UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

Facultatea de Matematică și Informatică
Universitatea din București

UNIVERSITY OF BUCHAREST

FACULTY OF MATHEMATICS AND INFORMATICS

MASTER'S DEGREE IN SECURITY AND APPLIED LOGIC

Master's Thesis

BIG DATA-DRIVEN APPROACHES FOR IMPROVING INTRUSION DETECTION

Supervisor

Lect. Dr. LETIȚIA MARIN

Graduate

SIMONA POP

Bucharest 2023

Security and Applied Logic

Master's Thesis

BIG DATA-DRIVEN APPROACHES FOR IMPROVING INTRUSION DETECTION

Supervisor

Lect. Dr. LETIȚIA MARIN

Graduate

SIMONA POP

Bucharest 2023

Abstract

This is a research paper that aims to explore how big data can be leveraged to enhance intrusion detection systems. Apart from an introduction to the existing preoccupations of security professionals who commit their attention to the discipline of malicious acting identification, which includes a background of this cybersecurity sphere, its specialised taxonomy, and state-of-the-art solutions, the present paper makes its own contribution to the vast study area by presenting two demonstrative implementations for practical experimentation. With the primary objective of understanding exactly what these big data-based approaches entail through hands-on experience, the machine learning experiments contained in this paper concentrate on the concrete addition that big data and machine learning bring to traditional methods as well as the possible drawbacks that must be assumed when engaging with them.

Contents

Preliminaries

Intrusion Detection in Cybersecurity

Many years have already passed since mankind started to intuit the great inherent risks that it was exposing itself to in exchange for technological advancements. As this trade persists, there is a growing realisation that these risks take many forms and permeate various facets of our existence. They range from the disputably nuanced shifts in societal values attributed to modern lifestyles all the way to the unequivocally perilous repercussions of cyber threats, in the realm of which topic resides the vehicle for the research background of the paper at hand.

Examining the sophisticated panorama of the actual networks covering this interconnected world, one can easily become aware of the fact that every participant is innately vulnerable to possible security breaches and targeted assaults. Regardless of your role or position within a network, even if you are just an individual with a smartphone, your digital persona and virtual property are prone to being compromised and exploited, which may lead to severe consequences for your offline everyday life as well (financial and intellectual property loss, invasion of privacy, deprivation of control over personal assets, and reputation damage are only some of the most common examples). As Gene Spafford has put it into words ever since 1989, "The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards - and even then I have my doubts." [9]

Yet, there is an even broader perspective to keep in mind. Companies and corporations, compared to individuals, are generally prone to a greater magnitude of harmful and detrimental, sometimes even devastating, consequences. A multitude of threats, unremittingly wearing novel disguises due to zero-day attacks and ingenious evasion techniques, are permanently parading on the stage of challenges that a business must be prepared to face day by day. Or, like former FBI Director Robert Muller once decreed, "There are only two types of companies: those that have been hacked, and those that will be."

To march modestly through history, it can be noted that, in early stages, both the limited accessibility to computers and the substantial expenses involved in their construction

and maintenance acted as constraints on security-related concerns. However, as the volume of sensitive information entrusted to these systems grew and remote access capabilities expanded, the vulnerabilities in security began to surface.

Recalling ARPANET, the first network to use packet switching technology and often considered the precursor of the Internet, highlights the comparatively simpler nature of network security during its era. This simplicity stemmed from the restricted community for which ARPANET was designed, as well as the relatively low significance attributed to the information transmitted through its communication protocols. Furthermore, in those days, the handshake agreement formed the foundation of connections, the number of account owners was infinitesimal, and the network users were generally familiar with one another. But the high-trust culture that characterised the network community of that time has vanished along with the early Internet age. [4]

Attempting to pinpoint the precise moment when the concept of intrusion detection emerged is a tremendous task given its vast scope. It is instinctive to draw parallels with more tangible areas, such as burglar alarms for a house, video-monitoring systems endowing stores and all kinds of institutions, or even civil defence and military activity requiring a certain level of systematic tracking and observation. In fact, cybersecurity analysts often rely on a familiar analogy to illustrate the importance of their field to the large public: the analogy of the house alarm. Suppose an unfamiliar individual stands outside a residence, carefully observing and attempting to gain entry through the doors and windows. Even with robust security measures in place, such as locked entrances, the decision to install an alarm system remains crucial. This parallel question often arises in the context of intrusion detection: why invest in detecting unauthorised access when firewalls, patched operating systems, and stringent password protocols are in effect? The answer is straightforward: intrusions can still transpire. Just as one may inadvertently overlook securing a window, even the most advanced protective measures can possess vulnerabilities. Recognising that achieving absolute system security is an elusive goal, intrusion detection techniques and systems are imperative to identify and respond to computer attacks. [26]

When it comes to intrusion detection systems, the prolific research and development date from around the year 1980. Anyhow, a more comprehensive exploration of the historical background, together with a more meticulous overview of the applied concept, will be presented in Chapter 1.

Nevertheless, it is essential to note that nowadays, SIEMs (*Security Information and Event Management Systems*) represent the industry standard of security threat protection for large organisations, as they are capable of providing some level of orchestration. SIEMs operate as centralised systems that gather, examine, and correlate records of security events from a

broad range of sources, including IDSs (*Intrusion Detection Systems*) and IPSs (*Intrusion Prevention Systems*), which are often incorporated within their architecture, as well as other security systems and tools, like UEBA (*User and Entity Behaviour Analytics*) for anomaly detection in corporate network users behaviour.

Big Data, Data Mining, and Machine Learning

The collection of all digital information generated by contemporary technology for both personal and professional objectives is defined as big data. Enterprise data, sensor data, online content, e-commerce transactions, social media interactions, data from connected devices, geolocation data, and more are all included in this category of data sources. The term "Big Data" first appeared in 1997, and Doug Laney from Meta Group, the actual Gartner [1], is credited with giving it a definition based on three fundamental characteristic principles (the three Vs, i.e., massive *volume*, wide *variety*, real-time *velocity*) in 2001. Since then, several other Vs joined the family of big data properties (like value, veracity, viability, and so on) [6].

