents. Originally proposed in 1998 (Papadimitriou, Raghavan, Tamaki, & Vempala, 2000), it was further developed from a probabilistic standpoint (Hofmann, 2013). It has been applied to multiple fields: themes and trends discovery in transportation research (Sun & Yin, 2017), anomaly detection in video data (Girdhar, Cho, Campbell, Pineda, Clarke, & Singh, 2016), anomalous event detection in text documents (Shin, Choi, Choi, Langevin, Bethune, Horne, 2017), spatio-temporal event analytics on social media (Choi, Shin, Choi, Langevin, Bethune, Horne, 2018), insider threat detection (Kim, Park, Kim, Cho, & Kang, 2019), anomaly detection in multi-view data (Zhang, Iwata, & Kashima, 2017), and supply-chain analysis (Bansal, Gualandris, & Kim, 2020). Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) is a topic modeling variant developed in 2003 that builds upon the original development (Papadimitriou, Raghavan, Tamaki, & Vempala, 2000), using a hierarchical Bayesian model.

LDA can describe network behaviors, and is a promising approach for detecting zero-day attacks and other network threats (Cramer & Carin, 2011). It has also been proposed to detect unauthorized access in SSH logs (Aswani, Cronin, Liu, & Zhao, 2015) and malicious user behavior (Huang, Kalbarczyk, & Nicol, 2014), by modeling topics of legitimate and attack-related activities.

3.3.3. Neural networks

3.3.3.1. Multi-Layer Perceptron

MLP is a feedforward neural network with at least 3 layers (input, output, and hidden). Neurons are fully connected and implement a non-linear activation function. Backpropagation is leveraged to minimize the loss function.

3.3.3.2. Convolutional Neural Network

CNN architecture is also based on multiple layers (input, output, and hidden). It's implemented as sequences of convolution (sliding dot product of input and a rectifier linear unit) and max pooling (non-linear down-sampling) operations with a fully connected layer.

3.3.3.3. Deep autoencoders

DAE is a neural network used to reduce dimensionality. It's leveraged in unsupervised settings. It compresses input and decompresses it back (output). Outliers are easily uncovered because they obtain a high reconstruction error.

An autoencoder is a class of neural network for unsupervised learning that produces efficient encoded representations of input data (Liou, Cheng, Liou, & Liou, 2014). It's commonly used for dimensionality reduction purposes. When its architecture includes several hidden layers, it's considered a deep autoencoder. A standard deep autoencoder architecture is visually depicted in Figure 3.

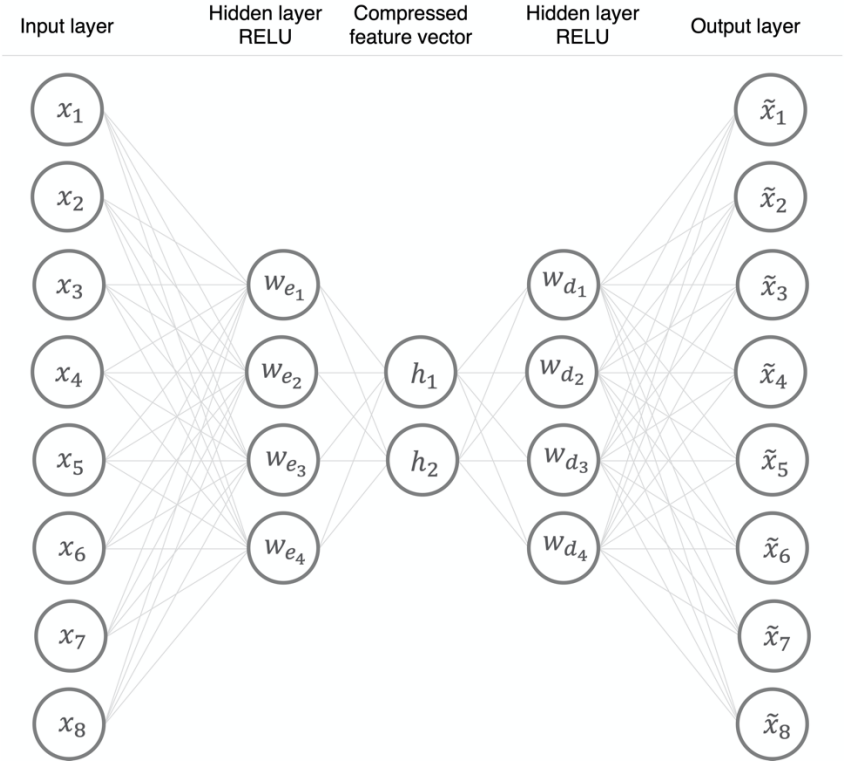

Figure 3. Deep autoencoder neural network structure.

The autoencoder learns compressed representations of training data, from which it generates output that resembles the original input. This reconstruction task is shown in Figure 4 for both handwritten digit recognition and (simplified) network traffic events. To achieve their goal, autoencoders try to minimize a loss function which express reconstruction error of frequently observed inputs. The reconstruction error is defined as follows:

$$reconstruction\ error\ = \frac{1}{N} \times \sum_{i=0}^{N} (x_i - \tilde{x}_i)^2$$

where x_i represents each component of the input vector, $N$is the input length, and $\tilde{x}_i$ represents each component of the reconstructed input.

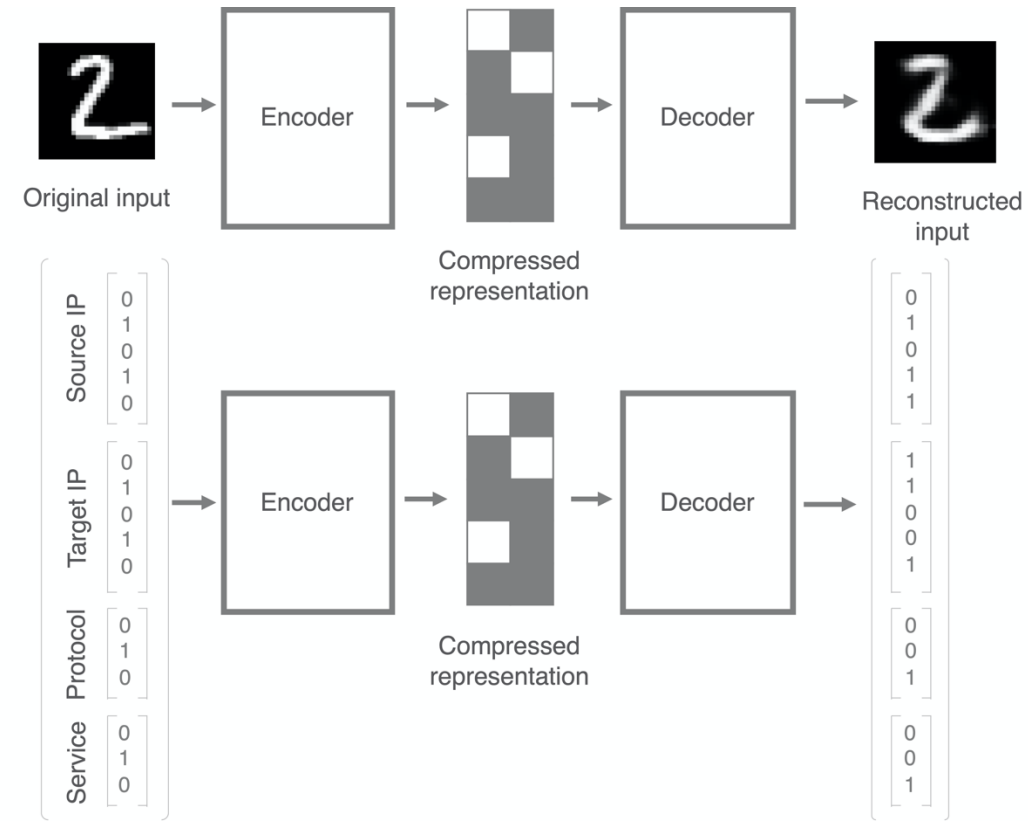

Figure 4. Deep autoencoder behavior in handwritten digit recognition and in intrusion detection.

3.4. Experimental setting

3.4.1. Unsupervised intrusion detection

3.4.1.1. Dataset

UNSW-NB15 (Moustafa & Slay, 2015) was used. Its features are enumerated in Appendix 8.1.

3.4.1.2. Feature engineering

The hypothesis for feature engineering is that a system owned by an attacker will connect to network nodes and ports that were not observed in the past, before it was compromised (Carrasco & Sicilia, 2018). In this context, only four features related to network connections were selected: source IP, destination IP, destination port and protocol.

The other features were not used, for the reasons described next:

- Aggregated features. Producing them requires a lot of computation power in a real network, which would impact real time performance.
- Protocol-related. These features are not generally available, and so including them would restrict proposed approach to very specific environments.

- Application-based. If included, they would restrict detection capabilities to those applications.
- Time-based. They can be easily manipulated by an attacker.

The overall objective was to design an approach that could be applied to real datasets.

Compared to other methods that use the 49 features (Bamakan, Wang, & Shi, 2017), proposed approach has low storage requirements (from 586.4 MB to 101.2 MB, an 82.7% reduction rate). The issue with having to process big data has been pointed out by other researchers (Garvey & Lunt, 1991) and is minimized with proposed approach (Carrasco & Sicilia, 2018).

The *dport* feature was transformed: only the most frequent values were kept, while the others were replaced by a default value. This transformation helped to increase performance because it allows to discriminate connections to well-known and unusual services (Carrasco & Sicilia, 2018).

3.4.1.3. Neural network design

TensorFlow (Abadi et al, 2016) is an open-source machine learning framework that has been applied in fields like robotics, computer vision, natural language processing and speech recognition. It was created by Google.

TensorFlow was used in this research as the underlying framework to implement skip-gram. The resulting neural network architecture had the following layers: input, embeddings, and a softmax classifier (output).

Figure 5 displays the neural network architecture used for skip-gram:

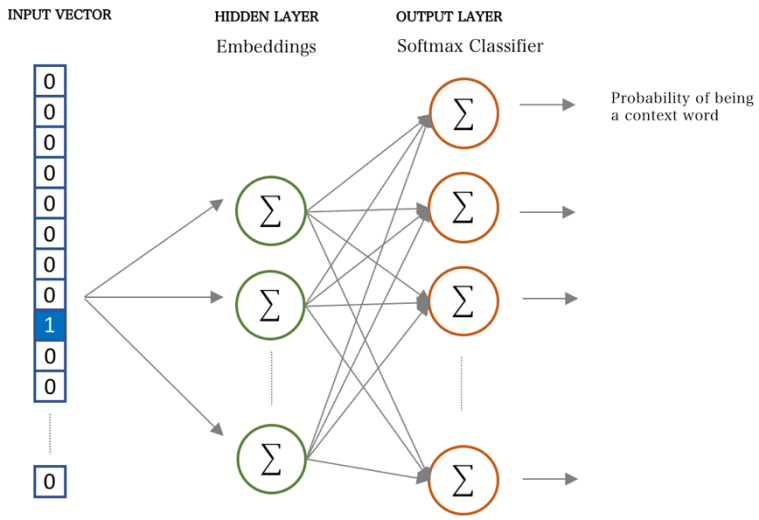

Figure 5. Skip-gram neural network architecture.

Embeddings are initialized with random values ([-1, 1]). These values represent a sample from a uniform distribution. Weighs are populated from a normal distribution and updated through SGD (Stochastic Gradient Descent) with Adagrad optimizer (which adjusts learning rate based on

parameter frequency), while all bias values are set to zero (0). Adagrad is the most suitable optimizer because it properly handles sparse data (as One-Hot encoding inputs used in our approach).

The training process creates a set of normalized embeddings that can be used to calculate cosine similarity between entities (network nodes and network connections) (Carrasco & Sicilia, 2018).

Other parameters in the design were adjusted through grid search, as shown later. On the other hand, the micro batch function was customized to properly encode inputs.

3.4.1.4. Algorithm reengineering

word2vec handle words in a document. Microbatches contains pairs of target word and context word. These data points are used to refine embeddings in each iteration. This approach allows to predict whether a word is in the context of a target word or not (negative sampling).

This approach is visually depicted in Figure 6.

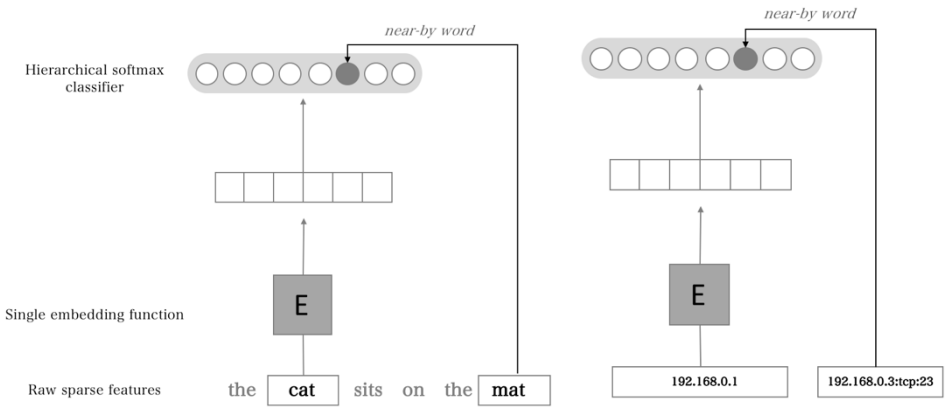

Figure 6. Skip-gram modeling for text classification and as IDS.

In proposed approach (Carrasco & Sicilia, 2018), target words represent network nodes, or *systems*, that connect to other nodes. Context words, in turn, represent connection details, or *connection types*. Connections are defined by a target IP, a protocol and a destination service, represented by a port number.

Therefore, context around a target word represent most probable connections of a given network node, and obtained embeddings (of network nodes and network connections) meet the following criteria: nodes whose embeddings are similar (close) connect to similar nodes and services; if the embedding of a node is similar (close) to the embedding of a connection, the node frequently creates that type of connection to other nodes; if the embeddings of two network connections are similar (close), then they are initiated by the same nodes (Carrasco & Sicilia, 2018). As a result, embeddings that are far from each other represent unusual connections for a network node, or nodes whose connection profile is very different.

During training, embeddings are obtained. After that, for each recorded event, a network node and its connection details are extracted, and their embeddings' similarity is obtained. Low similarity means infrequent activity. Multiple events associated with a given node represent infrequent activity are a signal of anomalous activity taking place.

This approach is visually depicted in Figure 7:

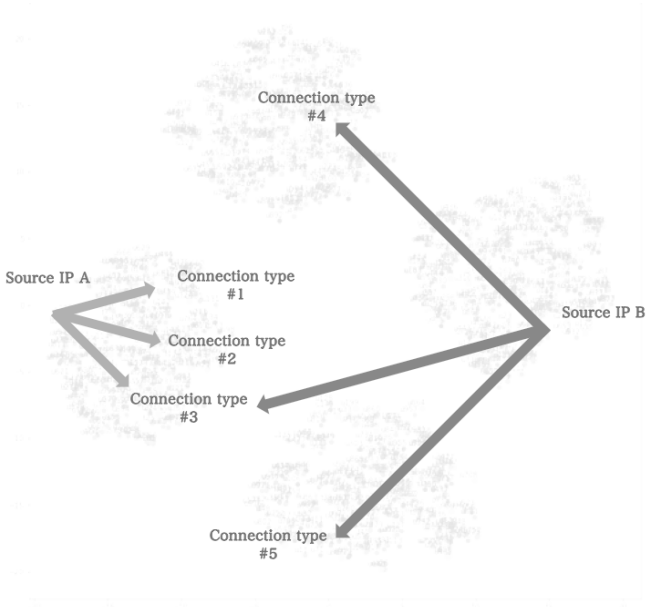

Figure 7. Examples of normal and anomalous interactions.

Leveraging a set of embeddings helps to reduce the FPR, because the model doesn't only spot infrequent activity of a given node by comparing current activity with historical records of that node, but also compares that activity with the one of similar nodes (which obtain similar embeddings, as explained before). This means that inference goes well beyond what it is known about a given node, and expands to what it is collectively known about nodes exhibiting similar behavior in the past. If a given normal behavior for a given node is, for some reason, not observed in the training phase, that lack of information won't generate false positives if that behavior is observed for other similar nodes (Carrasco & Sicilia, 2018).

3.4.1.5. Microbatch building function

In order to apply skip-gram to intrusion detection, the microbatch function originally included in TensorFlow implementation was redesigned. After the redesign, each record supplied to the neural network for processing is a list of items (the equivalent to a document in the original form of the algorithm), in which the first item (target word) is a network node ($p = 0.8$) or connection details ($p = 0.2$), and the remaining items are the connections initiated by that node (context). If a node initiates a connection multiple times (in the training data), the corresponding item is found in the list multiple times as well. Therefore, each record contains all connection types associated with a node (Carrasco & Sicilia, 2018).

The microbatch function takes consecutive samples (with no overlap) of the dataset. This sampling is controlled by the *batch_size* parameter. If the first item is a node, the algorithm learns the relationship between that node and its connections. If the first item is a connection, the algorithm learns the relationship between a connection of a node and the remaining connections of that node. This stochastic approach was included in order to learn accurate embeddings (Carrasco & Sicilia, 2018).

The set of connections included in each record is randomly chosen. The *num_skips* parameter controls how many connections are sampled for a record, that is, what is the *bandwidth* or *depth* of the context.

3.4.1.6. Distance measurement

The algorithm generates a vector for each word (node or connection). With these vectors, it's feasible to discover which words are close to a given word, that is, which items exhibit a high similarity (cosine similarity):

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The similarity ranges from -1 to 1, with 1 representing the shortest distance (angle with 0 degrees) and -1 representing the longest distance (angle with 180 degrees).

The similarity measure is obtained for each event observed in the detection stage, by multiplying the normalized embedding of the node initiation a connection by the transposed embedding of the initiated connection (Carrasco & Sicilia, 2018).

3.4.1.7. Visual inspection

In order to measure accuracy, a rigorous performance measurement method was applied. Nevertheless, before that method was run, visual inspection of the results obtained in the training phase was useful to confirm the initial hypothesis (Carrasco & Sicilia, 2018). This inspection provides a deeper understanding of the results of the modeling exercise. Moreover, it allows to check whether clusters of nodes and connections are naturally formed, that is, whether normal behavior actually exists and whether nodes' behavior can be classified into groups (for example, workstations and servers, which should exhibit radically different behavior).

In order to plot embeddings, they were transformed into bidimensional vectors using tSNE (t-Distributed Stochastic Neighbor Embedding) algorithm (Van Der Maaten, 2014). tSNE (Van der Maaten & Hinton, 2012) is a method to reduce dimensionality that allows to display in a XY graph records of a high-dimensional dataset (Van Der Maaten, 2009). This algorithm is included in *scikit-learn* Python package. The tSNE configuration used is included in Table 7. Figure 8 shows the scatter plot associated with the optimal neural network configuration (*C1*), which suggests that network nodes exhibit similar behavior and can be classified into groups according to that behavior.

Parameter	Value
perplexity	50
n_components	2
init	pca
n_iter	5000

Table 7. tSNE parameters.

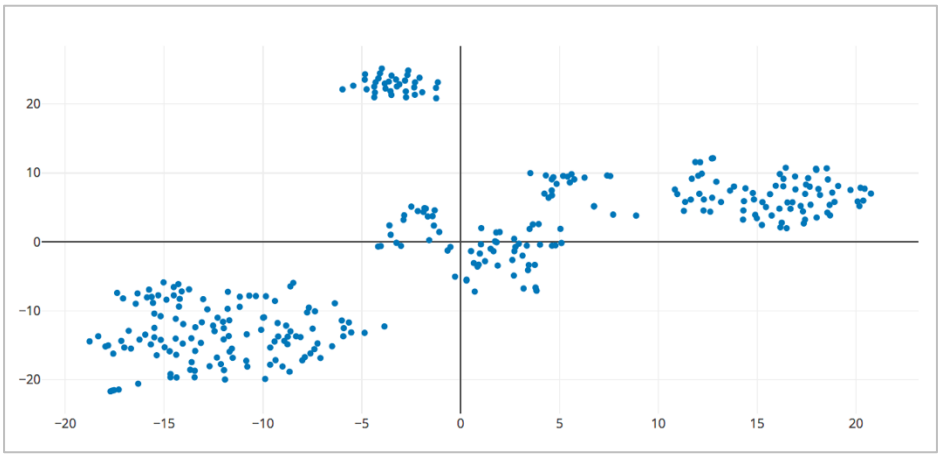

Figure 8. Scatter plot of obtained embeddings.

3.4.1.8. Performance evaluation criteria

Proposed evaluation criteria are based on popular metrics used in IDS research: precision and recall (Wu & Banzhaf, 2010). Their formulas are the following:

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

F-score, which is also commonly used in the IDS field, is derived from precision and recall:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

True positives (*tp*) represent events considered an attack that are an attack. False positives (*fp*) represent events wrongly tagged as an attack which are normal traffic. False negatives (*fn*) represent attack events erroneously tagged as normal. Precision represents the proportion of true positives detected among events tagged as attacks. Recall represents the proportion of attacks detected among all observed events.

3.4.1.9. Experiment setup

3.4.1.9.1. Training and test datasets

1,938,118 records (76.3%) were sampled to train the model. The training set included no attacks, as required in an unsupervised setting. 601,928 records (23.6%) were sampled to measure model accuracy.

280,645 records in the test set represented normal traffic, while 321,283 records were actual attacks.

3.4.1.9.2. Parametrization

Optimal neural network configuration (higher precision and higher recall, with recall having higher priority) was discovered through grid search.

Table 8 shows what parameters were considered for the grid search (Carrasco & Sicilia, 2018):

Parameter	Value
batch_size	# of records returned by microbatch function.
num_skips	# of context items for a target word.
valid_size	# of records used for validation.
num_steps	# of epochs.
embedding_size	# of embedding vector dimensions.

Table 8. Neural network parameters.

3.4.2. Semisupervised intrusion detection

3.4.2.1. Dataset

UNSW-NB15 was used to test proposed approach.

3.4.2.2. Feature engineering

UNSW-NB15 dataset contains 49 features that can be grouped into the following categories: IP addresses, categorical features, and numerical features. A different engineering technique was applied to each of these categories.

For source and destination IP addresses, each of the four octets was encoded as eight binary ([0, 1]) values.

One-Hot Encoding (OHE) was applied to categorical features. It produces a vector in which one single element is set to one (1), and the rest are set to zero. The length of the vector equals the number of unique values. Therefore, this encoding scheme creates orthogonal vectors, whose distance to all other vectors is the same. This property is required for categorical features in which there is no relationship between values, that is, all values are equally different from the rest. Table 9 and Table 10 show this encoding technique for two sample features (target port and network protocol, respectively). For unknown values in the test process, that is, those not found in the training dataset, a vector with all components set to zero was generated.

Target port value	Assigned Index	One-hot Encoded Value
0	0	100000000000
9999 (default)	1	010000000000
80	2	001000000000
25	3	000100000000

53	4	000010000000
111	5	000001000000
5190	6	000000100000
22	7	000000010000
21	8	000000001000
143	9	000000000100
6881	10	000000000010
179	11	000000000001

Table 9. Target port encoding.

Network protocol value	**Assigned Index**	**One-hot Encoded Value**
ospf	0	100000000
icmp	1	010000000
tcp	2	001000000
arp	3	000100000
udp	4	000010000
igmp	5	000001000
udt	6	000000100
rtp	7	000000010
esp	8	000000001

Table 10. Network protocol encoding.

For numerical features, the following logarithmic scale discretization formula was applied:

$$f(x) = \begin{cases} log_{10}(x), & x \neq 0 \\ 0, & x = 0 \end{cases}$$

Once discretized, these feature values were transformed into numerical vectors using the aforementioned One-Hot Encoding scheme.

Lastly, infrequent destination ports were identified in the dataset and relabeled with a default value (9999).

3.4.2.3. Feature selection

In proposed setting, three different feature selection approaches were tested.

In the first approach, all features were included. In the second approach, numerical features with more predictive power, IP addresses (source and target), and categorical features were selected.

Predictive power was obtained through xgboost's feature importance metric. This metric is calculated for a single decision tree by measuring the amount that each feature split point enhances performance in terms of the Gini index, weighted by the number of instances the node is responsible for. Obtained importance is then averaged across all decision trees in the model. Gini index is calculated as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

where C is the number of classes (two in proposed scenario, normal or attack), and p_i is the probability of an instance being classified to class i.

Most important features, as ranked by xgboost, are shown in Figure 9.

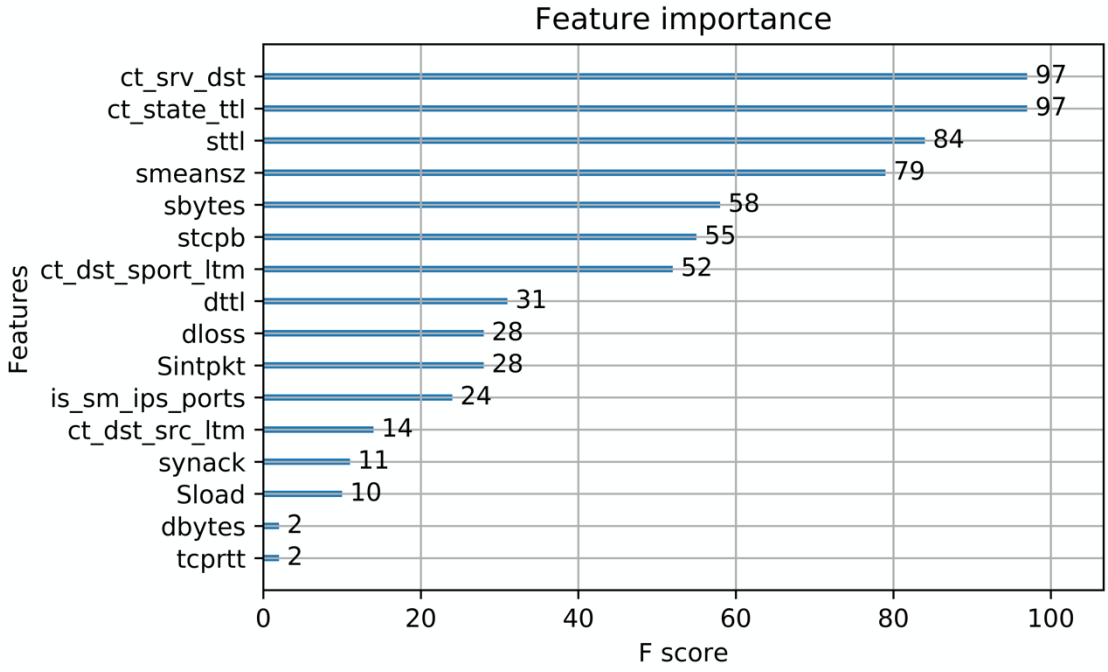

Figure 9. Top features by xgboost importance.

In the third approach, all features but IP addresses were selected. Table 11 enumerates the features included in each approach.

Approach	**Included features**
All features	srcip, dstip, dsport, proto, state, service, is_sm_ips_ports, ct_state_ttl, is_ftp_login, ct_ftp_cmd, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, Sload, Dload, Spkts, Dpkts, swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len, Sjit, Djit, Sintpkt, Dintpkt, tcprtt,

	synack, ackdat, ct_flw_http_mthd, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm
xgboost features	srcip, dstip, dsport, proto, state, service, is_sm_ips_ports, ct_state_ttl, is_ftp_login, ct_ftp_cmd, ct_srv_dst, sttl, smeansz, sbytes, stcpb, ct_dst_sport_ltm, dttl, dloss, Sintpkt, ct_dst_src_ltm, synack, Sload, dbytes, tcprtt
No IP addresses	dsport, proto, state, service, is_sm_ips_ports, ct_state_ttl, is_ftp_login, ct_ftp_cmd, dur, sbytes, dbytes, sttl, dttl, sloss, dloss, Sload, Dload, Spkts, Dpkts, swin, dwin, stcpb, dtcpb, smeansz, dmeansz, trans_depth, res_bdy_len, Sjit, Djit, Sintpkt, Dintpkt, tcprtt, synack, ackdat, ct_flw_http_mthd, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm

Table 11. UNSW-NB15 features included in each feature selection approach.

The motivation behind the second approach (letting xgboost select features) was to check whether excluding features with less predictive power would lead to higher performance. This has been investigated by other researchers using Linear Discriminant Analysis (Dahiya & Srivastava, 2018), with positive results. On the other hand, the motivation behind the third approach (excluding IP addresses) was to measure the impact on performance of training the model with frequent connections between known IP addresses. Lastly, timestamps were excluded in the three cases, because proposed detection method must not rely on attacks taking place at a given timeframe or in a given sequence.

3.4.2.4. Neural network design

TensorFlow (Abadi et al, 2016) is a machine learning framework created by Google. It allows to perform training and inference for neural networks. Keras is an open-source neural network Python package that leverages TensorFlow as backend. I

In this research, a deep autoencoder was implemented with Keras, using TensorFlow as backend engine. The design process implied setting the parameters enumerated in Table 12.

Parameter	**Description**
Layer 1 size	Size of the first hidden layer of the encoder.
Layer 2 size	Size of the second hidden layer of the encoder.
Layer 3 size	Size of the third hidden layer of the encoder, which represents the compressed input.
Batch size	# of records returned by microbatch function as input.
Epochs	# of epochs.

Table 12. Neural network parameters.

All configurations shared the structure of layers shown in Table 13.

Name	Type	Activation
Input	Dense	-
Hidden 1(encoder)	Dense	RELU
Hidden 2 (encoder)	Dense	RELU
Hidden 3 (compressed)	Dense	RELU
Hidden 2 (decoder)	Dense	RELU
Hidden 1 (decoder)	Dense	RELU
Output	Dense	sigmoid

Table 13. Autoencoders Layers Structure.

Lastly, Adam optimizer (Kingma & Ba, 2014) was selected to minimize loss function. The parameter values shown in Table 14 were set for proposed implementation.

Parameter	Symbol	Value
Learning rate	α	0.009
Forgetting factor for gradients	β_1	0.9
Forgetting factor for second moments of gradients	β_2	0.999
Small scalar used to prevent division by 0	ϵ	10^{-8}

Table 14. Adam Optimizer parameters.

3.4.2.5. Training and detection

The training process aimed at leveraging the proposed autoencoder to learn legitimate traffic patterns. For this purpose, the UNSW-NB15 dataset was split into a train dataset, containing normal instances, and a test dataset, containing both normal and attack instances. Table 15 shows the size of each dataset and their respective proportion of normal and attack instances.

Dataset	Size	# normal instances (%)	# attack instances (%)
Train	1,897,478	1,897,478 (100%)	0 (0%)
Test	642,566	321,283 (50%)	321,283 (50%)

Table 15. Train and test datasets.

The assumption was that, in the detection process, attack inputs would be rare, compared to normal activity. Therefore, they would exhibit a higher reconstruction error than normal inputs. In other

words, low reconstruction errors would be associated with normal traffic, while high reconstruction errors would be associated with attack traffic. This approach is depicted in Figure 10.

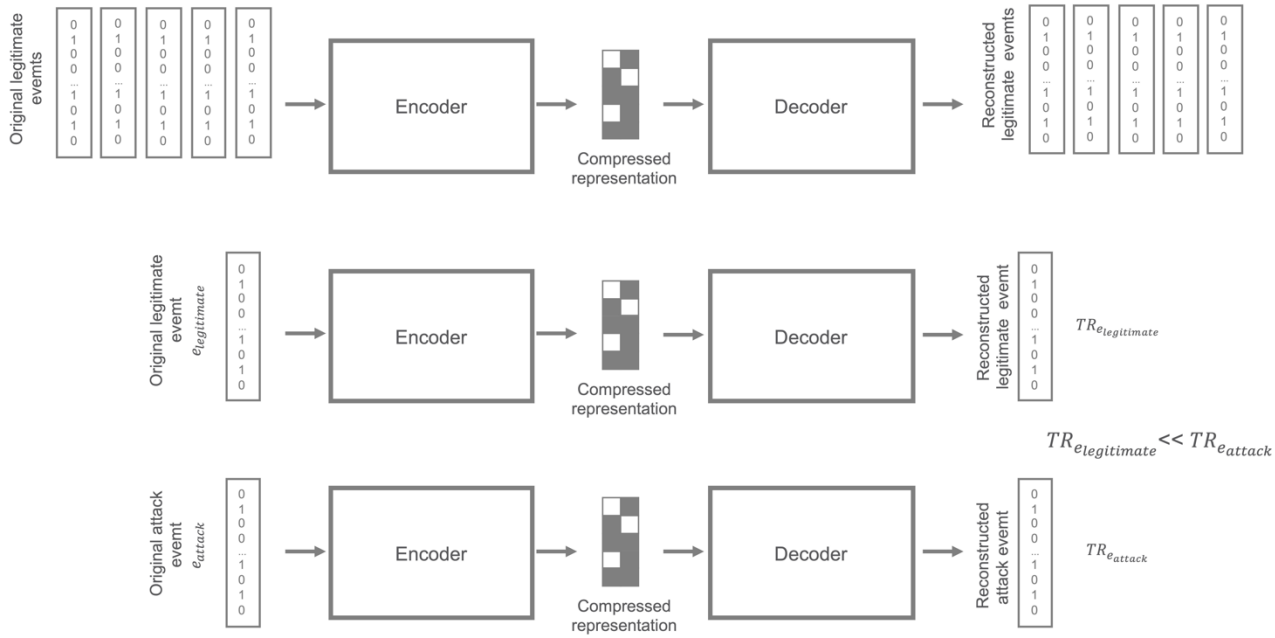

Figure 10. Training and detection approach.

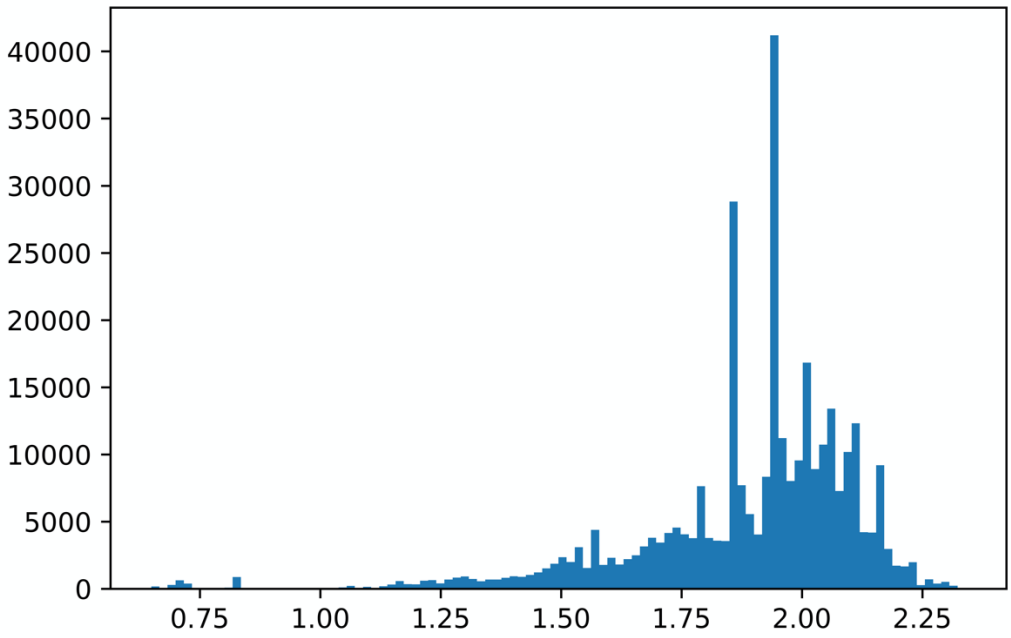

Figure 11. Transformed Reconstruction Errors for normal instances.

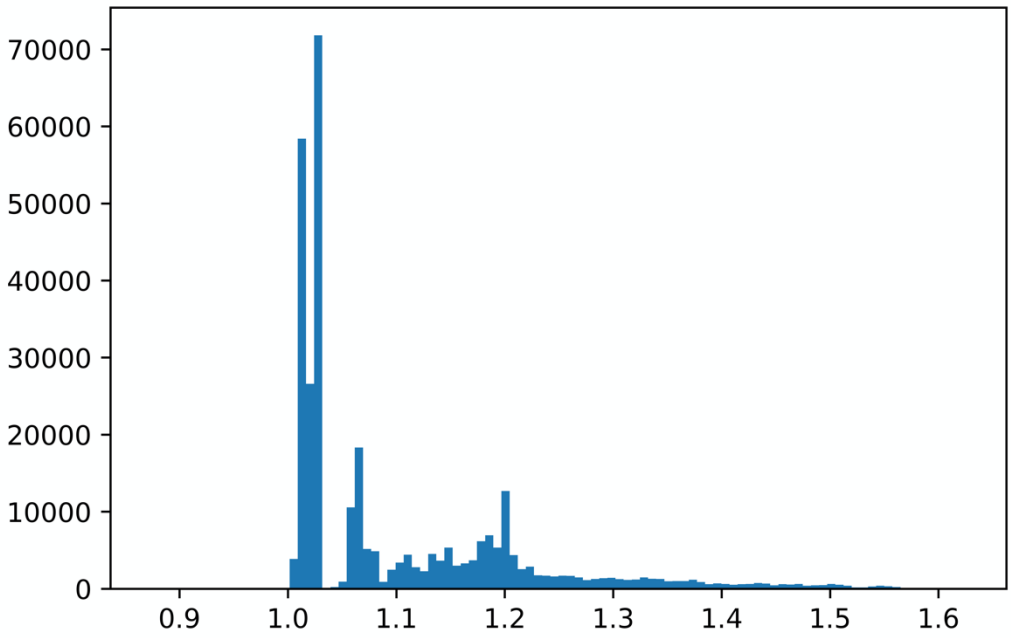

Figure 12. Transformed Reconstruction Errors for attack instances.

Therefore, setting a threshold allows proposed system to accurately classify inputs as normal or attack traffic. In order to find an accurate and stable threshold for each autoencoder configuration, the obtained reconstruction error was transformed by taking (and negating) log_{10} of its value, which represents its order of magnitude. The resulting value was designated as Transformed Reconstruction Error (TRE):

$$TRE = -log_{10}\left(\frac{1}{N} \times \sum_{i=1}^{N}(x_i - \tilde{x}_i)^2\right)$$

where N is input length.

Figures 11 and 12 show histograms of normal and attack Transformed Reconstruction Errors for an example autoencoder configuration.

3.4.2.6. Performance evaluation criteria

In the field of IDS, the following metrics are commonly used for model comparison: accuracy, precision, recall and F-score:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where true positives (tp) refer to attack traffic observations correctly classified, false positives (fp) are legitimate traffic observations incorrectly classified as attacks, false negatives (fn) are attack traffic observations incorrectly classified as legitimate traffic, and true negatives (tn) are legitimate traffic observations correctly classified.