Innovations in storage, data processing instruments, the introduction of cloud computing, and the occurrence of supercomputers have collectively contributed to the expansion of big data. Chapter 1 will feature a more thorough analysis of recent progress and innovations made possible by modern big data handlers, as well as brief remarks on how big data processing technologies shaped this field.

Eventually, it has been ascertained that handling Big Data represents the groundwork of cybersecurity analysis and analytics, whether we are considering a single source of events (e.g., network traffic) alone or an integral architecture accumulating events of heterogeneous origins. Needless to say, in the latter scenario, by correlating the aggregated events, a more holistic perspective and a greater circumstantial understanding can be ensured, at the expense of confronting even more ruthless Big Data challenges. [28]

All these things considered, it becomes clearer why data mining and machine learning may be captured in the picture of cybersecurity, as the paper at hand, among many other scientific surveys and specialist literature, proposes. For intrusions to be detected, certain data patterns must be uncovered, and meaningful linkages, insights, and trends must be identified for further predictions or decisions to be made. Additional related work preaching the engagement of machine learning in the process of intrusion detection will also be recalled in Chapter 1.

The purpose of the present research project is to examine the current state of machine learning-based cybersecurity solutions as well as their benefits and limitations compared to more conventional ones.

Chapter 1

Literature Review

1.1 The Cybercrime Backdrop

In the past, highly competent and dedicated hackers were the most prominent perpetrators of cyberattacks. However, nowadays, malicious actors with little technical expertise may also execute sophisticated assaults by virtue of technological advancements and the wide availability of hacking devices. The amount and variety of cyber risks and the rate of organised cybercrime have soared as a result of this democratisation, making it harder to traverse the cybersecurity landscape. Not only that, as technologies and trends like cloud infrastructures, SaaS (software as a service), and BYOD (bring your own device) continue to expand, the boundaries of enterprises have begun fading.

This section is also the most adequate to mark down the notion of APT (*Advanced Persistent Threat*), which refers to a targeted and laborious cyberattack, typically carried out by proficient and persistent adversaries (e.g., organised cybercriminal groups). APTs are characterised by their long duration and surreptitious nature, as well as their tendency to compromise high-value entities, often with objectives like espionage, data theft, or disruption of critical systems.

In accordance with the annual Internet Security Threat Report of 2017 issued by Symantec (now a division of Broadcom) [15], throughout the year of 2016, over three billion zero-day attacks were tracked down, and both their number and severity were significantly higher than they had been in the past. The same report highlights the fact that, while prior cybercriminals fundamentally targeted clients of banks by hijacking credit cards or bank accounts, the new malware generation has cultivated sharper skills and is now attacking banks directly, occasionally attempting to steal millions of dollars in a single hit.

Not only is the attack landscape in permanent bloom, but evasion techniques are gaining momentum too. Some common evasion tactics include disabling security tools through exploiting vulnerabilities or social engineering, masquerading to conceal malicious activity under legitimate appearances, obfuscating malicious code through encryption, encoding, and packing, employing encryption and tunnelling to hide communication, launching timing attacks with specific intervals or delays, resource exhaustion by overwhelming systems with traffic or false alarms, traffic fragmentation to confuse security systems, protocol-level misinterpretation by manipulating packet headers or implementing non-standard protocols, and traffic substitution and insertion. [19]

All these records, together with the reports of even more recent years, convey the great priority that must be attributed to elaborating powerful intrusion detection mechanisms that successfully diagnose zero-day attacks. As per the Handbook of Research on Machine and Deep Learning Applications for Cyber Security [10], the only viable solution to accomplish this objective is the attachment of ML to the detection schema.

1.2 Intrusion Detection - History, Overview, Taxonomy

First and foremost, formal definitions must be assigned to the terms in discussion. Intrusion refers to any unauthorised activities that impair an information system, jeopardising its *confidentiality*, *integrity*, or *availability*, i.e., its *CIA triad* (e.g., routines that prevent legitimate users from using computer systems). An intrusion detection system (IDS) is a software or hardware device that recognises hostile behaviour on computer systems and aids in achieving the overarching objective of preserving system security. [18] In more distinct words, an IDS aims to detect various forms of malicious network traffic and unauthorised computer usage that may go unnoticed by traditional firewalls. Its primary objective is to provide robust protection against activities that pose a risk to the CIA triad.

1.2.1 Intrusion Detection Timeline

Since one custom cannot be truly mastered without perceiving and assimilating its past, a modest timeline of the stages that the subject of intrusion detection has gone through during its short history has been compiled with the help of the following surveys: [26], [24], [5].

- **Phase I: The Early Attempts (1967-1986)**

  During this phase, initial efforts were launched to detect intrusions through the use of audit trails, passwords, and encryption. These strategies depended on the idea that malicious actors would behave differently from genuine users and would leave behind some sort of evidence. However, they proved ineffective in stopping or identifying elaborate

attacks. Back then, system administrators performed their duties manually via a console, through which they monitored user activity. To spot potential security breaches, they relied on observations like employees that were logged in during their time off or abnormal activity from a seldom used printer. While satisfactorily efficient at the time, this strategy lacked scalability and was ad hoc.

Eligible research and official IDS model prototypes were introduced by James P. Anderson (1972) and Dorothy Denning, together with Peter Neumann (1986), with their IDES (*Intrusion Detection Expert System*). They characterised the notion of *misuse detection* as the process of comparing observed events to known and predefined attack patterns. In contrast to it, they also presented the concept of *anomaly detection*, which compares observed events with typical user or system behaviour patterns. Later, different numerous projects were implemented, including some of commercial purposes.

- **Phase II: The Advancement of Techniques (1987-1991)**

  Whilst in the late '70s and early '80s system administrators used to review audit logs printed on fan-folded paper, which was terribly time-consuming due to their voluminous and noisy nature, this phase marks the advancement of intrusion detection with the emergence of automated tools for audit logs. These tools were programmes that analysed audit logs and generated alerts or reports, which helped reduce the workload of system administrators, but also posted the problems of false-positives, false-negatives, and scalability.