3.4.3. Cross-domain malicious behavior detection

3.4.3.1. Datasets

The following datasets were used: UNSW-NB15, Paysim1 and IoT-23.

3.4.3.2. Entity definition

Typically, behavior modeling techniques take the subject performing actions as the entity whose behavior has to be modeled (Kim, Park, Kim, Cho, & Kang, 2019) (Blaise, Bouet, Conan, & Secci, 2020). In this research, however, the action being performed was used as the entity. This decision was made to avoid detecting attacks or fraud by simply spotting subjects (attackers or fraudsters) that were not present in the training phase, which only contains legitimate activity.

Table 16 shows what features were used to form the entity in each dataset:

Dataset	Entity features	Description
UNSW-NB15	proto, service	Connection, represented by protocol and service features.
Paysim1	type	Transaction type.
IoT-23	proto, service	Connection, represented by protocol and service features.

Table 16. Features used as entity in each dataset.

3.4.3.3. Feature engineering

In this research, a feature engineering technique that translates events (or transactions) into words was developed. For each event, the entity (action being performed) was extracted. Each entity represented a document. Content (feature values) of all events of a given entity represented words of that document. The resulting corpus was a set of entities (documents) whose documents' words were the feature values present in their events.

Target datasets contained both numerical and categorical features. Therefore, data transformation was needed to ensure that the resulting item could be translated into a word regardless of the feature type. For this purpose, a two-stage encoding technique was used.

In the first stage, feature values were prefixed with the feature name. This was done to ensure that same values present in different features were translated into different words in the documents where they appear. In the second stage, numerical values were discretized using a logarithmic scale or by

dividing its value by ten (10), depending on their range. For numerical features with a range that spans more than one order of magnitude, the logarithmic scale was used, while the remaining numerical features were divided by ten.

Logarithmic scale is applied with the following formula:

$$f(x) = log10\left(\frac{x}{a}\right) \times 10 \qquad x \geq a$$

$$f(x) = 0 \qquad x \leq a$$

where a is parameter that is set to percentile 1 if all values are lower than 1, or to 1 otherwise.

3.4.3.4. Scoring

The initial purpose of the topic modeling algorithm was to discover latent topics in a corpus of documents. This translates into two probability distributions: the probability of a document (entity) containing a topic, and the probability of a word (feature value) representing a topic. These two probability distributions are represented by matrices E and T, respectively.

In this research, these probability distributions were used to compute the probability of an event (p) representing legitimate behavior. The following two probability metrics were tested to assess which one exhibited higher performance:

$$p_{min} = \min\{E_i \cdot T_1, \dots, E_i \cdot T_N\}$$

$$p_{sum} = \sum_{j=0}^{N} E_i \cdot T_j$$

$$1 \leq i \leq M$$

where M is the number of unique entities, N is the number of feature values in each event (except from entity-related features), E_i is the document-topic probability vector for document (entity) i, and T_j is the word-topic probability vector for word (feature value) j.

3.4.3.5. Performance evaluation

Performance evaluation was done using common metrics in the field of intrusion detection systems (Elhamahmy, Elmahdy, & Saroit, 2010): precision, recall, accuracy and F-score.

Accuracy is the proportion of the total number of the correct detections (both legitimate activity and attacks) relative to the dataset size. F-score (F1) measures the balance between precision and recall.

Their formulas are as follows:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where tp is the number of attacks correctly detected, tn is the number of legitimate events correctly classified, fp is the number of legitimate events incorrectly classified as attacks, and $fn$is the number of attacks incorrectly classified as legitimate activity.

3.4.3.6. Training and test datasets

Target datasets were split into legitimate and malicious activity. A subset of the legitimate activity and all the malicious activity (with a proportion of 50% each) was used for testing, and the remaining legitimate activity was used for training. This split approach (with the same number of observations for both classes) ensures that performance evaluation metrics remain consistent and reliable, with the same number of observations for both classes.

3.4.3.6.1. UNSW-NB15

Table 17 shows how the UNSW-NB15 dataset was split for training and testing:

Subset	Number of observations
Legitimate	2,218,763
Malicious	321,283
Train	1,897,480
Test (legitimate)	321,283
Test (malicious)	321,283
Test (all)	642,566

Table 17. UNSW-NB15 train and test datasets.

Table 18 shows how features were transformed, using the feature engineering method described in section 3.3.

Discretization technique	Features
Logarithmic scale	dur, sbytes, dbytes, sloss, dloss, Sload, Dload, Spkts, Dpkts, stcpb, dtcpb, smeansz, dmeansz, res_bdy_len, Sjit, Djit, Sintpkt, Dintpkt, ackdat, tcprtt, synack.
Divide by 10	sttl, dttl, swin, dwin, trans_depth, ct_srv_src, ct_srv_dst, ct_dst_ltm, ct_src_ltm, ct_src_dport_ltm, ct_dst_port_ltm, ct_dst_src_ltm.
None (categorical)	is_sm_ips_ports, ct_state_ttl, ct_flw_http_mthd, is_ftp_login, ct_ftp_cmd.

Table 18. Feature transformations for UNSW-NB15 dataset.

3.4.3.6.2. Paysim1

Table 19 shows how the Paysim1 dataset was split for training and testing:

Subset	Number of observations
Legitimate	6,354,407
Malicious	8,213
Train	6,346,194
Test (legitimate)	8,213
Test (malicious)	8,213
Test (all)	16,426

Table 19. Paysim1 train and test datasets.

Table 20 shows how features were transformed, using the feature engineering method described in section 3.3.

Discretization technique	Features
Logarithmic scale	amount, newbalanceOrig, oldbalanceDest, newbalanceDest, oldbalanceOrig
Divide by 10	-
None (categorical)	-

Table 20. Feature transformations for Paysim1 dataset.

3.4.3.6.3. IoT-23 MC11

Table 21 shows how the IoT-23 MC11 dataset was split for training and testing:

Subset	Number of observations
Legitimate	469,275
Malicious	539,473
Train	234,638
Test (legitimate)	234,637
Test (malicious)	539,473
Test (all)	774,110

Table 21. IoT-23 MC11 train and test datasets.

Table 22 shows how features were transformed, using the feature engineering method described in section 3.3.

Discretization technique	Features
Logarithmic scale	duration, orig_bytes, resp_bytes, orig_ip_bytes, resp_ip_bytes
Divide by 10	orig_pkts, resp_pkts
None (categorical)	conn_state, history

Table 22. Feature transformations for IoT-23 MC11 dataset.

3.4.3.7. Model parametrization

Gensim is a robust, open-source, scalable and platform-independent Python library for topic modeling ("Gensim: topic modelling for humans", n.d.). Gensim's LDA implementation was used in this research.

This implementation provides multiple parameters for customization. The two key parameters influencing performance, based on tests performed, were the number of topics and the number of passes.

Table 23 describes these parameters and their default values:

Name	Description	Default value
num_topics	# of requested latent topics to be extracted from the training corpus.	100
passes	# of passes through the corpus during training.	1

Table 23. LDA model parameters in scope.

3.4.4. Supervised fraud detection alert optimization

3.4.4.1. Dataset

The fraud dataset described in the section devoted to datasets was used.

3.4.4.2. Statement of the problem

Multiple neural network implementations were subject to a training exercise. The goal for these neural networks was to learn how to discard false alerts, in order to reduce the amount of alerts requiring manual review (Carrasco & Sicilia, 2020).

3.4.4.3. Feature engineering

Both numeric and categorical features are present in the dataset used for testing. All features were transformed into fixed-length binary vectors. The actual transformation procedures applied to features were as follows (Carrasco & Sicilia, 2020):

Binary. Applied to categorical features. Values were indexed with a positive (greater than or equal to zero) integer. The obtained index was then translated a binary vector. Vector length equaled the amount of bits required to represent the highest binary value.

Table 24 shows an example of the results for a feature with 5 values.

$$B = \{B_1, B_2, \dots, B_5\}$$

Label	# Records	Percentage (%)
B_1	0	000
B_2	1	001
B_3	2	010
B_4	3	011
B_5	4	100

Table 24. Binary encoding example.

OHE (One-Hot Encoding). Applied to categorical features. Values were indexed with a positive (greater than zero) integer. The obtained index was then translated a binary vector. Vector length is associated with the number of unique feature values. All vector values (bits) are set to zero, except from the value at the position represented by the index (set to one).

Table 25 shows an example of the results for a feature with 5 values.

$$O = \{O_1, O_2, \dots, O_5\}$$

Feature value	Index	Encoded value
O_1	0	00001
O_2	1	00010
O_3	2	00100
O_4	3	01000
O_5	4	10000

Table 25. One Hot Encoding example.

Binning. Applied to numeric features. Feature values were assigned to bins. Bins were defined by applying thresholds derived from the feature value range.

Table 26 shows which bins applied to numeric features.

Feature name	Bin thresholds
Score	[-∞, 10, 20, 30, 40, 50, 60, 70, 80, 90, ∞]
Day of month	[-∞, 10, 20, ∞]
Hour	[-∞, 4, 8, 12, 16, 20, ∞]
Amount	[-∞, 10, 100, 1000, 10000, 100000, ∞]

Table 26. Bin thresholds.

3.4.4.4. Architecture design criteria

Both supervised and unsupervised settings were considered in this research (Carrasco & Sicilia, 2020).

MLP (Multi Layer Perceptron) and CNN (Convolutional Neural Networks) are artificial neural networks used for classification (supervised learning) of observations with fixed length. DAE (Deep Autoencoders) leverage unsupervised learning, which requires no labeled data. In other words, Deep Autoencoders learn without prior knowledge of fraud patterns.

Table 27 shows each architecture parameters.

Label	# Records
MLP	# of hidden layers
	Hidden layer sizes
CNN	# of convolutional layers
	Input layer shape
DAE	# of hidden layers
	Hidden layer sizes
Common (MLP, CNN, DAE)	Batch size
	Learning rate
	# of epochs

Table 27. Neural network key parameters.

3.4.4.5. Performance evaluation criteria

The most popular evaluation metrics for classification tasks are precision and recall. They are also leveraged in the fraud detection domain (Kumari & Mishra, 2019).

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

where true positives (tp) refer to alerts flagged for fraudulent transactions, false positives (fp) refer to alerts flagged for legitimate transactions, false negatives (fn) refer to the lack of a triggered alert for fraudulent transactions, and true negatives (tn) refer to legitimate transactions that triggered no alert.

Precision and recall values were calculated for the assessed neural network architectures (Carrasco & Sicilia, 2020). The AUC (Area Under the Curve) was obtained from the ROC (Receiver Operator Characteristic). For supervised settings (MLP, CNN), a threshold representing the probability beyond which an observation would be assigned to the fraud class was set. For the unsupervised setting (DAE), a threshold representing the reconstruction error beyond which an observation would be assigned to the fraud class was set.

Reconstruction error was calculated as follows:

$$reconstruction\ error = -log_{10}\left(\frac{1}{N}\sum_{i=0}^{N}(x_i - \tilde{x}_i)^2\right)$$

where N equals input length, x_i represents each input dimension, and $\tilde{x}_i$ is represents each output dimension. Output represents the reconstructed (after compression and decompression) input.

The effective alert reduction rate was calculated for each configuration based on the aforementioned performance metrics (Carrasco & Sicilia, 2020). The rate, in this scenario, represents the percentage of alerts that can be automatically discarded as false positives, requiring no manual review by a human fraud expert. The rate is described together with a rate of misclassification, that is, how much actual fraud is missed due to the proposed approach tagging it as a false positive.

3.4.4.6. Training and test datasets

For both supervised and unsupervised settings, data was split into training and testing datasets. For supervised settings (MLP, CNN), 80% of the alerts were included in the training dataset, and 20% of the alerts were included in the testing dataset. In the case of the Deep Autoencoder, it was trained with 201,995 false positives. For testing, 48,816 false positive alerts and 195,265 true alerts were used as input.

Table 28 includes the number of alerts (for the training and testing datasets) in each configuration (Carrasco & Sicilia, 2020):

Identifier	Test dataset size	Test dataset size (legitimate)	Test dataset size (fraud)
MLP2BE256H82	89,216	50.168	39,048

MLP2BE128H164	89,216	50.168	39,048
MLP2OH128H918	89,216	49,624	39,592
MLP3OH256H512	89,216	49,907	39,309
MLP3BE256H512	89,216	50,117	39,099
MLP3OH256H918	89,216	49,911	39,305
CNN2OH100LR10-3	89,216	50,231	38,985
CNN2OH100LR10-1	89,216	50,137	39,079
DAE4BE256	244,081	48,816	195,265
DAE4OH256	244,081	48,816	195,265

Table 28. MLP neural network architectures.

3.4.4.7. Parametrization

Grid search was used to adjust the parameters initially set for each architecture. These initial parameters are included in Tables 29, 30, and 31 (Carrasco & Sicilia, 2020). Grid search was useful to measure how much each of these parameters was influencing performance.

Identifier	Hidden layers	Encoding (categorical)	Encoding (numerical)	Batch size	Learning rate	Hidden layer size	Epochs
MLP2BE256H82	2	Binary	Binning	256	10^{-3}	82	250
MLP2BE128H164	2	Binary	Binning	128	10^{-2}	164	1,000
MLP2OH128H918	2	OHE	Binning	128	10^{-2}	918	250
MLP3OH256H512	3	OHE	Binning	256	10^{-3}	512	250
MLP3BE256H512	3	Binary	Binning	256	10^{-3}	512	250
MLP3OH256H918	3	OHE	Binning	256	10^{-3}	918	250

Table 29. MLP parametrization.

Identifier	Convolutional layers	Encoding (categorical)	Encoding (numerical)	Batch size	Learning rate	Input layer shape	Epochs
CNN2OH100LR10-3	2	OHE	Binning	100	10^{-3}	31x31	25,000
CNN2OH100LR10-1	2	OHE	Binning	100	10^{-1}	31x31	25,000

Table 30. CNN parametrization.

Identifier	Hidden layers	Encoding (categorical)	Encoding (numerical)	Batch size	Learning rate	Hidden layer sizes	Epochs
DAE4BE256	4	Binary	Binning	256	10^{-2}	55 (1) 37 (2)	250
DAE4OH256	4	OHE	Binning	256	10^{-2}	612 (1) 408 (2)	500

Table 31. DAE parametrization.

4. Results

4.1. Introduction

Obtained results in each of this research work's developments are presented and discussed in this section.

4.2. Unsupervised intrusion detection

An attack was identified in proposed approach by measuring the cosine similarity between the network node and the connection observed in the event. The threshold was zero. Therefore, if the similarity was negative, the event was classified as an attack. In the same way, a positive value represented normal activity (Carrasco & Sicilia, 2018).

The total number of distinct network nodes and connections was 2,830 in the test dataset. Each of these combinations was evaluated to determine whether they represented legitimate or malicious activity. The results obtained with assessed configurations are enumerated in Table 32.

	batch_size	num_skips	num_steps	embedding_size	True positives	False positives	True negatives	False negatives	% of legitimate traffic (training)	Average loss	Precision	Recall
C1	**8**	**2**	**90,000**	**4**	**1,122**	**9**	**1,454**	**245**	**98.14**	**0.197208**	**0.9920**	**0.8207**
C2	16	2	90,000	4	1,103	7	1,456	264	99.01	0.360984	0.9936	0.8068
C3	32	2	90,000	4	1,109	8	1,455	258	99.18	0.626274	0.9928	0.8112
C4	16	4	90,000	4	1,114	7	1,456	253	98.48	0.364535	0.9937	0.8149
C5	16	4	90,000	8	1,119	9	1,454	248	97.55	0.367023	0.9920	0.8185
C6	16	4	90,000	16	1,118	14	1,449	249	97.79	0.366863	0.9876	0.8178
C7	16	4	180,000	4	1,103	6	1,457	264	98.89	0.355693	0.9945	0.8068
C8	16	4	360,000	4	1,096	7	1,456	271	99.30	0.359761	0.9936	0.8017
C9	32	4	360,000	8	1,088	5	1,458	279	99.30	0.624516	0.9954	0.7959

Table 32. Configuration parameters and results.

As seen in the table (Carrasco & Sicilia, 2018), performance was not influenced by batch size, embedding size or skips length. On the other hand, convergence was achieved with a low number of epochs, possibly due to the (small) size of the dataset.

C1 was the optimal configuration (Carrasco & Sicilia, 2018). Precision and recall values for this configuration were 99.20% and 82.07%, respectively. F-score was 89.82%, accuracy was 91.02%, and false positive rate was 0.61%. No knowledge of attack patterns was required to achieve these figures. On the other hand, while this precision value was outperformed by other configurations, the recall of those configurations was suboptimal.

UNSW-NB15 dataset has been extensively chosen to measure intrusion detection performance (Bamakan, Wang, & Shi, 2017). Table 33 shows how proposed approach's performance compares to other methods (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017):

Technique	Dataset	Requires prior knowledge	False positive rate	Accuracy	Detection rate	Precision
Skip-gram	**UNSW-NB15**	**No**	**0.61%**	**91.02%**	**82.07%**	**99.20%**
MLP-BP	KDD Cup'99	Yes	**8.51%**	-	81.96%	90.58%
RBF	KDD Cup'99	Yes	**1.2%**	-	99.2%	-
SOM	KDD Cup'99	No	**1.38%**	-	90.4%	-
ART	KDD Cup'99	No	**3.86%**	-	96.13%	-
LSTM-RNN	KDD Cup'99	No	**10.04%**	96.93%	98.88%	-
DBN	KDD Cup'99	Yes	**0.76%**	93.49%	-	99.18%
Ramp-KSVCR	UNSW-NB15	Yes	**2.46%**	93.52%	98.68%	98.60%

Table 33. Performance comparison with other neural network-based techniques.

Neither precision nor accuracy were disclosed for MLP-BP (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017). The value shown for precision was calculated from disclosed rates (detection rate, false positives, false negatives). The same exercise is done for SOM (Kayacik, Zincir-Heywood, & Heywood, 2007), using detection and false positive rate. Neither precision nor accuracy were disclosed (or could be obtained from disclosed rates) for ART (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017).

False positive rate was not included for LSTM-RNN in the study comparing methods (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson, 2017), but was found in the original research work (Kim, Kim, Thu, & Kim, 2016). For DBN implementation, supplied true positive and false positive rates allowed to calculate precision.

Proposed approach outperforms the aforementioned methods in key metrics like false positive rate. Keeping this rate as low as possible is a challenge in the field of IDS (Pietraszek, 2004). Proposed approach achieves a low false positive rate with notable accuracy and detection rate values. Moreover, these figures are achieved without training the algorithm with attack data. The algorithm is also compact in terms of input features, as it only requires four features that can be easily extracted from any network source.

The dataset used for testing has been recently produced (2015) and correct issues found in other datasets (Gogoi, Bhuyan, Bhattacharyya, & Kalita, 2012) used by the methods with which proposed approach is compared. These other datasets contain records that exhibit redundancy, which introduces bias (Gogoi, Bhuyan, Bhattacharyya, & Kalita, 2012), and leads to artificially high measures of accuracy (Buczak & Guven, 2015).

4.3. Semisupervised intrusion and fraud detection

The set of tested neural network configurations are presented, and the optimal configuration is identified. Obtained results are compared to similar techniques and discussed.

A set of configurations with varying values for these parameters (shown in Table 34) were defined, in order to measure their influence on performance levels.

Optimal parameters were discovered by measuring the performance obtained by each configuration in Table 34. In proposed setting, *optimal* meant higher accuracy. For each of the feature selection approaches described in section 3.3, three parameter configurations were tested. These configurations were designed to measure the influence of batch size and neural network layer sizes. Table 34 includes tested configurations and their performance levels.

Id.	Feature selection	Input size	Layer 1 size	Layer 2 size	Layer 3 size	Epochs	Batch size	Precision	Recall	Accuracy
C1	All features	305	16	4	1	10	64	0.9344	0.9743	0.9529
C2	All features	305	32	4	1	10	32	0.9365	0.9668	0.9506
C3	All features	305	32	4	1	10	64	0.9396	0.9826	0.9597
C4	Random forest	207	16	4	1	10	64	0.9322	0.9655	0.9477
C5	Random forest	207	32	4	1	10	32	0.9584	0.9999	0.9783
C6	Random forest	207	32	4	1	10	64	0.9462	0.9810	0.9626
C7	No IP addresses	241	16	4	1	10	64	0.9601	0.9577	0.9589
C8	No IP addresses	241	32	4	1	10	32	0.9048	0.8701	0.8893
C9	No IP addresses	241	32	4	1	10	64	0.9458	0.9563	0.9507

Table 34. Parameter configurations and results.

Table 34 shows that neither layer size structure nor batch size strongly influenced performance levels, regardless of the feature selection approach. On the other hand, ten epochs were set for all configurations, as more epochs didn't drove any performance improvement. This, however, might not hold for bigger datasets. As for the feature selection approach, including IP addresses in the training process didn't consistently improve performance: configurations excluding them (C7) performed better than those that included them (C1), in terms of accuracy. However, selecting features based on the importance attributed by xgboost slightly enhanced the metrics, as observed for configurations C5 and C2.

C5 was the optimal configuration, with 95.84% precision and 99.99% recall, leading to an F-score of 97.87%. Accuracy was 97.83%, and the FPR 4.33%.

Table 35 summarizes how proposed system's performance compares to previous research. Proposed system outperformed all other methods in terms of detection rate (recall) and obtained comparable

performance in other metrics like accuracy and precision. S-NDAE (Shone, Ngoc, Phai, & Shi, 2018) and SSAE+SVM (Yan & Han, 2018) techniques obtained higher accuracy, but they were tested against a dataset with less recent attacks and unrealistic traffic, as reported by other authors (Tavallaee, Bagheri, Lu, & Ghorbani, 2009). Both methods are based on a supervised setting, which fully relies on labelled data, while proposed approach is semisupervised (labeled data is only used for feature selection). On other hand, the system based on Taguchi method and stacked sparse autoencoders obtained higher accuracy and precision, but it is also based on a supervised setting.

Technique	Autoencoder	Dataset	Supervised	False Positive Rate	Accuracy	Detection rate	Precision
Proposed approach	Yes	UNSW-NB15	No	4.33%	97.83%	99.99%	95.84%
Ramp-KSVCR (Bamakan, Wang, & Shi, 2017)	No	UNSW-NB15	Yes	2.46%	93.52%	98.68%	98.60%
DBN+SVM (Marir, Wang, Feng, Li, & Jia, 2018)	No	UNSW-NB15	Yes	-	-	97.21%	90.47%
MDPCA-DBN (Yang, Zheng, Wu, Niu, & Yang, 2019)	No	UNSW-NB15	Yes	17.15%	90.21%	96.22%	87.30%
S-NDAE (Shone, Ngoc, Phai, & Shi, 2018)	Yes	KDD	Yes	2.15%	97.85%	97.85%	99.99%
Self-taught learning (STL) (Javaid, Niyaz, Sun, & Alam, 2016)	Yes	NSL KDD	Yes	-	88.39%	95.95%	85.44%
Stacked Dilated Convolutional Autoencoders (Yu, Long & Cai, 2017)	Yes	CTU-UNB / Contagio-CTU-UNB	Yes	-	-	88.80%	90.65%
SSAE+SVM (Yan & Han, 2018)	Yes	NSL KDD	Yes	0.13%	99.35%	99.01%	-
Deep Neural Net Ensemble (Ludwig, 2017)	Yes	NSL KDD	Yes	-	-	92.00%	93.00%
Taguchi + stacked sparse autoencoders (Karim, Güzel, Tolun, Kaya, & Çelebi, 2018)	Yes	UNSW-NB15	Yes	-	99.70%	-	99.70%
Autoencoder (Yousefi-Azar, Varadharajan, Hamey, & Tupakula, 2017)	Yes	KDD	No	-	83.84%	-	-

Table 35. Performance comparison.

4.4. Cross-domain malicious behavior detection

Based on the proposed model parametrization options, configurations shown in Table 36 were tested for the three datasets under consideration. For each configuration, the parameters used and performance evaluation metrics are displayed.

Target dataset	Number of topics	Number of passes	Probability metric	Precision (legitimate)	Precision (malicious)	Recall (legitimate)	Recall (malicious)	Accuracy	F1-score (legitimate)	F1-score (malicious)
UNSW-NB15	5	1,000	p_{min}	0.9707	0.9458	0.9443	0.9715	0.9579	0.9573	0.9585
UNSW-NB15	5	1,000	p_{sum}	0.7883	0.7041	0.6535	0.8245	0.7390	0.7146	0.7596
UNSW-NB15	5	25,000	p_{min}	0.9583	0.9485	0.9480	0.9588	0.9534	0.9531	0.9536
UNSW-NB15	5	25,000	p_{sum}	0.7984	0.7008	0.6423	0.8378	0.7401	0.7119	0.7632
UNSW-NB15	**10**	**25,000**	$\boldsymbol{p_{min}}$	**0.9719**	**0.9454**	**0.9438**	**0.9727**	**0.9582**	**0.9576**	**0.9588**
UNSW-NB15	10	25,000	p_{sum}	0.8162	0.7456	0.7136	0.8393	0.7765	0.7615	0.7897
Paysim1	5	1,000	p_{min}	0.7775	0.9023	0.9202	0.7366	0.8284	0.8429	0.8111
Paysim1	5	1,000	p_{sum}	0.9916	0.7073	0.5883	0.9950	0.7917	0.7385	0.8269
Paysim1	5	25,000	p_{min}	0.7788	0.9067	0.9241	0.7375	0.8308	0.8453	0.8134
Paysim1	5	25,000	p_{sum}	0.9916	0.7080	0.5896	0.9950	0.7923	0.7395	0.8273
Paysim1	**10**	**25,000**	$\boldsymbol{p_{min}}$	**0.7648**	**0.9414**	**0.9560**	**0.7060**	**0.8310**	**0.8498**	**0.8068**
Paysim1	10	25,000	p_{sum}	0.9916	0.7075	0.5887	0.9950	0.7919	0.7388	0.8270
IoT-23 MC11	5	1,000	p_{min}	0.9998	0.9291	0.8245	0.9999	0.9468	0.9037	0.9632
IoT-23 MC11	**5**	**1,000**	$\boldsymbol{p_{sum}}$	**0.9998**	**0.9296**	**0.8259**	**0.9999**	**0.9472**	**0.9046**	**0.9635**
IoT-23 MC11	5	25,000	p_{min}	0.9998	0.9291	0.8245	0.9999	0.9468	0.9037	0.9632
IoT-23 MC11	5	25,000	p_{sum}	0.9998	0.9296	0.8259	0.9999	0.9472	0.9046	0.9635
IoT-23 MC11	10	25,000	p_{min}	0.9998	0.9291	0.8245	0.9999	0.9468	0.9037	0.9632
IoT-23 MC11	10	25,000	p_{sum}	0.9998	0.9291	0.8245	0.9999	0.9468	0.9037	0.9632

Table 36. Parameter configurations and results.

Obtained results suggest that performance is not heavily influenced by the number of topics or algorithm's passes, even though optimal configurations were achieved with different parameter values in the three datasets.

As for the probability metric, p_{min} leads to poor performance in UNSW-NB15 and Paysim1 datasets but delivers a comparable (and slightly higher) accuracy in IoT-23 MC11. In fact, all configurations tested with IoT-23 MC11 led to similar performance. This could be explained by the stationarity behavior of IoT devices, and the fact that most attack traffic belongs to port scanning activities, which differ significantly from legitimate traffic. Therefore, legitimate traffic can be easily modeled by LDA, leading to high detection performance regardless of the parameters used.

Performance achieved for UNSW-NB15 and IoT-23 MC11 is notably higher than the one obtained for Paysim1. One possible reason for this is that the former two contain a richer feature set, leading to greater discriminative capabilities when topic modeling is applied. Also, UNSW-NB15 metrics are more uniform across classes (legitimate, malicious) than those for Paysim1 or IoT-23 MC11. Again, the nature of the datasets and their features could partially explain this behavior. Lastly, proposed method seems to perform better when classifying network traffic, but still obtains acceptable results for the other two datasets.

Figures 13, 14 and 15 show the ROC curves obtained for the optimal configuration in each dataset.

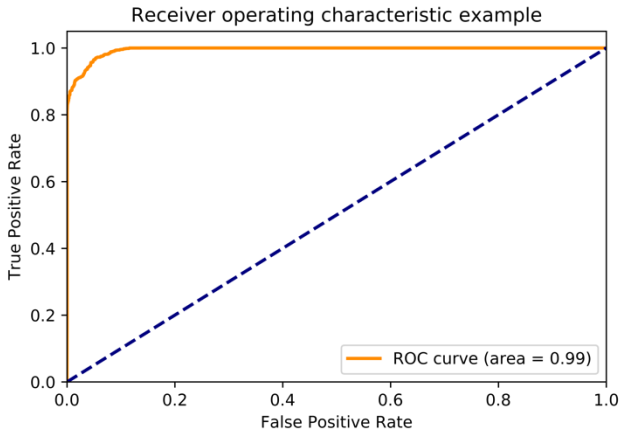

Figure 13. ROC curve for dataset UNSW-NB15.

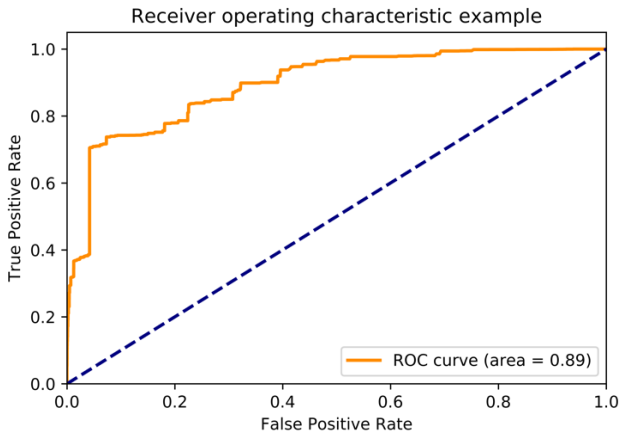

Figure 14. ROC curve for dataset Paysim1.

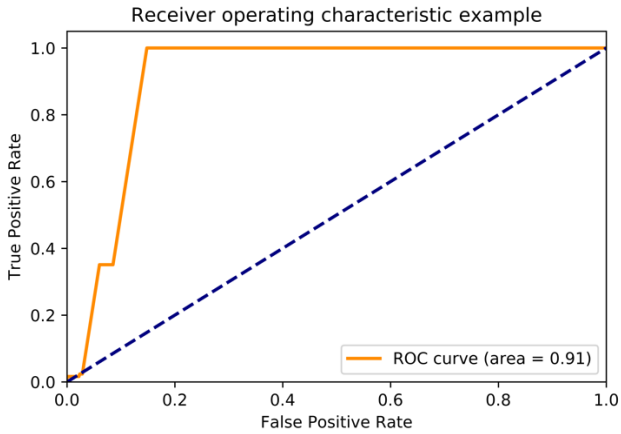

Figure 15. ROC curve for dataset IoT-23 MC11.

Method	Dataset	Prior knowledge	Accuracy	Precision	Recall
Proposed method	UNSW-NB15	**No**	**95.82%**	**95.86%**	**95.82%**
Autoencoder (Choi, Kim, Lee, & Kim, 2019)	KDD Cup 99	No	91.70%	-	-
Averaged One Dependence Estimator (AODE) (Nawir, Amir, Yaakob, & Lynn, 2019)	UNSW-NB15	Yes	97.26%	-	-
Support Vector Machine (SVM) + Recursive Feature Elimination, Random Forest (Meftah, Rachidi, & Assem, 2019)	UNSW-NB15	Yes	82.11%	-	-
Support Vector Machine (SVM) + Scaling method (Jing & Chen, 2019)	UNSW-NB15	Yes	85.99%	-	-

Table 37. Performance comparison for network attacks.

Method	Dataset	Prior knowledge	Accuracy	Precision 0 = legitimate 1 = fraud	Recall 0 = legitimate 1 = fraud
Proposed method	Paysim1	**No**	83.21%	77.92% (0)	92.68% (0)
				90.97% (1)	73,74% (1)
				84.45%	**83.21%**
Decision tree (Shahed, Ibrahim, & Akter, 2019)	Paysim1	Yes	94.04%	94.00% (0)	100.00% (0)
				98.00% (1)	**43.00% (1)**

Support Vector Machine (Shahed, Ibrahim, & Akter, 2019)	Paysim1	Yes	96.14%	96.00% (0) 98.00% (1)	100.00% (0) **64.00% (1)**
Artificial Neural Network (Shahed, Ibrahim, & Akter, 2019)	Paysim1	Yes	97.83%	98.00% (0) 92.00% (1)	99.00% (0) 87.00% (1)
Undercomplete autoencoder (Misra, Thakur, Ghosh, & Saha, 2020)	Paysim1	No	99.94%	**85.34%**	**80.15%**

Table 38. Performance comparison for payments fraud.

Method	**Dataset**	**Prior knowledge**	**Accuracy**	**Precision**	**Recall**
Proposed method	IoT-23	No	95.82 %	**95.05%**	**94.72%**
DEMISe (Parker, Yoo, Asyhari, Chermak, Jhi, & Taha, 2019)	Aegean Wi-Fi Impersonation Attack Detection	Yes	98.04 %	-	99.07%
BotFP-Clus ($\epsilon = 0.5b$) (Blaise, Bouet, Conan, & Secci, 2020)	CTU-13	Yes	-	74.00%	100.00%
BotFP-MLP (Blaise, Bouet, Conan, & Secci, 2020)	CTU-13	Yes	-	85.00%	85.00%
BotFP-SVM (Blaise, Bouet, Conan, & Secci, 2020)	CTU-13	Yes	-	93.00%	93.00%
BotFP-Clus ($\epsilon = 0.1b$) (Blaise, Bouet, Conan, & Secci, 2020)	CTU-13	Yes	-	100.00%	85.00%
Convolutional Neural Network (Van Huong & Hung, 2019)	Self-generated	Yes	98.9%	-	-

Table 39. Performance comparison for IoT malware traffic.

Table 37 shows that proposed method outperforms (Choi, Kim, Lee, & Kim, 2019), (Meftah, Rachidi, & Assem, 2019) and (Jing & Chen, 2019) in terms of accuracy, and provides comparable performance to AODE (Nawir, Amir, Yaakob, & Lynn, 2019). However, AODE works with labeled data, while proposed method doesn't require prior examples of attack traffic.

Table 38 shows that proposed method exhibits lower accuracy than the other techniques, but it provides higher recall ratios for the fraud class than decision trees (Shahed, Ibrahim, & Akter, 2019) or SVM (Shahed, Ibrahim, & Akter, 2019). It also delivers lower precision and recall than Artificial Neural Network (Shahed, Ibrahim, & Akter, 2019), but it requires no prior knowledge of fraudulent transactions. Lastly, it delivers precision and recall ratios comparable to those obtained with the undercomplete autoencoder (Misra, Thakur, Ghosh, & Saha, 2020).

Table 39 shows that proposed method outperforms BotFP-MLP (Blaise, Bouet, Conan, & Secci, 2020) and BotFP-SVM (Blaise, Bouet, Conan, & Secci, 2020) techniques in terms of precision and recall, while providing comparable (but more stable) performance to both configurations of BotFP-Clus. It performs slightly worse than DEMISe (Parker, Yoo, Asyhari, Chermak, Jhi, & Taha, 2019) and Convolutional Neural Network (Van Huong & Hung, 2019), but it requires no knowledge of malicious traffic for training.

4.5. Supervised fraud detection optimization

Obtained results are summarized in Tables 40 to 49 (Carrasco & Sicilia, 2020).

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.92	0.43	0.89	0.59	0.85	0.71	0.81	0.78	0.78	0.84
Fraud	0.44	1.00	0.57	0.95	0.63	0.90	0.69	0.84	0.73	0.77	0.77	0.70
Average / Total	0.19	0.44	0.77	0.66	0.78	0.73	0.78	0.76	0.78	0.78	0.78	0.78

Table 40. MLP2BE256H82 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.92	0.43	0.89	0.58	0.86	0.68	0.83	0.74	0.80	0.81
Fraud	0.44	1.00	0.56	0.95	0.63	0.91	0.67	0.85	0.71	0.81	0.75	0.74
Average / Total	0.19	0.44	0.76	0.66	0.77	0.72	0.78	0.76	0.78	0.77	0.78	0.78

Table 41. MLP2BE128H164 performance.

Threshold	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**

Class	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Legitimate	0.00	0.00	0.90	0.57	0.87	0.65	0.82	0.81	0.74	0.90	0.74	0.81
Fraud	0.44	1.00	**0.63**	**0.92**	**0.67**	**0.88**	0.76	0.78	0.83	0.61	0.84	0.69
Average / Total	0.20	0.44	**0.78**	**0.72**	**0.78**	**0.75**	0.79	0.79	0.78	0.77	0.78	0.76

Table 42. MLP2OH128H918 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.77	0.81	0.76	0.84	0.75	0.86	0.74	0.87	0.73	0.88
Fraud	0.44	1.00	0.74	0.70	0.76	0.66	0.78	0.63	0.79	0.61	0.80	0.59
Average / Total	0.19	0.44	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.76	0.75

Table 43. MLP3OH256H512 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.83	0.66	0.82	0.70	0.80	0.74	0.79	0.76	0.78	0.78
Fraud	0.44	1.00	0.66	0.83	0.68	0.80	0.70	0.77	0.71	0.75	0.72	0.72
Average / Total	0.19	0.44	0.76	0.73	0.76	0.74	0.76	0.75	0.76	0.76	0.76	0.76

Table 44. MLP3BE256H512 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.84	0.69	0.82	0.73	0.81	0.77	0.80	0.80	0.79	0.81

Fraud	0.44	1.00	0.68	0.83	0.70	0.80	0.73	0.76	0.74	0.74	0.75	0.73
Average / Total	0.19	0.44	0.77	0.75	0.77	0.76	0.77	0.77	0.77	0.77	0.77	0.77

Table 45. MLP3OH256H918 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.00	0.00	0.85	0.09	0.78	0.35	0.73	0.60	0.67	0.78
Fraud	0.44	1.00	0.44	1.00	0.46	0.98	0.51	0.88	0.58	0.71	0.65	0.51
Average / Total	0.19	0.44	0.19	0.44	0.68	0.48	0.66	0.58	0.66	0.65	0.66	0.66

Table 46. CNN2OH100LR10-3 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.00	0.00	0.95	0.27	0.91	0.47	0.86	0.65	0.82	0.76	0.78	0.84
Fraud	0.44	1.00	0.51	0.98	0.58	0.94	0.66	0.87	0.72	0.78	0.77	0.69
Average / Total	0.19	0.44	0.76	0.58	0.77	0.68	0.77	0.74	0.77	0.77	0.78	0.78

Table 47. CNN2OH100LR10-1 performance.