  In this phase as well, UC Davis developed the NSM (*Network Security Monitor*) as the first network-based intrusion detection system. NSM employed packet capture and analysis to detect network attacks like port scanning and SYN flooding. It introduced a distributed architecture with sensors deployed at various locations and reporting to a central manager.

- **Phase III: The Growth of Diversity (1992-1999)**

  The gain in complexity when it comes to intrusion detection is reflected in this phase as a genuine reaction to the same expansion tendency posed by diversity in networked setting. The occurrence of distributed systems, wireless networks, and the Internet itself paved the way for both the emancipation of malicious actors and their opposition. This is the era of DoS (Denial-of-Service) attacks, worms, and viruses, on the one hand, and of neural networks, data mining, and correlation analysis, on the other hand. Additionally, it was in the early '90s when researchers created real-time IDSs (audit material was examined as it was being produced). Thanks to this ability of grasping security incidents and violation attempts with little delay, real-time response and sometimes even preemption were made possible.

This phase also prides itself on accommodating Cisco's NetRanger (1997), the first commercial network-based IDS. Making use of signatures and heuristics for network attack detection and integrating firewalls and routers for response, it brought a modular architecture, allowing flexibility in adding or removing components as required.

- **Phase IV: The Establishment of Standards (2000-2005)**

  The constantly changing threat landscape began to make itself felt more and more acutely because of zero-day attacks and polymorphic malware. On the other side of the fence, the key aspects that asked for a place at the table of ID were scalability, interoperability, standardisation, evaluation, and legal issues. The ID area perseveres in exploring of potential in adaptive systems, cooperative systems, and proactive approaches.

  The in-vogue advancement at that time had just been established by Snort [2] in 1998 (the year when the first international conference on intrusion detection, RAID, was also held), which inspired a shade of accessibility and customisation. The first open-source network-based intrusion detection system, developed by Martin Roesch, proposed a modular architecture that allowed users to tailor its functionality through plugins and rules. On top of that, it pioneered the principle of community-driven development, where users could participate and impart their experience. Numerous other frameworks and standards were given birth to; from which CIDF (*Common Intrusion Detection Framework*) and IDMEF (*Intrusion Detection Message Exchange Format*) are worth mentioning. This was the era of research exchange flourishing.

- **Phase VI: The Transition to Prevention (2006-2010)**

  The quintessence of this period is the transition from detection to prevention. IPSs detach from the passive character of IDSs, embracing an active behaviour in relation to computer security threats. They materialised as ample programmes that incorporated firewalls, IDSs, and anti-virus functionalities altogether. The principle behind them consists in obstructing or altering harmful traffic in line between the network and the Internet before it reaches its target. To boost the preventive capabilities of IDSs, abundant technologies like host-based IPSs (protecting devices and individual hosts), network-based IPSs (protecting network segments or subnets), inline IPSs (operating in line with the network traffic and blocking or modifying packages on the fly), and hybrid IPSs were put forward.

- **Phase VII: The Evolution of Paradigms (2011-2015)**

  The focus of this period falls upon the encompassing dynamic character of modern network architectures and the inferred responsibility of both IDSs and IPSs. Taking a look at

the surroundings, one instantly notices the rise of cloud computing, mobile computing, and the IoT (*Internet of Things*) that took control over that period and requested the founding of paradigms. During this time, several approaches rose to prominence, including software-defined networking, data fusion, machine learning, and artificial intelligence.

- **The Exploration of Innovations (2016-present)**

  This final stage deals with ongoing trends and obstacles that must be surpassed by the fields of intrusion detection and prevention, respectively. Some of the major issues that must be confronted in this phase are scalability, accuracy, efficiency, and privacy. At the same time, deep learning, blockchain, quantum computing, and bio-inspired computing are some of the breakthroughs that the area continues to pursue as potential paths for further study and development.

1.2.2 SIDS versus AIDS: A Comparative Evaluation

An **SIDS** (*Signature-based Intrusion Detection System*), also known as *knowledge-based detection* or *rule-based detection* system (notions derived from the classic *misuse detection*), engages with pattern-matching approaches to recognise previously recorded attacks. As is intuitive, SIDSs rely on a database of signatures (rules) that have been constructed based on the knowledge of classic attacks and through scrutinising the traits, actions, or patterns exhibited by past security incidents. These signatures can involve different features, for example, particular tendencies in network traffic, malicious code fragments, peculiar system behaviour, or other indicators of compromise. When incoming network traffic or system logs captured by an SIDS match the signatures stored in the aforementioned database, an alarm signal is emitted. [16]

To simplify understanding of this rule-based practice, Snort [2] may be put under the spotlight. The rules that it focuses its functionality on possess the following structure: one header section (indicating the action, the protocol, the source and destination IP addresses and ports), and one options section (to specify the payload and metadata). Below is a fictional Snort rule for the sole purpose of presenting a concrete example:

```
reject udp 84.232.193.207 1035 -> any any ( msg:"Example"; priority:2; sid:4; rev:2222; )
```

With regard to the advantages and disadvantages of SIDSs, it is crucial to take note of their excellent capacity for accurately identifying well-known attacks but also of their terribly unsatisfying results when confronted with zero-day attacks, malicious behaviours that are not configured in the signature database, or variations in the previously observed template of a known intrusion. This is the reason why frequent updates to the signature database are absolutely vital to the proper real-world functioning of SIDSs. Despite this significant drawback, motivation to opt for SIDSs is fuelled by the convenience of adjusting while upholding

preconfigured rules, as well as the low computational cost, simple design, and high accuracy. [21]

Traditional misuse detection methods also struggle to expose multi-packet attacks. As such, modern sophisticated malware requires from an SIDS the analysing of signatures across multiple packets and recalling earlier packet contents. Moreover, it is worth specifying that signatures can be created under any of the following forms: state machines, string patterns, or semantic conditions. [16]

To sum up the aspects above and draw a frugal conclusion, traditional SIDSs master known threat detection but are put to the test by targeted or multi-step attacks (APTs), polymorphic malware, zero-day attacks. This limitation is the one that AIDSs intends to combat.