Threshold	**0.0**		**0.1**		**0.2**		**0.3**		**0.4**		**0.5**	
Class	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Legitimate	0.21	0.65	0.20	0.52	0.21	0.42	0.23	0.34	0.23	0.24	0.25	0.13
Fraud	0.82	0.39	0.80	0.49	0.81	0.61	0.81	0.71	0.81	0.80	0.81	0.91
Average / Total	0.69	0.44	0.68	0.50	0.69	0.57	0.69	0.64	0.69	0.69	0.70	0.75

Table 48. DAE4BE256 performance.

Threshold	0.0		0.1		0.2		0.3		0.4		0.5	
Class	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Legitimate	0.20	0.62	0.19	0.49	0.17	0.32	0.17	0.25	0.17	0.17	0.18	0.09
Fraud	0.81	0.39	0.79	0.49	0.78	0.59	0.79	0.69	0.79	0.80	0.80	0.90
Average / Total	0.69	0.44	0.67	0.49	0.66	0.54	0.66	0.60	0.67	0.67	0.68	0.74

Table 49. DAE4OH256 performance.

The supervised learning approaches (MLP, CNN) obtained higher performance than the Deep Autoencoder.

The AUC was 0.52 for the Deep Autoencoder when using binary encoding, and 0.48 for OHE. This performance level might be explained by the fact that alerts contain biased normal activity (transactions tagged as potential fraud by the fraud detection system (Carrasco & Sicilia, 2020).

MLP architectures with less hidden layers exhibited higher AUC values. OHE exhibited higher AUC performance than binary encoding, while batch size, epochs and learning rate didn't impacted obtained performance (Carrasco & Sicilia, 2020).

MLP2OH128H918 achieved top AUC (0.87). With a value of 0.2 set for the threshold, tradeoff between average precision (0.78) and recall (0.75) was optimal. Fraud class obtained 0.67 precision and 0.88 recall (Carrasco & Sicilia, 2020).

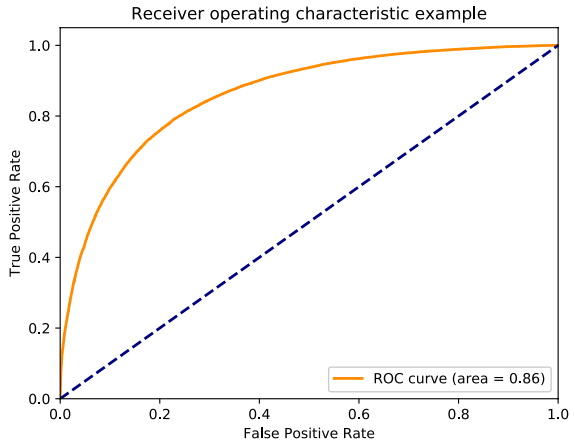

Figure 16. MLP2BE256H82 ROC.

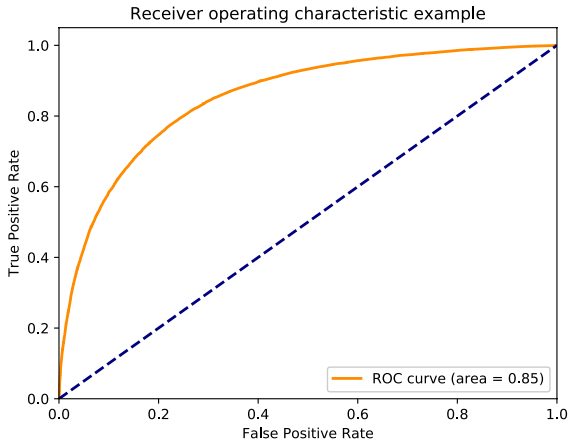

Figure 17. MLP2BE128H164 ROC.

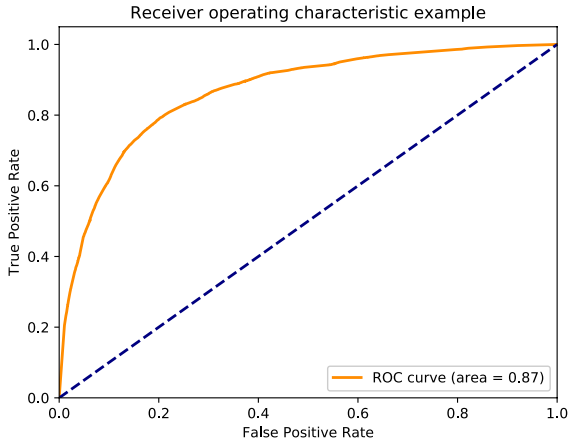

Figure 18. MLP2OH128H918 ROC.

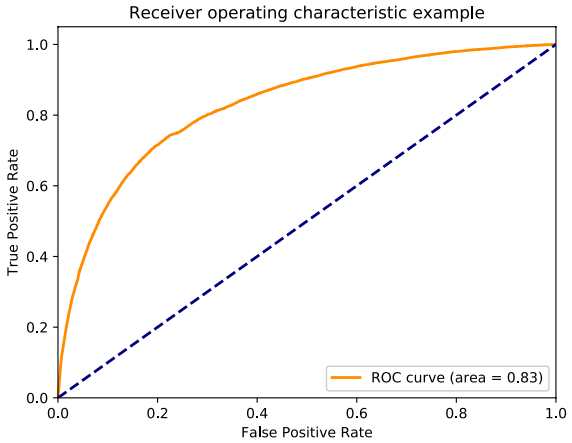

Figure 19. MLP3OH256H512 ROC.

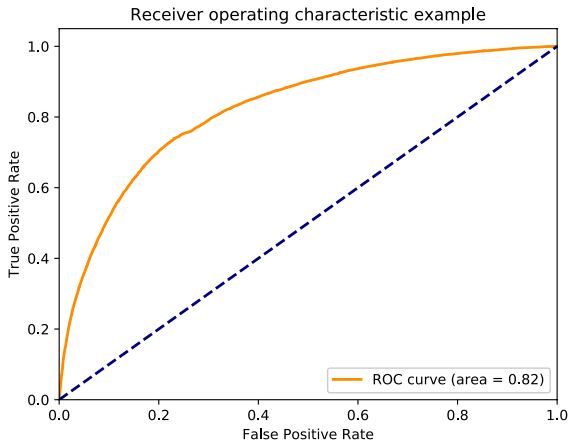

Figure 20. MLP3BE256H512 ROC.

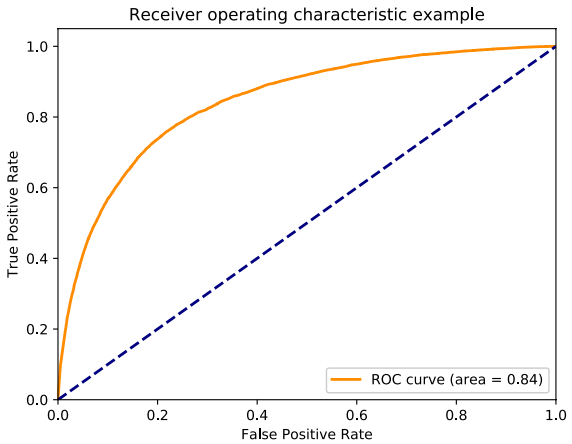

Figure 21. MLP3OH256H918 ROC.

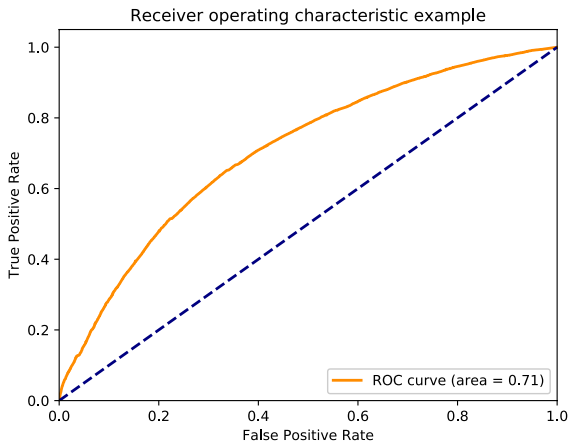

Figure 22. CNN2OH100LR10-3 ROC.

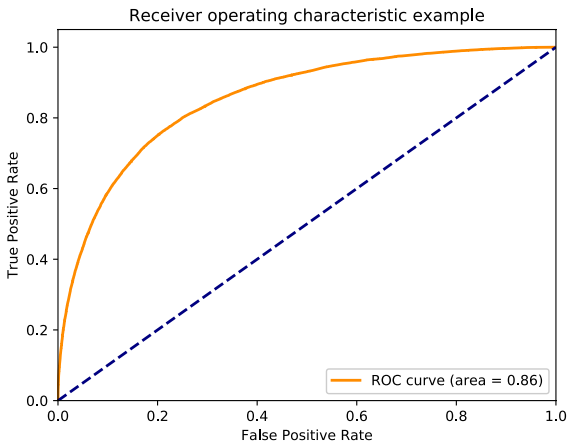

Figure 23. CNN2OH100LR10-1 ROC.

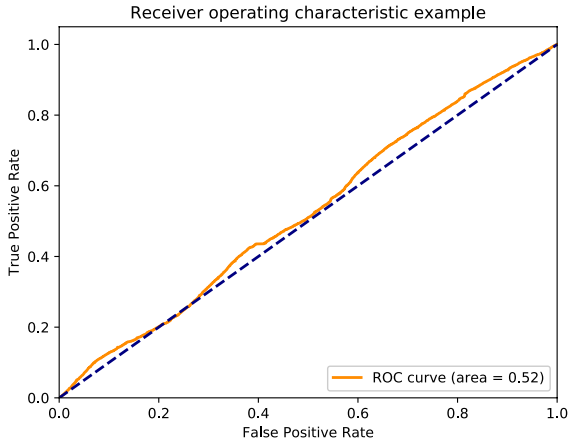

Figure 24. DAE4BE256 ROC.

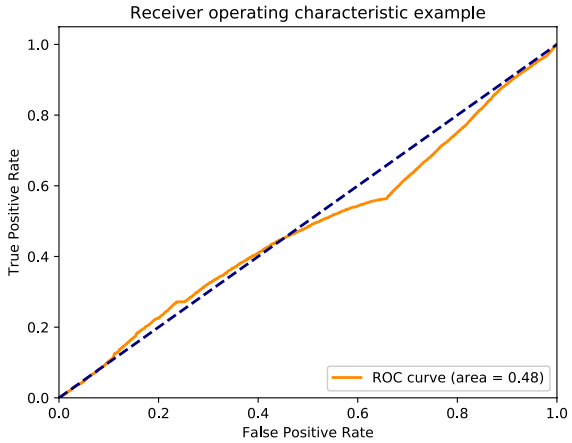

Figure 25. DAE4OH256 ROC.

Table 50 and Table 51 shows the confusion matrixes for threshold values 0.1 and 0.2:

Predicted class	**Legitimate**	**Fraud**	**Total**
Real class			
Legitimate	28,120	21,504	49,624
Fraud	3,249	36,343	39,592
Total	31,369	57,487	89,216

Table 50. Confusion matrix (threshold = 0.1)

Predicted class	**Legitimate**	**Fraud**	**Total**
Real class			
Legitimate	32,367	17,257	49,624
Fraud	4,849	34,743	39,592
Total	37,216	52,000	89,216

Table 51. Confusion matrix (threshold = 0.2)

The alert reduction rate was 35.16%:

$$\frac{31{,}369}{89{,}216} = 0.3516\ (35.16\%)$$

to detect 91.79% of fraud (8.21% misclassification rate):

$$\frac{36{,}343}{39{,}592} = 0.9179\ (91.79\%)$$

Alert reduction rate could grow to 41.47% with a slightly higher misclassification rate:

$$\frac{37{,}216}{89{,}216} = 0.4147\ (41.47\%)$$

to detect 87.75% of fraud (12.25% misclassification rate):

$$\frac{34{,}743}{39{,}592} = 0.8775\ (87.75\%)$$

5. Discussion

5.1. Introduction

In this section, the conclusions extracted from obtained results are presented and discussed. Final conclusions, related to the motivation and objectives of this research work, are also included.

5.2. Unsupervised intrusion detection

Proposed approach obtained 99.20% precision, 82.07% recall, 91.02% accuracy, and 0.61% false positive rate. The obtained false positive rate outperforms other similar methods (Carrasco & Sicilia, 2018). Furthermore, these figures were obtained without training the algorithm with attack data.

Proposed approach only requires four basic network-related features, which significantly lowers storage requirements. Other methods (Bamakan, Wang, & Shi, 2017) require the 49 features of UNSW-NB15. It also delivers a model in which results can be explained. Modeled behavior considers both known behavior of the node, and the behavior observed for similar nodes (Carrasco & Sicilia, 2018), which allows to extend the method to perform Peer Group Analysis (Botros, Diep, & Izenson, 2004) (Diep, Botros, & Izenson, 2003).

5.3. Semisupervised intrusion and fraud detection

In this research, a semisupervised learning system for intrusion was designed and tested. It was based on a deep autoencoder, a popular neural network architecture for anomaly detection. The proposed feature engineering approach transformed both categorical and numerical features into binary vectors that could be processed as inputs by the autoencoder. On the other hand, three feature selection techniques were tested, showing that selecting features based on their importance led to higher performance, but modeling frequent connections didn't consistently improve results.

The optimal configuration of proposed semisupervised intrusion detection system achieved 95.84% precision, 99.99% recall, 97.83% accuracy, and 97.87% F-score, with a false positive rate of 4.33%. With such recall (detection rate) value, proposed approach outperforms other similar techniques. On the other hand, the proposed system delivers this performance under a semisupervised setting, which is more suitable for detecting unknown attacks than techniques based on learning known attack patterns.

5.4. Cross-domain malicious behavior detection

Optimal configurations of proposed LDA-based technique led to accuracies of 95.82%, 83.21% and 94.72% for the UNSW-NB15 (network attacks), Paysim1 (payments fraud), IoT-23 (IoT malware traffic) datasets, respectively. These accuracies are comparable to, or even outperform, several state-of-art methods in each of the targeted fields.

This work demonstrates that topic modeling, particularly LDA, can be successfully applied to malicious behavior detection in three different, unrelated datasets used in previous literature. It achieves this objective without relying on a particular feature set structure, by transforming categorical and numerical features into words. It redefines the traditional concept of entity, switching from IP address or user to connection (protocol, service) or transaction type. Lastly, it applies a tailored scoring mechanism for anomaly detection, leveraging the data structures created by the LDA algorithm.

5.5. Supervised fraud detection optimization

Alert reduction rates was the motivation behind the assessment of several deep neural networks used in the fraud detection field. This reduction rate was enabled by the ability of each architecture to automate detection of false alerts (Carrasco & Sicilia, 2020).

The set of alerts used as input was processed to determine which of them were false positives.

A reduction rate of 35.16% was obtained with the optimal architecture. This configuration caught 91.79% of fraud (obtaining a 8.21% misclassification rate). The reduction rate was 41.47% if the percentage of caught fraud cases dropped to 87.75% (obtaining 12.25% misclassification rate) (Carrasco & Sicilia, 2020). This reduction rate implies less cost of human specialists, whose workload would be significantly lowered by proposed approach.

As a result, deep neural networks are a promising development path to obtain cost efficiencies in fraud detection (Carrasco & Sicilia, 2020).

6. Conclusions

The conclusions of this research are strongly connected with the original objectives of the work. For this reason, they are included in separate sections, one for each objective.

6.1. Reduce the dependence on well-known attack patterns through AI

In this research work, a novel unsupervised learning technique based on skip-gram models was designed, developed and tested against a public dataset with popular intrusion patterns. A high accuracy and a low false positive rate were achieved by modeling legitimate activity, without knowledge of these attack patterns.

Therefore, capturing frequent network behavior is an effective method to discover known and unknown attacks, as the related activity is significantly different from legitimate interactions. Moreover, relying on well-known attack patterns can provide a false sense of security, since unknown patterns will let sophisticated attacks go unnoticed.

6.2. Cross-domain applications and use cases of AI

The current state of artificial intelligence is far from AGI (Artificial General Intelligence), in which a machine can use general problem-solving methods and doesn't require to be trained on specific use cases. However, designing AI techniques that can solve a set of related problems is feasible, as demonstrated by this research.

Particularly, it was proven that an AI method like topic modeling can be successfully applied to three related domains (network attacks, payments fraud, IoT malware traffic) that share some structure and principles. A high accuracy was achieved in the three scenarios, even though the malicious activity significantly differs from one domain to the other. However, as the topic modeling was performed against legitimate activity, these differences didn't impact results.

This exercise also proves that, while choosing the right algorithm is important to solve a problem, framing the problem properly can be even more important. This is especially true when the chosen algorithm shouldn't solve a standalone problem but a set of related problems.

6.3. Augment unsupervised intrusion detection with supervised learning

A thorough discussion exists in the research community as to whether supervised or unsupervised techniques perform better when it comes to intrusion detection. There have been multiple proposals that leverage ensemble or hybrid models, in which both are integrated to maximize performance.

However, no extensive literature exists on how a supervised learning method can augment an unsupervised setting in the intrusion detection use case. In this research work, this augmentation was achieved by prioritizing or selecting features that could discriminate best whether a given activity was legitimate or malicious. Obtained results showed that this technique can outperform other similar techniques.

Therefore, augmentation of unsupervised learning with supervised-related insights is an effective method to increase intrusion detection accuracy of an algorithm trained with legitimate activity only, while not limiting these detection capabilities with well-known attack patterns.

6.4. Reduce false positive ratio of fraud detection systems

While automated detection is key in any malicious behavior detection use case, manual review by human experts is still required. Therefore, making this process more efficient helps to further improve the overall detection process.

In this research work, it was proven that part of this manual review can also be automated without significantly impacting accuracy. In other words, human judgement can be captured by an algorithm (in this case, a neural network) to quickly discriminate legitimate activity.

Therefore, in the malicious behavior detection use case, and particularly in fraud detection, the focus must go beyond the initial or real-time automation and expand to optimizing manual work performed by experts as well.

6.5. Research directions

Part of this research work was published as journal papers (Carrasco & Sicilia, 2018) (Carrasco & Sicilia, 2020). This opens up the opportunity to extend the research in a number of directions.

First, cross-domain application of AI methods can be further explored in the intrusion detection field. As businesses and supporting technology becomes more complex and heterogeneous, the need to develop methods that can be reused across fields becomes more important. Having the ability to apply an AI technique to multiple set of problems can lead to significant efficiencies and pave the road to more general reasoning systems that could detect malicious behavior regardless of what the exact environment is.

Also, augmentation of a given technique with another that takes a different approach might have more applications to intrusion detection than the one shown in this research. As the intrusion detection literature is extensive and well-grounded, it might be the time to rethink how existing knowledge on the field can be combined in such a way that a higher-order knowledge can be extracted.

Lastly, while using fully automated and optimized methods to solve problems is desirable, exploring ways to mimic human decision-making processes can also help to further optimize a process. In this research work the focus was on reducing the false positive ratio, but intrusion detection requires multiple other activities where human experts play a relevant role. These could be candidates to achieve similar optimizations.

7. References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

Ahmad, I., Abdullah, A. B., & Alghamdi, A. S. (2010, June). Evaluating neural network intrusion detection approaches using Analytic Hierarchy Process. In 2010 International Symposium on Information Technology (Vol. 2, pp. 885-890). IEEE.

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. Neurocomputing, 262, 134-147.

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.

Ahmed, T., Coates, M., & Lakhina, A. (2007, May). Multivariate online anomaly detection using kernel recursive least squares. In IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications (pp. 625-633). IEEE.

Aissa, N. B., & Guerroumi, M. (2015, June). A genetic clustering technique for Anomaly-based Intrusion Detection Systems. In 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) (pp. 1-6). IEEE.