An **AIDS** (*Anomaly-based Intrusion Detection System*) aims to be able to distinguish unknown attacks or novel behaviours that deviate significantly from established norms or baselines and do not align with standard or expected flow. It basically replaces the principle of using predefined malice patterns with the concept of designating normal ways of acting and scouring for anomalies. These anomalies represent the deviations of the observed events from the habitual model designed by system administrators and cybersecurity experts via machine learning, statistical-driven methods, or even knowledge-guided means. This model-designing stage is called training, and AIDSs can be further classified according to the training procedure used (which will be one of the above three). The testing stage brings a fresh data set to assess the system's ability to extrapolate to formerly undiscovered attacks. [16]

To make things more clear, AIDSs may collect data on the behaviour of authorised users over time and apply statistical tests to determine if the observed actions performed by a particular user at a given moment in time are legitimate or not. [21]

Now, weighting the upsides and downsides of AIDSs, the most trivial remark that can be made when contemplating why one would choose an AIDS over an SIDS is that AIDSs have been purposely created to realise when they are dealing with previously unencountered intrusions. They are capable of discerning damaging internal activities too. For instance, alarms will certainly be raised for intruders conducting transitions on a hacked account since this would pass as irregular user activity. By using customised profiles, the AIDS can better distinguish between normal user or system behaviour and suspicious activity, making it tougher for cybercriminals to go unnoticed without triggering an alert. [16]

On the down side, AIDSs are prone to confusing new types of legitimate practices for malicious actions, so they tend to generate a disturbingly high rate of false positives, which may lead to major drawbacks to a business' production environment, as the organisation would

have to interrupt the workflow, isolate the presumed-to-be-compromised devices, and allocate time to deeply examine an incident that is actually innocent. An AIDS doesn't comprehend that abnormal doesn't always carry a negative connotation. [10]

Furthermore, the fact that AIDSs require an initial training phase and the difficulty of building customised profiles for highly dynamic computer systems are counted as impediments. Likewise, the fact that the alerts come unclassified and the AIDS' inability to process encrypted packets (potentially causing attacks to remain undiscovered) pose heavy hazards. [16]

1.2.3 Scratching the surface on NIDS and HIDS

Digging deeper into the manners of categorising IDSs, the input data source does also play a consecrated criterion. Relative to this canon, the two main types of IDSs are HIDSs (*Host-based Intrusion Detection Systems*) and NIDSs (*Network-based Intrusion Detection Systems*).

As its name suggests, an **HIDS** monitors the activities on a single host, such as a computer or server, focusing its attention on the host's system calls, file system, and audit logs originating from the operating system, firewalls, application systems, and databases. Hence, suspicious activity, unauthorised access attempts, and other signs of possible intruders or malicious intentions directed at the host system can all be found using HIDSs. The thorough analysis of these numerous data sources and the fact that this IDS type is also capable of inspecting end-to-end encrypted transmissions offer a tiered defence strategy and improve the host environment's overall security posture. Clearly, there are inconvenient aspects of this model too, among which are the consumption of host resources and the narrow area of visibility. [16]

In contrast, an **NIDS** keeps track of the traffic on an entire network, searching for any signs of intrusion captured beneath extracted network packets (TCP, UDP, and ICMP), router NetFlow records, and data of other provenances. It provides network-wide monitoring capabilities without needing to be installed on every participating host, although dedicated hardware is solicited. On top of that, it allows the detection of external malicious activities at an early stage, before they propagate to other systems. Yet, NIDSs struggle with encrypted traffic, and due to the high volume of data flowing through modern high-speed networks, they may face limitations in inspecting all data effectively, particularly in high-bandwidth environments. [16]

When used together with firewalls, multiple HIDSs strategically disposed at various positions within a network topology, along with an NIDS, can provide robust and multi-tiered protection against both external and insider threats. Being complementary components, together they contribute to an effective and comprehensive defence strategy, safeguarding critical

assets and ensuring the integrity, confidentiality, and availability of network resources.

1.3 Anomaly-based Intrusion Detection

As admirably structured in the International Journal of Scientific and Research Publications [14], academic literature tends to endorse the following taxonomy given the subject of AIDSs:

1. *Statistical-based Anomaly Detection Systems*

   This category concerns statistical modelling's contribution to the development of AIDSs. The technology revolves around the principle of putting statistical properties and tests in the service of comparing observed behaviour with expected behaviour. By analysing statistics-oriented particularities (like mean and variance) of ordinary activities and constructing normal profiles relative to them, statistical-based AIDSs are in charge of further performing statistical tests to determine significant deviations from these normal profiles. An alert is set off by the IDS when it attributes an unusual activity a score that is higher than a specific threshold.

2. *Knowledge-based Detection Mechanisms*

   One can recall this versatile knowledge-based principle from the SIDSs review drawn up earlier. This time, the technique implies building AIDS-specific legitimate traffic profiles starting from a knowledge base. Having stated that, the primary distinction between knowledge-based AIDSs and SIDSs is the information source used to identify intrusions. While an SIDS employs a database of known attack signatures, a knowledge-based AIDS entails expert knowledge to build a baseline of typical behaviour. To put it differently, SIDSs seek to map fresh observations to entries in the rules database, but knowledge-based AIDSs are designed to spot deviations from the standard of conduct.

3. *Data Mining and Machine Learning Driven Detection Methods*

   As contoured subtly earlier in the paper at hand, the optimal choice for an IDS lies in harnessing the power of Data Mining, a technique centred on pattern finding, i.e., the extraction of valuable and previously unnoticed models or patterns from vast data repositories. By employing Data Mining, the IDS can reduce the volume of data required for historical network activity comparisons, resulting in more meaningful data for anomaly detection.

1.3.1 Statistical-based Anomaly Detection Systems

The list of globally accredited advantages of opting for a statistically oriented AIDS has circulated through specialty literature more or less in the following form: [14]

- Advanced detection and precise alerting: Statistical AIDSs excel in spotting malicious activity in spite of lacking prior knowledge of the system's vulnerabilities and common attacks' signatures, which makes them efficient in uncovering zero-day attacks. They are also capable of accurately detecting prolonged malicious initiatives and APTs, as well as supplying particular notifications of detrimental actions that unravel gradually over time, resulting in successful anticipation of approaching attacks like DoS. Their granular capabilities ensure early recognition and effective response, as they scrutinise individual elements of intrusions without waiting for the completion of entire sequences. Statistical AIDSs make it possible for unusual suspicious activity to immediately activate alerts, allowing prompt reaction to potential threats. They are especially acclaimed for their results with "low and slow" attacks.

- Simplified maintenance: Ease of maintenance is ensured by reducing the need for human contribution. As previously brought up, deprived of a signature database, no periodic interventions have to be made by the cybersecurity analysts and system administrators, which is certainly lowering their efforts.

To examine the disadvantages of the same anomaly-based solution, the following observations were collected: [14]

- Insufficient behaviour modelling: Pure statistical modelling and solely statistical ways of operating cannot appropriately capture all behaviours.

- Learning time and high false positive rate: Not only is the amplitude of the learning process exhausting and time-consuming for statistic-based AIDSs, but this system also encounters complications when dealing with fluctuations in user behaviour.

- Quasi-stationary assumption: It is possible for statistical AIDSs to be unreliable if the core belief they were fabricated under is that the statistical characteristics of normal behaviour stay largely consistent or steady throughout time. In real-world scenarios, data (people's habits, patterns of system usage, and network activities) can be dynamic and subject to alterations over time.

- Threshold setting challenges: Selecting an adequate threshold can be an exhausting task. It must be kept in mind that a low threshold might lead to an excessive degree of false positives, while a high threshold could overlook important signals.

To further explore the ramifications of the category in discussion, the taxonomy outlined in [14] proves again of great utility. Presented below are the classes of AIDS with a statistical orientation, which are typically covered in scholarly publications:

(a) *The Operational or Threshold Model*

As suggested by the name, this model relies on the presumption that anomalies may be detected by comparing observations with predefined limits. Its baseline depends on alarms being raised on the principle of event quantity over a time interval. It works effectively for metrics when specific values are regularly linked to intrusions, such as identifying unsuccessful log-in attempts when the number of password failures reaches a predetermined threshold.

The model turns out to be suitable when the normal behaviour remains relatively stable and the tolerance level for a certain event is known beforehand. However, in cases where the adversarial activity resides in more than one event (i.e., when a series of events watched all together indicate malicious intent), where setting upper and lower boundaries for some data is not meaningful to the desired study, or where legitimate behaviours periodically undergo modifications, the model is not applicable.

(b) *The Markov Process or Marker Model*

This model analyses the event counter metric by taking the prior events' sequence into account, thereby assessing the normalcy of particular events. Each incremental observation is treated as a state, and a state transition matrix is used to determine if an event's probability is high (showing normal behaviour) based on the antecedent events. Markov chains periodically monitor the system and record its state. When a state change occurs, the model computes the likelihood for that state at the given time frame, and if that turns out to be low, the event is labelled as anomalous.

Hence, the Markovian model is useful when the activities' order is relevant, in exchange for a sizeable computational requirement when building user profiles. So it should not be applied in scenarios where time is a constraint or significant user pattern shifts are at stake.

(c) *The Mean and Standard Deviation Model*

Classic statistical methods endow this model with the power to evaluate the normalcy of an observed event by sinking it into a confidence range. Intuitively, those events that fall outside the fixed range are labelled as anomalies.

One significant advantage of this model lies in its adaptability and potential to adjust to

a user's behaviour over time instead of depending on initial knowledge of patterns. Unlike threshold models, it does not require predefined limits to be set but rather gains insights into the standard workflow from observed data, so it presents wider flexibility compared to threshold-based approaches. On the downside, this model's more intricate and complex structure may complicate its implementation and place maintenance obstacles in the way.

(d) *The Multivariate Model*

Similar to the previous univariate model, this one also operates on traditional statistics, but instead of taking into consideration only one measure, it aims to discover connections among multiple measures. In situations where experimental data demonstrates that integrating correlated variables improves classification accuracy over examining them separately, the model is truly valuable.

Failure rates of multivariate models are considerably inferior to those of univariate models, and adversary events can be discovered in really early stages. However, the resource-intensive nature of these multivariate process control strategies falls short of meeting the requisites of intrusion detection, which demands the management of extensive, high-dimensional data volumes, including various and redundant activity measurements.

(e) *The Time Series Model*

The elementary principle of time series boils down to a sequence of observations collected over a time frame. Similar to the Markov process, one event is labelled abnormal if the computed probability of it occurring in that time interval is actually too low.

This model proves advantageous when circumstances demand future forecasting with respect to current findings and can evaluate trends in behaviour over time, successfully identifying moderate yet substantial alterations in behaviour, making it appropriate for detecting series-based attacks. Its weaknesses are similar to those enumerated for the Markovian process model.

1.3.2 Knowledge-based Detection Mechanisms

Similarly to the previous discussion, in terms of benefits and limitations of this method [14], one should note the below advantages:

- Favourable accuracy: This method brings a reduction in false positives.

- Key features: The technique is robust, flexible and scalable.

- In-depth knowledge gathering: The amount of details to which the knowledge collection

rises eases the human cybersecurity authorities preventative and remedial process.

The limitations that must be taken into account can be found in the following lines:

- Resource-intensive maintenance: There is a constant need for the knowledge base to be refined regularly in order to reflect up-to-date information. This method also demands overwhelming human efforts in terms of analytics and knowledge management.

- Generalisation issue: Although this AIDS type may be able to detect so far undiscovered attacks, its effectiveness is constrained by the knowledge base that serves as the baseline.