Akhi, A. B., Kanon, E. J., Kabir, A., & Banu, A. (2019). Network intrusion classification employing machine learning: A survey (Doctoral dissertation).

Al-Subaie, M., & Zulkernine, M. (2007, June). The power of temporal pattern processing in anomaly intrusion detection. In 2007 IEEE International Conference on Communications (pp. 1391-1398). IEEE.

Alam, M. S., Fernando, B. R., Jaoudi, Y., Yakopcic, C., Hasan, R., Taha, T. M., & Subramanyam, G. (2019, July). Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection. In Proceedings of the International Conference on Neuromorphic Systems (pp. 1-8).

Aldwairi, T., Perera, D., & Novotny, M. A. (2018). An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection. Computer Networks, 144, 111-119.

Aleskerov, E., Freisleben, B., & Rao, B. (1997, March). Cardwatch: A neural network based database mining system for credit card fraud detection. In Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr) (pp. 220-226). IEEE.

Alshammari, R., Sonamthiang, S., Teimouri, M., & Riordan, D. (2007, May). Using neuro-fuzzy approach to reduce false positive alerts. In Fifth Annual Conference on Communication Networks and Services Research (CNSR'07) (pp. 345-349). IEEE.

Alston, A. (2017). Extending the Metasploit Framework to Implement an Evasive Attack Infrastructure. arXiv preprint arXiv:1705.04853.

Amer, M., Goldstein, M., & Abdennadher, S. (2013, August). Enhancing one-class support vector machines for unsupervised anomaly detection. In Proceedings of the ACM SIGKDD workshop on outlier detection and description (pp. 8-15).

Amini, M., Jalili, R., & Shahriari, H. R. (2006). RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. computers & security, 25(6), 459-468.

Ashraf, N., Ahmad, W., & Ashraf, R. (2018). A comparative study of data mining algorithms for high detection rate in intrusion detection system. Annals of Emerging Technologies in Computing (AETiC), Print ISSN, 2516-0281.

Aswani, K., Cronin, A., Liu, X., & Zhao, H. (2015, April). Topic modeling of SSH logs using latent dirichlet allocation for the application in cyber security. In 2015 Systems and Information Engineering Design Symposium (pp. 75-79). IEEE.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy (Vol. 99). Technical report.

Bakar, N. A., Belaton, B., & Samsudin, A. (2005, November). False positives reduction via intrusion alert quality framework. In 2005 13th IEEE International Conference on Networks Jointly held with the 2005 IEEE 7th Malaysia International Conf on Communic (Vol. 1, pp. 6-pp). IEEE.

Balthrop, J., Forrest, S., & Glickman, M. R. (2002, May). Revisiting lisys: Parameters and normal behavior. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 2, pp. 1045-1050). IEEE.

Bamakan, S. M. H., Wang, H., & Shi, Y. (2017). Ramp loss K-Support Vector Classification-Regression; a robust and sparse multi-class approach to the intrusion detection problem. Knowledge-Based Systems, 126, 113-126.

Bansal, P., Gualandris, J., & Kim, N. (2020). Theorizing supply chains with qualitative Big Data and Topic Modeling. Journal of Supply Chain Management, 56(2), 7-18.

Barot, K., Zhang, J., & Son, S. W. (2016). Using natural language processing models for understanding network anomalies. In Proceedings of the IEEE High Performance Extreme Computing Conference.

Bashah, N., Shanmugam, I. B., & Ahmed, A. M. (2005). Hybrid intelligent intrusion detection system. World Academy of Science, Engineering and Technology, 11, 23-26.

Bass, T. (2000). Intrusion detection systems and multisensor data fusion. Communications of the ACM, 43(4), 99-105.

Bertero, C., Roy, M., Sauvanaud, C., & Trédan, G. (2017, October). Experience report: Log mining using natural language processing and application to anomaly detection. In 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE) (pp. 351-360). IEEE.

Blaise, A., Bouet, M., Conan, V., & Secci, S. (2020). Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection. IEEE Transactions on Network and Service Management, 17(3), 1701-1714.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. the Journal of machine Learning research, 3, 993-1022.

Bolzoni, D., Etalle, S., & Hartel, P. (2006, April). Poseidon: a 2-tier anomaly-based network intrusion detection system. In Fourth IEEE International Workshop on Information Assurance (IWIA'06) (pp. 10-pp). IEEE.

Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., & Benini, L. (2019, July). Anomaly detection using autoencoders in high performance computing systems. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 9428-9433).

Botros, S. M., Diep, T. A., & Izenson, M. D. (2004). U.S. Patent No. 6,769,066. Washington, DC: U.S. Patent and Trademark Office.

Bridges, S. M., & Vaughn, R. B. (2000, October). Fuzzy data mining and genetic algorithms applied to intrusion detection. In Proceedings of 12th Annual Canadian Information Technology Security Symposium (pp. 109-122).

Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications surveys & tutorials, 18(2), 1153-1176.

Burbeck, K., & Nadjm-Tehrani, S. (2004, December). Adwice–anomaly detection with real-time incremental clustering. In International Conference on Information Security and Cryptology (pp. 407-424). Springer, Berlin, Heidelberg.

Cannady, J. (1998, October). Artificial neural networks for misuse detection. In Proceedings of the 1998 National Information Systems Security Conference (NISSC'98) (pp. 443-456).

Cannady, J. (2000, October). Next generation intrusion detection: Autonomous reinforcement learning of network attacks. In Proceedings of the 23rd national information systems security conference (pp. 1-12).

Cao, S., Yang, X., Chen, C., Zhou, J., Li, X., & Qi, Y. (2019). Titant: Online real-time transaction fraud detection in ant financial. arXiv preprint arXiv:1906.07407.

Carcillo, F., Le Borgne, Y. A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2019). Combining unsupervised and supervised learning in credit card fraud detection. Information Sciences.

Carlin, A., Hammoudeh, M., & Aldabbas, O. (2015). Intrusion detection and countermeasure of virtual cloud systems-state of the art and current challenges. International Journal of Advanced Computer Science and Applications, 6(6).

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer vision, graphics, and image processing, 37(1), 54-115.

Carpenter, G. A., & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. Applied optics, 26(23), 4919-4930.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural networks, 4(6), 759-771.

Carrasco, R. S. M., & Sicilia, M. A. (2018). Unsupervised intrusion detection through skip-gram models of network behavior. Computers & Security, 78, 187-197.

Carrasco, R. S. M., & Sicilia-Urbán, M. Á. (2020). Evaluation of Deep Neural Networks for Reduction of Credit Card Fraud Alerts. IEEE Access, 8, 186421-186432.

Casas, P., Mazel, J., & Owezarski, P. (2012). Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. Computer Communications, 35(7), 772-783.

Chakraborty, K., Bhattacharyya, S., Bag, R., & Hassanien, A. E. (2018, February). Comparative sentiment analysis on a set of movie reviews using deep learning approach. In International Conference on Advanced Machine Learning Technologies and Applications (pp. 311-318). Springer, Cham.

Chan, P. K., Mahoney, M. V., & Arshad, M. H. (2003). A machine learning approach to anomaly detection.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

Chapaneri, R., & Shah, S. (2019). A comprehensive survey of machine learning-based network intrusion detection. Smart Intelligent Computing and Applications, 345-356.

Chauhan, M., Pratap, A., & Dixit, A. (2015, October). Designing a technique for detecting intrusion based on modified Adaptive Resonance Theory Network. In 2015 International Conference on Green Computing and Internet of Things (ICGCIoT) (pp. 448-451). IEEE.

Chebrolu, S., Abraham, A., & Thomas, J. P. (2005). Feature deduction and ensemble design of intrusion detection systems. Computers & security, 24(4), 295-307.

Chen, J., Sathe, S., Aggarwal, C., & Turaga, D. (2017, June). Outlier detection with autoencoder ensembles. In Proceedings of the 2017 SIAM international conference on data mining (pp. 90-98). Society for Industrial and Applied Mathematics.

Chen, K. (1988). An Inductive Engine for the Acquisition of Temporal Knowledge (Doctoral dissertation, University of Illinois at Urbana-Champaign).

Chen, K., Lu, S. C., & Teng, H. S. (1990, May). Adaptive real-time anomaly detection using inductively generated sequential patterns,". In Fifth Intrusion Detection Workshop, SRI International, Menlo Park, CA.

Chen, L., Zhang, Z., Liu, Q., Yang, L., Meng, Y., & Wang, P. (2019, May). A method for online transaction fraud detection based on individual behavior. In Proceedings of the ACM Turing Celebration Conference-China (pp. 1-8).

Chen, Y., Dong, F., Chen, H., & Xu, L. (2016, August). Can Cross-Listing Mitigate the Impact of an Information Security Breach Announcement on a Firm's Values?. In IOP Conference Series: Materials Science and Engineering (Vol. 142, No. 1, p. 012133). IOP Publishing.

Chimphlee, W., Abdullah, A. H., Sap, M. N. M., Srinoy, S., & Chimphlee, S. (2006, November). Anomaly-based intrusion detection using fuzzy rough clustering. In 2006 International Conference on Hybrid Information Technology (Vol. 1, pp. 329-334). IEEE.

Cho, S., & Cha, S. (2004). SAD: web session anomaly detection based on parameter estimation. Computers & Security, 23(4), 312-319.

Choi, H., Kim, M., Lee, G., & Kim, W. (2019). Unsupervised learning approach for network intrusion detection system using autoencoders. The Journal of Supercomputing, 75(9), 5597-5621.

Choi, M., Shin, S., Choi, J., Langevin, S., Bethune, C., Horne, P., ... & Choo, J. (2018, April). Topicontiles: Tile-based spatio-temporal event analytics via exclusive topic modeling on social

media. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-11).

Choi, T. M., Chan, H. K., & Yue, X. (2016). Recent development in big data analytics for business operations and risk management. IEEE transactions on cybernetics, 47(1), 81-92.

Choksi, K., Shah, B., & Kale, O. (2014). Intrusion detection system using self organizing map: a survey. International Journal of Engineering Research and Applications, 4(12), 11-16.

Ciritsis, A., Rossi, C., Eberhard, M., Marcon, M., Becker, A. S., & Boss, A. (2019). Automatic classification of ultrasound breast lesions using a deep convolutional neural network mimicking human decision-making. European radiology, 29(10), 5458-5468.

Cohen, W. W. (1995). Fast effective rule induction. In Machine learning proceedings 1995 (pp. 115-123). Morgan Kaufmann.

Cook, D. J., & Holder, L. B. (2000). Graph-based data mining. IEEE Intelligent Systems and Their Applications, 15(2), 32-41.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Cramer, C., & Carin, L. (2011, May). Bayesian topic models for describing computer network behaviors. In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1888-1891). IEEE.

Cybersecurity Challenges in the Middle East. (n.d.). GCSP. Retrieved January 31, 2021, from https://css.ethz.ch/content/dam/ethz/special-interest/gess/cis/center-for-securities-studies/resources/docs/GCSP-Cybersecurity%20Challenges%20in%20the%20Middle%20East.pdf

D'haeseleer, P. (1996). An Immunological Approach to Change Detection: Theoretical Results. In Proceedings of the 9th IEEE Computer Security Foundations Workshop (pp. 66-71). IEEE Computer Society Press.

da Costa, K. A., Papa, J. P., Lisboa, C. O., Munoz, R., & de Albuquerque, V. H. C. (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. Computer Networks, 151, 147-157.

Dahiya, P., & Srivastava, D. K. (2018). Network intrusion detection in big dataset using spark. Procedia computer science, 132, 253-262.

Darwish, S. M. (2020). An intelligent credit card fraud detection approach based on semantic fusion of two classifiers. Soft Computing, 24(2), 1243-1253.

Dasgupta, D. (1999, October). Immunity-based intrusion detection system: A general framework. In Proc. of the 22nd NISSC (Vol. 1, pp. 147-160).

Davison, B. D., & Hirsh, H. (1998, July). Predicting sequences of user actions. In Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis (pp. 5-12).

de Paula, F. S., de Castro, L. N., & de Geus, P. L. (2004, June). An intrusion detection system using ideas from the immune system. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753) (Vol. 1, pp. 1059-1066). IEEE.

Debar, H., Becker, M., & Siboni, D. (1992, May). A neural network component for an intrusion detection system. In Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy (pp. 240-240). IEEE Computer Society.

Debar, H., Dacier, M., & Wespi, A. (2000, July). A revised taxonomy for intrusion-detection systems. In Annales des télécommunications (Vol. 55, No. 7, pp. 361-378). Springer-Verlag.

Deokar, B., & Hazarnis, A. (2012). Intrusion detection system using log files and reinforcement learning. International Journal of Computer Applications, 45(19), 28-35.

Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert systems with Applications, 29(4), 713-722.

Dhillon, I. S., Mallela, S., & Kumar, R. (2003). A divisive information theoretic feature clustering algorithm for text classification. The Journal of machine learning research, 3, 1265-1287.

Diep, T. A., Botros, S. M., & Izenson, M. D. (2003). U.S. Patent No. 6,671,811. Washington, DC: U.S. Patent and Trademark Office.

Dornadula, V. N., & Geetha, S. (2019). Credit card fraud detection using machine learning algorithms. Procedia Computer Science, 165, 631-641.

dos Santos, L. B., Duran, M. S., Hartmann, N. S., Candido, A., Paetzold, G. H., & Aluisio, S. M. (2017, August). A lightweight regression method to infer psycholinguistic properties for Brazilian Portuguese. In International conference on text, speech, and dialogue (pp. 281-289). Springer, Cham.

Durgin, N. A., & Zhang, P. (2005). Profile-based adaptive anomaly detection for network security. Sandia National Laboratories Technical Report, SAND2005-7293.

Elhamahmy, M. E., Elmahdy, H. N., & Saroit, I. A. (2010). A new approach for evaluating intrusion detection system. CiiT International Journal of Artificial Intelligent Systems and Machine Learning, 2(11), 290-298.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognition, 58, 121-134.

Eshghi, A., & Kargari, M. (2019, January). Introducing a method for combining supervised and semi-supervised methods in fraud detection. In 2019 15th Iran International Industrial Engineering Conference (IIIEC) (pp. 23-30). IEEE.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In Applications of data mining in computer security (pp. 77-101). Springer, Boston, MA.

Estevez-Tapiador, J. M., Garcia-Teodoro, P., & Diaz-Verdejo, J. E. (2004). Anomaly detection methods in wired networks: a survey and taxonomy. Computer Communications, 27(16), 1569-1584.

Fan, W., Miller, M., Stolfo, S., Lee, W., & Chan, P. (2004). Using artificial anomalies to detect unknown and known network intrusions. Knowledge and Information Systems, 6(5), 507-527.

Fan, Y., Wen, G., Li, D., Qiu, S., Levine, M. D., & Xiao, F. (2020). Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder. Computer Vision and Image Understanding, 195, 102920.

Feng, Y., Wu, Z. F., Wu, K. G., Xiong, Z. Y., & Zhou, Y. (2005, August). An unsupervised anomaly intrusion detection algorithm based on swarm intelligence. In 2005 International Conference on Machine Learning and Cybernetics (Vol. 7, pp. 3965-3969). IEEE.

Feng, Y., Zhong, J., Xiong, Z. Y., Ye, C. X., & Wu, K. G. (2007, June). Network anomaly detection based on DSOM and ACO clustering. In International Symposium on Neural Networks (pp. 947-955). Springer, Berlin, Heidelberg.

Feng, Y., Zhong, J., Ye, C. X., & Wu, Z. F. (2006, October). Clustering based on self-organizing ant colony networks with application to intrusion detection. In Sixth International Conference on Intelligent Systems Design and Applications (Vol. 2, pp. 1077-1080). IEEE.

Fernandes, G., Rodrigues, J. J., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2019). A comprehensive survey on network anomaly detection. Telecommunication Systems, 70(3), 447-489.

Fernández, G. C., & Xu, S. (2019, November). A case study on using deep learning for network intrusion detection. In MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM) (pp. 1-6). IEEE.

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Information Sciences, 479, 448-455.

Firdausi, I., Erwin, A., & Nugroho, A. S. (2010, December). Analysis of machine learning techniques used in behavior-based malware detection. In 2010 second international conference on advances in computing, control, and telecommunication technologies (pp. 201-203). IEEE.

Fogla, P., Sharif, M. I., Perdisci, R., Kolesnikov, O. M., & Lee, W. (2006, August). Polymorphic Blending Attacks. In USENIX security symposium (pp. 241-256).

Forrest, S., Perelson, A. S., Allen, L., & Cherukuri, R. (1994, May). Self-nonself discrimination in a computer. In Proceedings of 1994 IEEE computer society symposium on research in security and privacy (pp. 202-212). Ieee.

Gao, D., Reiter, M. K., & Song, D. (2004, October). Gray-box extraction of execution graphs for anomaly detection. In Proceedings of the 11th ACM conference on Computer and communications security (pp. 318-329).

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

Garvey, T. D., & Lunt, T. F. (1991, October). Model based intrusion detection. In Proceedings of the 14th national computer security conference (Vol. 10, pp. 372-385).

Gates, C., & Taylor, C. (2006, September). Challenging the anomaly detection paradigm: a provocative discussion. In Proceedings of the 2006 workshop on New security paradigms (pp. 21-29).

Gensim: topic modelling for humans. (n.d.). Gensim. Retrieved January 31, 2021, from https://radimrehurek.com/gensim/

Ghosh, A. K., Wanken, J., & Charron, F. (1998, December). Detecting anomalous and unknown intrusions against programs. In Proceedings 14th annual computer security applications conference (Cat. No. 98Ex217) (pp. 259-267). IEEE.

Girardin, L. (1999, April). An Eye on Network Intruder-Administrator Shootouts. In Workshop on Intrusion Detection and Network Monitoring (pp. 19-28).

Girdhar, Y., Cho, W., Campbell, M., Pineda, J., Clarke, E., & Singh, H. (2016, May). Anomaly detection in unstructured environments using Bayesian nonparametric scene modeling. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2651-2656). IEEE.

Gogoi, P., Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2012, August). Packet and flow based network intrusion dataset. In International Conference on Contemporary Computing (pp. 322-334). Springer, Berlin, Heidelberg.

Gómez, J., González, F., & Dasgupta, D. (2003, May). An immuno-fuzzy approach to anomaly detection. In The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03. (Vol. 2, pp. 1219-1224). IEEE.

González, F. A., & Dasgupta, D. (2003). Anomaly detection using real-valued negative selection. Genetic Programming and Evolvable Machines, 4(4), 383-403.

Gonzalez, F., & Dasgupta, D. (2002, September). Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In First International Conference on Artificial Immune Systems (pp. 203-211).

Gonzalez, F., Dasgupta, D., & Niño, L. F. (2003, September). A randomized real-valued negative selection algorithm. In International Conference on Artificial Immune Systems (pp. 261-272). Springer, Berlin, Heidelberg.

Greensmith, J., Twycross, J., & Aickelin, U. (2006, July). Dendritic cells for anomaly detection. In 2006 IEEE International Conference on Evolutionary Computation (pp. 664-671). IEEE.

Gupta, G. (2019). MACHINE LEARNING APPROACH FOR CREDIT CARD FRAUD DETECTION. Global Journal For Research Analysis (GJRA), 8(9). https://doi.org/10.36106/gjra

Gyanchandani, M., Rana, J. L., & Yadav, R. N. (2012). Taxonomy of anomaly based intrusion detection system: a review. International Journal of Scientific and Research Publications, 2(12), 1-13.

Han, S. J., & Cho, S. B. (2006). Evolutionary neural networks for anomaly detection based on the behavior of a program. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36(3), 559-570.

Harang, R., & Guarino, P. (2012, October). Clustering of Snort alerts to identify patterns and reduce analyst workload. In MILCOM 2012-2012 IEEE Military Communications Conference (pp. 1-6). IEEE.

Hashim, F., Munasinghe, K. S., & Jamalipour, A. (2010). Biologically inspired anomaly detection and security control frameworks for complex heterogeneous networks. IEEE Transactions on Network and Service Management, 7(4), 268-281.

He, Z., Xu, X., Huang, Z. J., & Deng, S. (2005). FP-outlier: Frequent pattern based outlier detection. Computer Science and Information Systems, 2(1), 103-118.

Hick, P., Aben, E., Claffy, K., & Polterock, J. (2007). the CAIDA DDoS attack 2007 dataset. 2012)[2015-07-10]. http://www. caida. org.

Hill, T., & Remus, W. (1994). Neural network models for intelligent support of managerial decision making. Decision Support Systems, 11(5), 449-459.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2018). A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. arXiv preprint arXiv:1806.03517.

Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017). Shallow and deep networks intrusion detection system: A taxonomy and survey. arXiv preprint arXiv:1701.02145.

Hofmann, T. (2013). Probabilistic latent semantic analysis. arXiv preprint arXiv:1301.6705.

Hofmeyr, S. A. (1999). An immunological model of distributed detection and its application to computer security (Doctoral dissertation, PhD thesis, University of New Mexico).

Hofmeyr, S. A., & Forrest, S. (1999, July). Immunity by design: An artificial immune system. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2 (pp. 1289-1296).

Hoglund, A. J., Hatonen, K., & Sorvari, A. S. (2000, July). A computer host-based user anomaly detection system using the self-organizing map. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium (Vol. 5, pp. 411-416). IEEE.

Hollis, G., Westbury, C., & Lefsrud, L. (2017). Extrapolating human judgments from skip-gram vector representations of word meaning. Quarterly Journal of Experimental Psychology, 70(8), 1603-1619.

Huang, J., Kalbarczyk, Z., & Nicol, D. M. (2014, June). Knowledge discovery from big data for intrusion detection using LDA. In 2014 IEEE International Congress on Big Data (pp. 760-761). IEEE.

Huang, L., Nguyen, X., Garofalakis, M., Jordan, M. I., Joseph, A., & Taft, N. (2006, December). In-network PCA and anomaly detection. In NIPS (Vol. 2006, pp. 617-624).

Huang, Z., & Zeng, D. D. (2006, October). A link prediction approach to anomalous email detection. In 2006 IEEE International Conference on Systems, Man and Cybernetics (Vol. 2, pp. 1131-1136). IEEE.

Idika, N., & Mathur, A. P. (2007). A survey of malware detection techniques. Purdue University, 48, 2007-2.

IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic. (n.d.). Stratosphere IPS. Retrieved January 31, 2021, from https://www.stratosphereips.org/datasets-iot23

Islam, S. A., Heil, B. J., Kearney, C. M., & Baker, E. J. (2017). Protein classification using modified n-gram and skip-gram models. bioRxiv, 170407.

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016, May). A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS) (pp. 21-26).

Jing, D., & Chen, H. B. (2019, October). SVM based network intrusion detection for the UNSW-NB15 dataset. In 2019 IEEE 13th International Conference on ASIC (ASICON) (pp. 1-4). IEEE.

Joshi, B., Joshi, B., & Rani, K. (2017). Mitigating data segregation and privacy issues in cloud computing. In Proceedings of International Conference on Communication and Networks (pp. 175-182). Springer, Singapore.

Joshi, S. S., & Phoha, V. V. (2005, March). Investigating hidden Markov models capabilities in anomaly detection. In Proceedings of the 43rd annual Southeast regional conference-Volume 1 (pp. 98-103).

Kang, H. (2019). Fraud Detection in Mobile Money Transactions Using Machine Learning.

Karim, A. M., Güzel, M. S., Tolun, M. R., Kaya, H., & Çelebi, F. V. (2018). A new generalized deep learning framework combining sparse autoencoder and Taguchi method for novel data classification and processing. Mathematical Problems in Engineering, 2018.

Karthick, R. R., Hattiwale, V. P., & Ravindran, B. (2012, January). Adaptive network intrusion detection system using a hybrid approach. In 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012) (pp. 1-7). IEEE.

Kasa, N., Dahbura, A., Ravoori, C., & Adams, S. (2019, April). Improving credit card fraud detection by profiling and clustering accounts. In 2019 Systems and Information Engineering Design Symposium (SIEDS) (pp. 1-6). IEEE.

Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2007). A hierarchical SOM-based intrusion detection system. Engineering applications of artificial intelligence, 20(4), 439-451.

Keegan, N., Ji, S. Y., Chaudhary, A., Concolato, C., Yu, B., & Jeong, D. H. (2016). A survey of cloud-based network intrusion detection analysis. Human-centric Computing and Information Sciences, 6(1), 1-16.

Kephart, J. O. (1994). A biologically inspired immune system for computers. In In proc. Of the fourth international workshop on synthesis and simulation of living systems, artificial life IV.

Khan, F. A., Gumaei, A., Derhab, A., & Hussain, A. (2019). A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access, 7, 30373-30385.

Khreich, W., Granger, E., Sabourin, R., & Miri, A. (2009, June). Combining hidden markov models for improved anomaly detection. In 2009 IEEE International Conference on Communications (pp. 1-6). IEEE.

Kim, A. Y., Ha, J. G., Choi, H., & Moon, H. (2018). Automated text analysis based on skip-gram model for food evaluation in predicting consumer acceptance. Computational intelligence and neuroscience, 2018.

Kim, E., Lee, J., Shin, H., Yang, H., Cho, S., Nam, S. K., ... & Kim, J. I. (2019). Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning. Expert Systems with Applications, 128, 214-224.

Kim, J., & Bentley, P. (1999, September). The artificial immune model for network intrusion detection. In 7th European congress on intelligent techniques and soft computing (EUFIT'99) (Vol. 158).

Kim, J., & Bentley, P. J. (2001, July). An evaluation of negative selection in an artificial immune system for network intrusion detection. In Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (pp. 1330-1337).

Kim, J., & Bentley, P. J. (2001, May). Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546) (Vol. 2, pp. 1244-1252). IEEE.

Kim, J., & Bentley, P. J. (2002, May). Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600) (Vol. 2, pp. 1015-1020). IEEE.

Kim, J., Bentley, P. J., Aickelin, U., Greensmith, J., Tedesco, G., & Twycross, J. (2007). Immune system approaches to intrusion detection–a review. Natural computing, 6(4), 413-466.

Kim, J., Greensmith, J., Twycross, J., & Aickelin, U. (2010). Malicious code execution detection and response immune system inspired by the danger theory. arXiv preprint arXiv:1003.4142.

Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016, February). Long short term memory recurrent neural network classifier for intrusion detection. In 2016 International Conference on Platform Technology and Service (PlatCon) (pp. 1-5). IEEE.

Kim, J., Park, M., Kim, H., Cho, S., & Kang, P. (2019). Insider threat detection based on user behavior modeling and anomaly detection algorithms. Applied Sciences, 9(19), 4018.

Kind, A., Stoecklin, M. P., & Dimitropoulos, X. (2009). Histogram-based traffic anomaly detection. IEEE Transactions on Network and Service Management, 6(2), 110-121.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kiran, B. R. (2017). Multi-scale streaming anomalies detection for time series. arXiv preprint arXiv:1706.06910.

Kloft, M., & Laskov, P. (2010, March). Online anomaly detection under adversarial impact. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (pp. 405-412). JMLR Workshop and Conference Proceedings.

Kloft, M., & Laskov, P. (2012). Security analysis of online centroid anomaly detection. The Journal of Machine Learning Research, 13(1), 3681-3724.

Kruegel, C., & Vigna, G. (2003, October). Anomaly detection of web-based attacks. In Proceedings of the 10th ACM conference on Computer and communications security (pp. 251-261).

Kruegel, C., Kirda, E., Mutz, D., Robertson, W., & Vigna, G. (2005, August). Automating mimicry attacks using static binary analysis. In USENIX Security Symposium (Vol. 14, pp. 11-11).

Kruegel, C., Mutz, D., Robertson, W., & Valeur, F. (2003, December). Bayesian event classification for intrusion detection. In 19th Annual Computer Security Applications Conference, 2003. Proceedings. (pp. 14-23). IEEE.

Krügel, C., Toth, T., & Kirda, E. (2002, March). Service specific anomaly detection for network intrusion detection. In Proceedings of the 2002 ACM symposium on Applied computing (pp. 201-208).

Kumar, A., Glisson, W., & Cho, H. (2020). Network attack detection using an unsupervised machine learning algorithm.

Kumar, M. S., Soundarya, V., Kavitha, S., Keerthika, E. S., & Aswini, E. (2019, February). Credit card fraud detection using random forest algorithm. In 2019 3rd International Conference on Computing and Communications Technologies (ICCCT) (pp. 149-153). IEEE.

Kumar, V. D., & Radhakrishnan, S. (2014, April). Intrusion detection in MANET using self organizing map (SOM). In 2014 International Conference on Recent Trends in Information Technology (pp. 1-8). IEEE.

Kumari, P., & Mishra, S. P. (2019). Analysis of credit card fraud detection using fusion classifiers. In Computational Intelligence in Data Mining (pp. 111-122). Springer, Singapore.

Kuypers, M. A., Maillart, T., & Paté-Cornell, E. (2016). An empirical analysis of cyber security incidents at a large organization. Department of Management Science and Engineering, Stanford University, School of Information, UC Berkeley, 30.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. Cluster Computing, 22(1), 949-961.

Lane, T., & Brodley, C. E. (1997, July). Sequence matching and learning in anomaly detection for computer security. In AAAI Workshop: AI Approaches to Fraud Detection and Risk Management (pp. 43-49).

Lane, T., & Brodley, C. E. (1997, October). An application of machine learning to anomaly detection. In Proceedings of the 20th National Information Systems Security Conference (Vol. 377, pp. 366-380). Baltimore, USA.

Lane, T., & Brodley, C. E. (1998, August). Approaches to Online Learning and Concept Drift for User Identification in Computer Security. In KDD (pp. 259-263).

Lane, T., & Brodley, C. E. (1999). Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information and System Security (TISSEC), 2(3), 295-331.

Lane, T., & Brodley, C. E. (2003). An empirical study of two approaches to sequence learning for anomaly detection. Machine learning, 51(1), 73-107.

Last, M., Shapira, B., Elovici, Y., Zaafrany, O., & Kandel, A. (2003, May). Content-based methodology for anomaly detection on the web. In International Atlantic Web Intelligence Conference (pp. 113-123). Springer, Berlin, Heidelberg.

Latif, S., Kulkarni, A., Molkire, R., & Nangare, P. (2020). Study of Credit Card Fraud Recognition using machine learning classification methods. CLIO An Annual Interdisciplinary Journal of History, 6(4), 87-91.

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. (2003, May). A comparative study of anomaly detection schemes in network intrusion detection. In Proceedings of the 2003 SIAM international conference on data mining (pp. 25-36). Society for Industrial and Applied Mathematics.

Lee, W. (1999). A date mining framework for constructing features and models for intrusion detection systems (Doctoral dissertation, Columbia University).

Lee, W., Stolfo, S. J., & Mok, K. W. (1999, May). A data mining framework for building intrusion detection models. In Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344) (pp. 120-132). IEEE.

Leung, K., & Leckie, C. (2005, January). Unsupervised anomaly detection in network intrusion detection using clusters. In Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38 (pp. 333-342).

Li, Y., Fang, B., Guo, L., & Chen, Y. (2007, March). Network anomaly detection based on TCM-KNN algorithm. In Proceedings of the 2nd ACM symposium on Information, computer and communications security (pp. 13-19).

Liang, Z., Zhang, G., Huang, J. X., & Hu, Q. V. (2014, November). Deep learning for healthcare decision making with EMRs. In 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 556-559). IEEE.

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

Linda, O., Manic, M., Vollmer, T., & Wright, J. (2011, April). Fuzzy logic based anomaly detection for embedded network security cyber sensor. In 2011 IEEE Symposium on computational intelligence in cyber security (CICS) (pp. 202-209). IEEE.

Liou, C. Y., Cheng, W. C., Liou, J. W., & Liou, D. R. (2014). Autoencoder for words. Neurocomputing, 139, 84-96.

Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., & Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. Computer networks, 34(4), 579-595.

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 9(20), 4396.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26.

Lopez-Rojas, E., Elmir, A., & Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. In 28th European Modeling and Simulation Symposium, EMSS, Larnaca (pp. 249-255). Dime University of Genoa.

Lucas, Y., Portier, P. E., Laporte, L., Calabretto, S., Caelen, O., He-Guelton, L., & Granitzer, M. (2019, April). Multiple perspectives HMM-based feature engineering for credit card fraud detection. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (pp. 1359-1361).

Lucas, Y., Portier, P. E., Laporte, L., Calabretto, S., He-Guelton, L., Oblé, F., & Granitzer, M. (2019, June). Dataset shift quantification for credit card fraud detection. In 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) (pp. 97-100). IEEE.

Ludwig, S. A. (2017, October). Intrusion detection of multiple attack classes using a deep neural net ensemble. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-7). IEEE.

Lukasik, S. J. (2000). Protecting the global information commons. Telecommunications Policy, 24(6-7), 519-531.

Lyudchik, O. (2016). Outlier detection using autoencoders (No. CERN-STUDENTS-Note-2016-079).

Mahoney, M. V., & Chan, P. K. (2001). PHAD: Packet header anomaly detection for identifying hostile network traffic.

Mahoney, M. V., & Chan, P. K. (2002, July). Learning nonstationary models of normal network traffic for detecting novel attacks. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 376-385).

Mahoney, M. V., & Chan, P. K. (2003, November). Learning rules for anomaly detection of hostile network traffic. In Third IEEE International Conference on Data Mining (pp. 601-604). IEEE.

Mahoney, M. V., & Chan, P. K. (2003, September). An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In International Workshop on Recent Advances in Intrusion Detection (pp. 220-237). Springer, Berlin, Heidelberg.

Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M. S., & Zeineddine, H. (2019). An experimental study with imbalanced classification approaches for credit card fraud detection. IEEE Access, 7, 93010-93022.

Marir, N., Wang, H., Feng, G., Li, B., & Jia, M. (2018). Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. IEEE Access, 6, 59657-59671.

Maxion, R. A., & Tan, K. M. (2000, June). Benchmarking anomaly-based detection systems. In Proceeding International Conference on Dependable Systems and Networks. DSN 2000 (pp. 623-630). IEEE.

McElwee, S., Heaton, J., Fraley, J., & Cannady, J. (2017, October). Deep learning for prioritizing and responding to intrusion detection alerts. In MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM) (pp. 1-5). IEEE.

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Transactions on Information and System Security (TISSEC), 3(4), 262-294.

Meftah, S., Rachidi, T., & Assem, N. (2019). Network based intrusion detection using the UNSW-NB15 dataset. International Journal of Computing and Digital Systems, 8(5), 478-487.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. (2018). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12-22.

Meira, J., Andrade, R., Praça, I., Carneiro, J., Bolón-Canedo, V., Alonso-Betanzos, A., & Marreiros, G. (2020). Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. Journal of Ambient Intelligence and Humanized Computing, 11(11), 4477-4489.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546.

Miller, P., & Inoue, A. (2003, July). Collaborative intrusion detection system. In 22nd International Conference of the North American Fuzzy Information Processing Society, NAFIPS 2003 (pp. 519-524). IEEE.

Mimura, M., & Tanaka, H. (2017, November). Reading network packets as a natural language for intrusion detection. In International Conference on Information Security and Cryptology (pp. 339-350). Springer, Cham.

Minoli, D., Sohraby, K., & Kouns, J. (2017, January). IoT security (IoTSec) considerations, requirements, and architectures. In 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 1006-1007). IEEE.

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089.

Misra, S., Thakur, S., Ghosh, M., & Saha, S. K. (2020). An autoencoder based model for detecting fraudulent credit card transaction. Procedia Computer Science, 167, 254-262.

MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation. (n.d.). MIT. Retrieved January 31, 2021, from https://archive.ll.mit.edu/ideval/data/index.html

Mittal, S., & Tyagi, S. (2019, January). Performance evaluation of machine learning algorithms for credit card fraud detection. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 320-324). IEEE.

Moustafa, N., & Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective, 25(1-3), 18-31.

Münz, G., Li, S., & Carle, G. (2007, September). Traffic anomaly detection using k-means clustering. In GI/ITG Workshop MMBnet (pp. 13-14).

Nauck, D., Nauck, U., & Kruse, R. (1999, June). NEFCLASS for JAVA-new learning algorithms. In 18th International Conference of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No. 99TH8397) (pp. 472-476). IEEE.

Nawir, M., Amir, A., Yaakob, N., & Lynn, O. B. (2019). Effective and efficient network anomaly detection system using machine learning algorithm. Bulletin of Electrical Engineering and Informatics, 8(1), 46-51.

Net Losses: Estimating the Global Cost of Cybercrime. (2014). CSIS. http://csis.org/files/attachments/140609_McAfee_PDF.pdf

Newton, B. D. (2012). Anomaly Detection in Network Traffic Traces Using Latent Dirichlet Allocation. dated Dec, 31.

Niu, X., Wang, L., & Yang, X. (2019). A comparison study of credit card fraud detection: Supervised versus unsupervised. arXiv preprint arXiv:1904.10604.

Njogu, H. W., & Jiawei, L. (2010, July). Using alert cluster to reduce IDS alerts. In 2010 3rd International Conference on Computer Science and Information Technology (Vol. 5, pp. 467-471). IEEE.

Noble, C. C., & Cook, D. J. (2003, August). Graph-based anomaly detection. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 631-636).

Oka, M., Oyama, Y., Abe, H., & Kato, K. (2004, September). Anomaly detection using layered networks based on eigen co-occurrence matrix. In International Workshop on Recent Advances in Intrusion Detection (pp. 223-237). Springer, Berlin, Heidelberg.

Pambudi, B. N., Hidayah, I., & Fauziati, S. (2019, December). Improving Money Laundering Detection Using Optimized Support Vector Machine. In 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 273-278). IEEE.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. Journal of Computer and System Sciences, 61(2), 217-235.

Parker, L. R., Yoo, P. D., Asyhari, T. A., Chermak, L., Jhi, Y., & Taha, K. (2019, August). Demise: Interpretable deep extraction and mutual information selection techniques for IoT intrusion detection. In Proceedings of the 14th International Conference on Availability, Reliability and Security (pp. 1-10).

Parthasarathy, G., Ramanathan, L., JustinDhas, Y., Saravanakumar, J., & Darwin, J. (2019, February). Comparative case study of machine learning classification techniques using imbalanced credit card fraud datasets. In Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India.

Perdisci, R., Gu, G., & Lee, W. (2006, December). Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In Sixth International Conference on Data Mining (ICDM'06) (pp. 488-498). IEEE.

PerfectStorm. (n.d.). Keysight. Retrieved January 31, 2021, from https://www.keysight.com/us/en/products/network-test/network-test-hardware/perfectstorm.html

Pietraszek, T. (2004, September). Using adaptive alert classification to reduce false positives in intrusion detection. In International Workshop on Recent Advances in Intrusion Detection (pp. 102-124). Springer, Berlin, Heidelberg.

Pillai, T. R., Palaniappan, S., Abdullah, A., & Imran, H. M. (2015, February). Predictive modeling for intrusions in communication systems using GARMA and ARMA models. In 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW) (pp. 1-6). IEEE.