Descending further down this branch, presented below are the classification criteria for knowledge-based anomaly detection mechanisms, as described in [16]:

(a) *The Finite State Machine (FSM)*

This is a computation model for execution flow representation and control. Generally speaking, the model is illustrated as an ensemble of states, transitions between states, and activities that can be performed in each state. An FSM can be used to describe a system's lawful behaviour in the context of anomaly detection, i.e., the knowledge base. The model's states indicate various system phases or circumstances, and transitions between states signify modifications to the system's behaviour. Each state's activities correspond to the practices that are seen as acceptable or typical for that state. The actual system's behaviour, which is being observed, is compared against the model represented by the FSM, and every discrepancy between the observed flow and the one prescribed by the model is labelled as an anomaly. For instance, consider a straightforward FSM that simulates the actions of a web server: "Idle" and "Processing Request" are the two states that the FSM can be in. The server is awaiting incoming requests while it is in the "Idle" state. After receiving a request, the server enters the "Processing Request" state and begins processing the request. The server switches back to the "Idle" state after processing the request. The server would stay in the "Processing Request" state for an extended period if an attacker tried to overwhelm it with requests, and it would never return to the "Idle" state. This departure from the FSM's representation of usual behaviour would be recognised as an anomaly and can indicate a denial-of-service.

(b) *Expert System*

An expert system may establish both SIDS and AIDS solutions since it consists of a set of rules reflecting diverse attacks that are manually collected by experts in the field. The system also ingests audit data, which is abstracted and translated into legible entries for itself by means of semantic attachment. In addition, the expert system encodes

records of past cybersecurity incidents, acknowledged and assumed vulnerabilities of the informational system, and compliance policies. The inference engine, operating according to a reasoning process, makes use of all the aforementioned data to draw a baseline of expected behaviour and further identify deviations from it within current observation in order to activate alarms.

(c) *Description Languages*

By engaging description languages in an AIDS, a system's typical characterisation can be entailed in a formal language, then utilised to spot variations that could represent signs of an attack. Using a formal language, such as first-order logic or a rule-based language, the description language technique describes the system's typical behaviour. The intrusion detection system uses this description to monitor the system's behaviour and identify any deviations from the typical behaviour indicated by the language.

1.3.3 Data Mining and Machine Learning Driven Detection Methods

Briefly studying the pros and cons of the data mining approach as [14] put them into words, below are the consideration that would encourage the use of data mining AIDSs:

- Amplifying the visibility of genuine attacks: Data mining helps eliminate legitimate activity from intrusion alarm data by being able to distinguish false alarm generators and inappropriate sensor signatures from anomalous activity that exposes authentic attacks.

- Recognition of prolonged and recurring patterns: Regardless of the origin's IP address, if identical activity is exhibited, it thereby surpasses some of the adversary's evasion attempts.

The AIDS classes of this section are outlined as follows:

(a) *Unsupervised Learning*

I. *Clustering*

Clustering is by far the most prevalent unsupervised learning routine (i.e., an ML routine that uses merely unlabelled multidimensional data for the machine training stage). The guiding principle of its presence in IDS development is that, when implemented to monitor the behaviour of an informational system, regular occurrences ought to generate sizeable clusters, while outliers are predisposed to producing negligible-sized clusters. To illustrate this underlying mechanism, suppose a cyber-attack exhibits a behavioural pattern scattered across various security logs. Initially, feature selection must be performed. The system picks the features that a cyber-

security specialist would normally examine to search for indicators of compromise (e.g., uploaded traffic volume or associations with uncommon domains). Next, the feature extraction phase is carried out for the selected properties. The logs are scoured for values, usually according to a set time window and to individual entities (e.g., n minutes of logs for each user or IP), such that to each individual entity is allocated a resulting vector of extracted features. Subsequently, the vectors are fed into a clustering algorithm, which groups the entities corresponding to the vectors. Some of the existing clustering algorithms are K-means, Gaussian mixture modelling, spectral clustering, hierarchical clustering, principal component analysis, and independent component analysis, as listed in [10].

Before the powerful techniques of clustering and classification became widespread, the cybersecurity domain was forced to resort to the very slow procedure of association rule discovery when a data mining solution was desired. Data mining's association rules proved compatible with the depiction of intended behaviours, which are consistent. An association rule by itself is a means of finding relationships between the items that make up a dataset. Pretend there is a dataset of audit log records, each record representing a user's activity session and consisting of user activities (file accesses, system logins, software installations). The goal would be to apply association rule mining to discover relationships among the user activities. Given the association rule "$login\_success \longrightarrow file\_access\_restricted.txt; [0.7, 0.18]$", it is composed of two events (successful login and accessing the file "restricted.txt"), a confidence degree of 0.7 (meaning that 70% of the time a user successfully logs in, they also access restricted.txt), and a support value of 0.18 (suggesting that 18% of the entire recorded activity contains both events). To a security analyst, this rule would indicate that successfully logged-in users who don't access the file in discussion deviate from the norm and should be investigated, or that repeated accessing of the restricted file in the absence of a successful login is suspicious.

K-means, the most popular practice when clustering is in sight, is a distance-oriented procedure that intends to split n objects into a forethought number of k clusters by enclosing these objects into the cluster with the nearest centre. It iteratively calculates the proximity of a new data point to the clusters' means, which are also iteratively recalculated, so values can be re-assigned. This forces the algorithm to convert to a stable clustering configuration, optimising the final outcome. Even though the most recurrent similarity measure throughout the literature is the Euclidean distance, other metrics have also been proposed.