Portnoy, L. (2000). Intrusion detection with unlabeled data using clustering (Doctoral dissertation, Columbia University).

Prabhakara, E., Kumarb, M. N., Ponnarb, K., Sureshb, A., & Jayandhiranb, R. (2019). Credit card fraud detection using boosted stacking. South Asian J. Eng. Technol., 8(S1), 149-153.

Prusti, D., & Rath, S. K. (2019, July). Fraudulent transaction detection in credit card by applying ensemble machine learning techniques. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

Prusti, D., & Rath, S. K. (2019, October). Web service based credit card fraud detection by applying machine learning techniques. In TENCON 2019-2019 IEEE Region 10 Conference (TENCON) (pp. 492-497). IEEE.

Prusti, D., Padmanabhuni, S. S., & Rath, S. K. (2019). Credit card fraud detection by implementing machine learning techniques.

Qayyum, A., Islam, M. H., & Jamil, M. (2005, September). Taxonomy of statistical based anomaly detection techniques for intrusion detection. In Proceedings of the IEEE Symposium on Emerging Technologies, 2005. (pp. 270-276). IEEE.

Qin, M., & Hwang, K. (2004). Frequent episode rules for intrusive anomaly detection with internet datamining. In USENIX Security Symposium (pp. 1-15).

Qin, M., & Hwang, K. (2004). Network Anomaly Detection Against Frequent Episodes of Internet Connections.

Raab, C., & Szekely, I. (2017). Data protection authorities and information technology. Computer Law & Security Review, 33(4), 421-433.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

Radford, B. J., Apolonio, L. M., Trias, A. J., & Simpson, J. A. (2018). Network traffic anomaly detection using recurrent neural networks. arXiv preprint arXiv:1803.10769.

Raj, S. B. E., & Portia, A. A. (2011, March). Analysis on credit card fraud detection methods. In 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET) (pp. 152-156). IEEE.

Ramadas, M., Ostermann, S., & Tjaden, B. (2003, September). Detecting anomalous network traffic with self-organizing maps. In International Workshop on Recent Advances in Intrusion Detection (pp. 36-54). Springer, Berlin, Heidelberg.

Ramos, V., & Abraham, A. (2005). Antids: Self organized ant-based clustering model for intrusion detection system. In Soft Computing as transdisciplinary science and technology (pp. 977-986). Springer, Berlin, Heidelberg.

Rhodes, B. C., Mahaffey, J. A., & Cannady, J. D. (2000, October). Multiple self-organizing maps for intrusion detection. In Proceedings of the 23rd national information systems security conference (pp. 16-19).

Rudd, E. M., Rozsa, A., Günther, M., & Boult, T. E. (2016). A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions. IEEE Communications Surveys & Tutorials, 19(2), 1145-1172.

Rushin, G., Stancil, C., Sun, M., Adams, S., & Beling, P. (2017, April). Horse race analysis in credit card fraud—deep learning, logistic regression, and Gradient Boosted Tree. In 2017 systems and information engineering design symposium (SIEDS) (pp. 117-121). IEEE.

Ryan, J., Lin, M. J., & Miikkulainen, R. (1998). Intrusion detection with neural networks. Advances in neural information processing systems, 943-949.

Safa, M. U., & Ganga, R. M. (2019). Credit Card Fraud Detection Using Machine Learning. International Journal of Research in Engineering, Science and Management, 2(11), 372-374.

Sakurada, M., & Yairi, T. (2014, December). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis (pp. 4-11).

Sarasamma, S. T., & Zhu, Q. A. (2006). Min-max hyperellipsoidal clustering for anomaly detection in network security. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36(4), 887-901.

Sarasamma, S. T., Zhu, Q. A., & Huff, J. (2005). Hierarchical Kohonenen net for anomaly detection in network security. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 35(2), 302-312.

Saxe, J., & Berlin, K. (2017). eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv preprint arXiv:1702.08568.

Sears, D. R., Arzt, A., Frostel, H., Sonnleitner, R., & Widmer, G. (2017). Modeling harmony with skip-grams. arXiv preprint arXiv:1707.04457.

Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., & Zhou, S. (2002, November). Specification-based anomaly detection: a new approach for detecting network intrusions. In Proceedings of the 9th ACM conference on Computer and communications security (pp. 265-274).

Shahed, M., Ibrahim, K., & Akter, P. (2019). Fraud Detection in Mobile Money Transaction: A Data Mining Approach.

Sharmila, V. C., Kumar, K., Sundaram, R., Samyuktha, D., & Harish, R. (2019, April). Credit card fraud detection using anomaly techniques. In 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT) (pp. 1-6). IEEE.

Shin, S., Choi, M., Choi, J., Langevin, S., Bethune, C., Horne, P., ... & Choo, J. (2017, November). Stexnmf: Spatio-temporally exclusive topic discovery for anomalous event detection. In 2017 IEEE International Conference on Data Mining (ICDM) (pp. 435-444). IEEE.

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security, 31(3), 357-374.

Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. Information Sciences, 177(18), 3799-3821.

Shon, T., Kim, Y., Lee, C., & Moon, J. (2005, June). A machine learning framework for network anomaly detection using SVM and GA. In Proceedings from the sixth annual IEEE SMC information assurance workshop (pp. 176-183). IEEE.

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE transactions on emerging topics in computational intelligence, 2(1), 41-50.

Shukur, H. A., & Kurnaz, S. (2019). Credit card fraud detection using machine learning methodology. International Journal of Computer Science and Mobile Computing, 8(3), 257-260.

Shyu, M. L., Chen, S. C., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

Singer, N. F. Y. (1999). Efficient Bayesian parameter estimation in large discrete domains. Advances in neural information processing systems, 11, 417.

Singh, A., & Jain, A. (2019). Adaptive credit card fraud detection techniques based on feature selection method. In Advances in computer communication and computational sciences (pp. 167-178). Springer, Singapore.

Soh, W. W., & Yusuf, R. M. (2019). Predicting credit card fraud on a imbalanced data. International Journal of Data Science and Advanced Analytics (ISSN 2563-4429), 1(1), 12-17.

Srivastava, R. P. (1992, April). Automating judgmental decisions using neural networks: a model for processing business loan applications. In Proceedings of the 1992 ACM annual conference on Communications (pp. 351-357).

Stibor, T., Mohr, P., Timmis, J., & Eckert, C. (2005, June). Is negative selection appropriate for anomaly detection?. In Proceedings of the 7th annual conference on Genetic and evolutionary computation (pp. 321-328).

Stibor, T., Timmis, J., & Eckert, C. (2005, August). A comparative study of real-valued negative selection to statistical anomaly detection techniques. In International Conference on Artificial Immune Systems (pp. 262-275). Springer, Berlin, Heidelberg.

Stolfo, S. J., Hershkop, S., Bui, L. H., Ferster, R., & Wang, K. (2005, May). Anomaly detection in computer security and an application to file system accesses. In International Symposium on Methodologies for Intelligent Systems (pp. 14-28). Springer, Berlin, Heidelberg.

Sun, J., Wang, X., Xiong, N., & Shao, J. (2018). Learning sparse representation with variational auto-encoder for anomaly detection. IEEE Access, 6, 33353-33361.

Sun, L., & Yin, Y. (2017). Discovering themes and trends in transportation research using topic modeling. Transportation Research Part C: Emerging Technologies, 77, 49-66.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2012, April). Unsupervised clustering approach for network anomaly detection. In International conference on networked digital technologies (pp. 135-145). Springer, Berlin, Heidelberg.

Synthetic Financial Datasets For Fraud Detection. (2017, April 3). Kaggle. https://www.kaggle.com/ntnu-testimon/paysim1

Tan, C. L., Quah, T. S., & Teh, H. H. (1996). An artificial neural network that models human decision making. Computer, 29(3), 64-70.

Tan, K. (1995, November). The application of neural networks to UNIX computer security. In Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 1, pp. 476-481). IEEE.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). IEEE.

Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. expert systems with applications, 36(10), 11994-12000.

Tsakanyan, V. (2017). The role of cybersecurity in world politics. VESTNIK RUDN INTERNATIONAL RELATIONS. 17. 339-348.

Udo, G., Bagchi, K., & Kirs, P. (2018). ANALYSIS OF THE GROWTH OF SECURITY BREACHES: A MULTI-GROWTH MODEL APPROACH. Issues in Information Systems, 19(4).

Ullah, F., & Babar, M. A. (2019). Architectural tactics for big data cybersecurity analytics systems: a review. Journal of Systems and Software, 151, 81-118.

Upadhyay, R., & Pantiukhin, D. (2017, September). Application of convolutional neural network to intrusion type recognition. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics, Udupi, India (pp. 13-16).

Vaccaro, H. S. (1988). Detection of anomalous computer session activity (No. LA-UR-88-3656; CONF-890536-2). Los Alamos National Lab., NM (USA).

Van Der Maaten, L. (2009, April). Learning a parametric embedding by preserving local structure. In Artificial Intelligence and Statistics (pp. 384-391). PMLR.

Van Der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. The Journal of Machine Learning Research, 15(1), 3221-3245.

Van der Maaten, L., & Hinton, G. (2012). Visualizing non-metric similarities in multiple maps. Machine learning, 87(1), 33-55.

Van Huong, P., & Hung, D. V. (2019, December). Intrusion detection in IoT systems based on deep learning using convolutional neural network. In 2019 6th NAFOSTED Conference on Information and Computer Science (NICS) (pp. 448-453). IEEE.

Vardhani, P. R., Priyadarshini, Y. I., & Narasimhulu, Y. (2019). CNN data mining algorithm for detecting credit card fraud. In Soft Computing and Medical Bioinformatics (pp. 85-93). Springer, Singapore.

Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019, March). Credit card fraud detection-machine learning methods. In 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH) (pp. 1-5). IEEE.

Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., & Li, K. (2016, April). AI^ 2: training a big data machine to defend. In 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS) (pp. 49-54). IEEE.

Viegas, E., Santin, A., Abreu, V., & Oliveira, L. S. (2017, July). Stream learning and anomaly-based intrusion detection in the adversarial settings. In 2017 IEEE Symposium on Computers and Communications (ISCC) (pp. 773-778). IEEE.

Wagner, A., & Plattner, B. (2005, June). Entropy based worm and anomaly detection in fast IP networks. In 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05) (pp. 172-177). IEEE.

Wang, K., & Stolfo, S. J. (2004, September). Anomalous payload-based network intrusion detection. In International workshop on recent advances in intrusion detection (pp. 203-222). Springer, Berlin, Heidelberg.

Wang, W., Guan, X., & Zhang, X. (2004, August). A novel intrusion detection method based on principle component analysis in computer security. In International Symposium on Neural Networks (pp. 657-662). Springer, Berlin, Heidelberg.

Whitehead, M., & Johnson, D. K. (2017, May). A Tool for Visualizing and Exploring Relationships Among Cancer-Related Patents. In FLAIRS Conference (pp. 235-239).

Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. Applied soft computing, 10(1), 1-35.

Xiao, F., Jin, S., & Li, X. (2010). A novel data mining-based method for alert reduction and analysis. Journal of networks, 5(1), 88.

Xie, Y., & Yu, S. Z. (2008). A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM transactions on networking, 17(1), 54-65.

Xu, X. (2006). Adaptive intrusion detection based on machine learning: feature extraction, classifier construction and sequential pattern prediction. International Journal of Web Services Practices, 2(1-2), 49-58.

Xu, X. (2010). Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies. Applied Soft Computing, 10(3), 859-867.

Xu, X., Sun, Y., & Huang, Z. (2007, April). Defending DDoS attacks using hidden Markov models and cooperative reinforcement learning. In Pacific-Asia Workshop on Intelligence and Security Informatics (pp. 196-207). Springer, Berlin, Heidelberg.

Yan, B., & Han, G. (2018). Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. IEEE Access, 6, 41238-41248.

Yang, K., & Xu, W. (2019, January). FraudMemory: Explainable memory-enhanced sequential neural networks for financial fraud detection. In Proceedings of the 52nd Hawaii International Conference on System Sciences.

Yang, W., Zhang, Y., Ye, K., Li, L., & Xu, C. Z. (2019, June). FFD: a federated learning based method for credit card fraud detection. In International Conference on Big Data (pp. 18-32). Springer, Cham.

Yang, Y., Zheng, K., Wu, C., Niu, X., & Yang, Y. (2019). Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. Applied Sciences, 9(2), 238.

Ye, N. (2000, June). A markov chain model of temporal behavior for anomaly detection. In Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop (Vol. 166, p. 169). West Point, NY.

Ye, N., & Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. Quality and Reliability Engineering International, 17(2), 105-112.

Yeung, D. Y., & Ding, Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. Pattern recognition, 36(1), 229-243.

Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U. (2017, May). Autoencoder-based feature learning for cyber security applications. In 2017 International joint conference on neural networks (IJCNN) (pp. 3854-3861). IEEE.

Yu, Y., Long, J., & Cai, Z. (2017). Network intrusion detection through stacking dilated convolutional autoencoders. Security and Communication Networks, 2017.

Zamani, M., & Movahedi, M. (2013). Machine learning techniques for intrusion detection. arXiv preprint arXiv:1312.2177.

Zanero, S., & Savaresi, S. M. (2004, March). Unsupervised learning techniques for an intrusion detection system. In Proceedings of the 2004 ACM symposium on Applied computing (pp. 412-419).

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, 84, 25-37.

Zhang, G., Iwata, T., & Kashima, H. (2017, September). Robust multi-view topic modeling by incorporating detecting anomalies. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 238-250). Springer, Cham.

Zhang, J., & Zulkernine, M. (2006, June). Anomaly based network intrusion detection with unsupervised outlier detection. In 2006 IEEE International Conference on Communications (Vol. 5, pp. 2388-2393). IEEE.

Zhang, X., Han, Y., Xu, W., & Wang, Q. (2019). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. Information Sciences.

Zhang, Y., Lee, W., & Huang, Y. A. (2003). Intrusion detection techniques for mobile wireless networks. Wireless Networks, 9(5), 545-556.

Zhang, Z., Li, J., Manikopoulos, C. N., Jorgenson, J., & Ucles, J. (2001, June). HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In Proc. IEEE Workshop on Information Assurance and Security (Vol. 85, p. 90).

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing, 115, 213-237.

Zhou, C., & Paffenroth, R. C. (2017, August). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 665-674).

Zhou, Z., Fu, B., Qiu, H., Zhang, Y., & Liu, X. (2017, April). Modeling medical texts for distributed representations based on Skip-Gram model. In 2017 3rd International Conference on Information Management (ICIM) (pp. 279-283). IEEE.

Zhuo, X., Zhang, J., & Son, S. W. (2017, December). Network intrusion detection using word embeddings. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 4686-4695). IEEE.

8. Appendices

8.1. UNSW-NB 15 dataset feature set

Id.	Name	Type	Description
1	srcip	nominal	Source IP address.
2	sport	integer	Source port number.
3	dstip	nominal	Destination IP address.
4	dsport	integer	Destination port number.
5	proto	nominal	Transaction protocol.
6	state	nominal	Indicates to the state and its dependent protocol.
7	dur	float	Record total duration.
8	sbytes	Integer	Source to destination transaction bytes.
9	dbytes	integer	Destination to source transaction bytes.
10	sttl	integer	Source to destination time to live value.
11	dttl	integer	Destination to source time to live value.
12	sloss	integer	Source packets retransmitted or dropped.
13	dloss	integer	Destination packets retransmitted or dropped.
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service.
15	Sload	float	Source bits per second.
16	Dload	float	Destination bits per second.
17	Spkts	integer	Source to destination packet count.
18	Dpkts	integer	Destination to source packet count.
19	swin	integer	Source TCP window advertisement value.
20	dwin	integer	Destination TCP window advertisement value.
21	stcpb	integer	Source TCP base sequence number.
22	dtcpb	integer	Destination TCP base sequence number.
23	smeansz	integer	Mean of the packet size transmitted by the src.
24	dmeansz	integer	Mean of the packet size transmitted by the dst.
25	trans_depth	integer	Pipelined depth into the connection of http request/response transaction.
26	res_bdy_len	integer	Uncompressed content size of data transferred from the http server.
27	Sjit	float	Source jitter (ms).

28	Djit	float	Destination jitter (ms).
29	Stime	timestamp	Record start time.
30	Ltime	timestamp	Record last time.
31	Sintpkt	float	Source interpacket arrival time (ms).
32	Dintpkt	float	Destination interpacket arrival time (ms).
33	tcprtt	float	TCP connection setup round-trip time: sum of synack and ackdat.
34	synack	float	TCP connection setup time: time between SYN and SYN_ACK.
35	ackdat	float	TCP connection setup time: time between SYN_ACK and ACK.
36	is_sm_ips_ports	binary	If source (1) and destination (3) IP addresses equal and port numbers (2) (4) equal then 1 else 0.
37	ct_state_ttl	integer	Number for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	integer	Flows with methods such as GET and POST in http service.
39	is_ftp_login	binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	Flows with a command in ftp session.
41	ct_srv_src	integer	Connections with same service (14) and source address (1) in 100 connections according to last time (26).
42	ct_srv_dst	integer	Connections that contain the same service (14) and destination address (3) in 100 connections according to last time (26).
43	ct_dst_ltm	integer	Connections of the same destination address (3) in 100 connections according to last time (26).
44	ct_src_ ltm	integer	Connections of the same source address (1) in 100 connections according to last time (26).
45	ct_src_dport_ltm	integer	Connections with same source address (1) and destination port (4) in 100 connections according to last time (26).
46	ct_dst_sport_ltm	integer	Connections with same destination address (3) and source port (2) in 100 connections according to last time (26).

47	ct_dst_src_ltm	integer	Connections with same source (1) and destination address (3) in 100 connections according to last time (26).
48	attack_cat	nominal	Name of each attack category.
49	Label	binary	0 for normal and 1 for attack records.

8.2. Paysim1 dataset

Name	Type	Description
step	numeric	A unit of time. 1 step is 1 hour of time. Total steps 744 (30 days).
type	categorical	CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
amount	numeric	Amount of the transaction in local currency.
nameOrig	categorical	Customer who started the transaction.
oldbalanceOrg	numeric	Initial balance before the transaction.
newbalanceOrig	numeric	New balance after the transaction.
nameDest	categorical	Customer who is the recipient of the transaction.
oldbalanceDest	numeric	Initial balance recipient before the transaction.
newbalanceDest	numeric	New balance recipient after the transaction.
isFraud	binary	Flag: legitimate transaction, fraudulent transaction.
isFlaggedFraud	binary	Flag: an attempt to transfer more than 200.000 in a single transaction.

8.3. IoT-23 MC11 dataset

Name	Type	Description
ts	time	Timestamp.
uid	string	Unique identifier.
id.orig_h	addr	Source IP address.
id.orig_p	port	Source port.
id.resp_h	addr	Target IP address.
id.resp_p	port	Target port.
proto	enum	Protocol.

service	string	Service.
duration	interval	Connection duration.
orig_bytes	count	Bytes sent by source IP address to target IP address.
resp_bytes	count	Bytes sent by target IP address to source IP address.
conn_state	string	Connection state.
local_orig	bool	Local source address.
local_resp	bool	Local target address.
missed_bytes	count	Missed bytes.
history	string	History.
orig_pkts	count	Packets sent by source IP address to target IP address.
orig_ip_bytes	count	IP bytes sent by source IP address to target IP address.
resp_pkts	count	Packets sent by target IP address to source IP address.
resp_ip_bytes	count	IP bytes sent by target IP address to source IP address.
tunnel_parents	set[string]	Tunnel parents.
label	string	Label: Benign, Malicious.
detailed_label	string	Detailed label.