Beehive, the solution proposed by the conference paper [27] and one of the earliest IDSs to face the problem put forth by big data in the context of cybersecurity ana-

lytics for real-world log entries, handed its diligence to a clustering implementation derived from the K-means algorithm. Beehive's desideratum was to detect infected or at-imminent-risk-of-infection hosts—i.e., those contacting C&C (Command and Control) servers, those exfiltrating files, or those carrying malware—and hosts that violate the company's policy regulations. The Beehive system operates on an abundant and heterogeneous quantity of log data, originating from web proxy, DHCP, domain controller, antivirus, and VPN, which is stored in a commercial SIEM. Given the challenges posed by these raw logs (collected in real time within a genuine enterprise environment), like differing timestamps, out-of-order arrivals, and divergent identifiers, the first layer in the Beehive's structure needed to incorporate the parsing, filtering, and normalising of the logs (e.g., bringing them to the same timezone and correlating IP addresses to hosts). Naturally, this layer preceded the feature selection, performed by notorious experts on the normalised material and spread in four groups: destination-based, host-based, policy-based, and traffic-based. Nevertheless, clustering joined the table in the posture of the unsupervised method, which was imperative at that moment due to the lack of ground truth regarding the behaviours caught in the dataset. The governing logic is simple: Considering the clear-cut roles and tasks of employees in an enterprise, determining families of hosts with similar behaviours corresponding to specific job functions should help isolate outliers that exhibit unique patterns. In order to reduce the extracted features' dependency and also lower the dimensionality of the resulted hosts' feature vectors, the authors applied Principal Component Analysis (PCA), projecting the original vectors onto a new set of axes (principal components) chosen to capture a minimum of 95% of the data variance. The adaptation of the K-means algorithm to the demands of the input data nature subsists in the following steps: designating a random host vector as the centre of the cluster and inserting all the vectors into that cluster; taking the vector that is farthest from its cluster's centre and creating another cluster around it by transferring every remaining vector to the cluster with the nearest centre; repeating this perpetuation until all vectors are closer to their respective centres than half of the average distance between the centres. The similarity measure (i.e., the distance between vectors) is given in this case by the L1 distance $\sum_{i=1}^{m} |x_i - y_i|$, where x and y are m-dimensional vectors. The conclusion reached by this project was that Beehive successfully detected 784 security incidents (25% malware, 39% policy violation, 35% unknown software) compared to only 8 detected by existing tools. Limitations include manual analysis for outlier meaning, missing exact detection metrics, and longer processing times for large log data.

The upside to using the clustering method stands out in aspects like:

- Unsupervised process: Security specialists don't need to label the data beforehand.

- Adaptability to different data types: By using appropriate clustering algorithms tailored for a certain data type, the solution is fueled by flexibility and can handle a wide range of complex data types.

- Efficient testing phase: With a small number of test clusters against which to compare the test instances, the evaluation process is fast and computationally efficient.

On the downside, there are:

- Dependency on the chosen algorithm: Many clustering-based algorithms may perform less accurately in recognising abnormalities since they are not all designed with anomaly detection in mind.

- Mandatory assignment: Some clustering methods force every instance to belong to a cluster, which may not be the best option for all data circumstances.

- Anomalies cluster: When anomalies create large clusters among themselves, clustering algorithms may face trouble finding these closely packed abnormalities, which might lead to decreased accuracy.

II. *Neural Network Autoencoder*

An autoencoder is a neural network comprised of an input layer, several encoding layers followed by several decoding layers arranged in a bottleneck design, and an output layer. The network is optimised for input data reconstruction accuracy, meaning that the encoding-decoding process aims to produce as similar an output to the input as possible. Due to the bottleneck design, the autoencoder is expected to deduce relationships between the features during the training phase. In the context of anomaly detection, the autoencoder learns to subsequently encode and decode records of legitimate behaviour with high reconstruction accuracy, while records of anomalies are reconstructed poorly with a greater dissimilarity rate.

An emblematic scholastic example was implemented for the online Kitsune NIDS [20]. Kitsune aspired to overcome the regular deficiencies that IDSs run by ANNs (*Artificial Neural Networks*) were known to encounter, such as the demand for supervised learning (implicitly for labelled datasets and a strong CPU for training) and the obligation to accommodate the trained models in the NIDS of each independent company and provide constant updates as the models are trained with more

data. This NIDS, configured to function in a real-time manner on the local network routers, is made up of the usual packet capture layer, parser, and feature extractor, along with an ensemble of autoencoders to identify anomalies, which makes the NIDS extremely flexible and effortlessly deployable on a wide variety of systems.

(b) *Supervised Learning*

I. *Decision Trees (Random Forest)*

Classification is the supervised learning method that, in security analytics, has the objective of partitioning instances into groups of normal and intrusive, each intrusive one corresponding to one attack. Decision trees mark the most classic stratagem towards classification, predicting the threat category each observed instance belongs to. A decision tree's anatomy is constituted by nodes, branches, and leaves (like any other tree, by the way), with the following functionalities: A node in a decision tree acts as a test attribute used for decision making; the branches denote possible paths that a decision founded on an attribute's value pushes the flow towards; each leaf holds one possible outcome (the class to which the instance belongs). The trees can be built either in a top-down or bottom-up fashion.

Random Forest stipulates an ensemble of decision trees, targeting more robust and faultless results. Multiple random trees are trained with arbitrary portions of the dataset, and arbitrary subsets of features are taken into account at each decision node for increased diversification. Aggregating all the decision trees' individual predictions (e.g., by majority of votes), Random Forest yields its final decision.

Classification in intrusion detection is typically used when sharply distinguishing between diverse types of attacks is desired and is preferred for the fast testing phase. In spite of this, it somewhat depends on accurate labels for various classes, which may not always be feasible.

One instance of a Random Forest model being used for the purpose of intrusion detection is in the paper Random Forest Modelling for Network Intrusion Detection Systems [12], which applied it with the goal of classifying attacks from the NSL-KDD dataset, achieving a 99.67% accuracy on a set of features selected with the Symmetrical Uncertainty measure, showing that Random Forests are viable for classifying the attacks in an IDS. Moreover, the model has a low false alarm rate on the testing dataset.

II. *Naive Bayes*

This classification model enjoys the most appreciation among supervised learning applications for IDSs due to its simplicity and computational efficiency. Both of these conveniences arise from the conditional-independence-between-attributes assumption exerted on the Bayes rule. Conditional probability is employed in determining the probability of an attack occurrence conditioned by the standard informational system functioning.

Acclaimed for its notable power in dealing with incomplete datasets (with missing entries) and the exceptional prevention of data over-fitting, this Bayesian approach still retains a major drawback, namely poor performance when the independence presumption is not respected (datasets with advanced dependencies among attributes).

When it comes to previous uses of Bayesian classifiers in intrusion detection, one should first look at the paper Intrusion Detection using Naive Bayes Classifier with Feature Reduction [22], which evaluates the performance of a Naive Bayes classifier on the NSL-KDD dataset. This work also focuses on feature selection, proving that the model can reach an accuracy in the range of 97-98% with carefully picked features and achieving 98.7% accuracy on the DoS attack class, a good result against a wide-spread threat, justifying the use of this method in an IDS.

III. *Fuzzy Logic Classification*

Fuzzy Logic is a system that represents truth values as a continuous range between absolute truth (1) and completely false (0). This concept allows for statements to be attributed to fuzzy sets, each set representing some degree of uncertainty (probably true, maybe false etc.). The goal of Fuzzy sets in the context of intrusion detection is to create a controller, classifying traffic data by using loose if statements, called fuzzy rules, such as "IF set THEN action", which allow for some real-world variation in contrast to boolean-based decisions. These fuzzy rules are usually defined by experts beforehand. More recently, however, this has been approached as a machine learning problem, in which rules are generated from intrusion detection datasets.

The paper Network Intrusion Detection System using Fuzzy Logic [23] presents an IDS that automatically generates Fuzzy rules from 34 selected features of the KDD-cup99 dataset via single-length item mining, definite rule fitting and filtering. This experiment achieved over 90% average accuracy on the testing dataset, showing arguably low effectiveness of this method.

IV. *Support Vector Machine (SVM)*

SVM provides a binary, discriminative, and linear classification for IDSs. It places data points into a multidimensional space, which is further split by a hyperplane to divide the two classes (intrusion and non-intrusion). Generally, a kernel function is applied to the data points in order to add one more dimension to the dataset, thereby making the creation of a hyperplane possible even in cases where the data points are chaotically distributed across the space. Depending on the choice of kernel function (linear, polynomial, or radial basis), various kinds of hyperplanes can be obtained. Considering the applications of SVM on IDS datasets, which have numerous features with little to no value, feature extraction represents a core part of the training process, with each eliminated feature greatly cutting the computational costs of the algorithm.

The key points of SVMs are their simplicity, generalisation capabilities, and performance on feature-rich datasets. Yet, when it comes to implementing SVMs in intrusion detection, several limitations have to be considered, such as lower performance on datasets with massive amounts of data points or the inability of a single SVM model to distinguish between more than two classes.

A noteworthy implementation of SVMs appears in the paper [17], where gradual feature reduction is showcased via a pipeline of ML models, including SVMs, resulting in a 98.67% accuracy on the NSL-KDD dataset. To play to the strengths of SVMs, the dataset is undersampled into a small and robust dataset, with the SVM being trained on a varying number of features in order to determine their importance.

V. *K-Nearest Neighbours (KNN)*

This non-parametric classifier imposes the following course of action: The training stage receives labelled dataset vectors (normal and paranormal events), and a metric to compute the distances between these vectors must also be chosen. Then, the testing stage brings to front an unlabelled data point (a vector) and calculates the distance between it and the training vectors, relative to the metric chosen, to ascertain its nearest k ones, where k has a prefixed value. The majoritarian label among these k labels becomes also the label of the unlabelled data point. In a colloquial way of speaking, the k-nearest neighbours of the unlabelled vector "vote" for its label.

Since implementing KNN is a simple thing to do and the classifier has an adaptive character often admired for its performance in IDSs and seamless adaptability to parallel implementations, KNN can be viewed as a benchmark in the field. De-

spite these strengths, KNN is also known for its substantial storage demands and susceptibility to the problem of dimensionality, which frequently leads to data test classification slowness.

Applied to the NSL-KDD dataset in [3], KNN demonstrated a faster build time in comparison to other classifiers on the same dataset (only 0.03 seconds to build the model on the network training data). The detection rate was lower than desired, at only cca. 94.56%.

VI. *Neural Networks (NN) and Deep Learning*

NN, the broadly spread ML procedure structured to resemble the human brain (neurons), fructifies an adaptive schema according to which computers continually develop by learning from their past failures. The IDES (intrusion detection expert system) [8] was the first to use neural networks in intrusion detection as an alternative to statistical approaches. A pure feedforward neural network (i.e., where data is processed in one direction: input nodes $\longrightarrow$ output nodes) with backpropagation (i.e., backward adjustment of the model's layer parameters/ weights dictated by the error obtained for a training data point) that has been properly trained performs on par with a straightforward signature-based system.

NNs' formidable capacities at detecting all sorts of malware stand out in the literature. So does their ability to capture convoluted connections between input data and classification labels through the resulting nonlinear models, especially in the case of deep learning (multiple hidden layers). Needless to say, neural networks come with their own limitations. The most remarkable one may be their reduced detection accuracy and precision with fewer intrusions due to their sensibility to noise.

An abstract example of anomaly detection where neural networks are recommended is within the scope of UEBA. Audit log sets can be collected daily from each user with a view to training an NN. A data point vector would thereby encapsulate the distributions of the commands executed by one user in a day, and the NN would be trained to determine the user corresponding to each vector. Thus, in this scenario, the testing phase will label as anomalous those new data points that are classified as corresponding to users other than the one for whom the logs were truthfully obtained.

Now, let us invoke DeepLog [11], an AIDS centred around a LSTM (*long short-term memory*) recurrent neural network (feedback loops connecting one layer with itself in order to maintain short-term memory about the state of said layer for a previous data point). As LSTM is designed to operate with sequential data (e.g., text and time

series) and log entries can be perceived as sequential over time, this deep learning model construes the log entries as a series of patterns resembling natural (structured) language. DeepLog is first trained on logs describing normal behaviour to then be capable of labelling anomalies. Due to the architecture of the NN, DeepLog is able to process logs from a real-time stream (instead of batch processing them), allowing a security expert to act on a potentially malign situation as it unfolds. In an ideal scenario, an attack would be flagged and repelled with no delay between its start and its detection. DeepLog shows great performance when tested on the Openstack I and Openstack II logs, with an F-measure of 97% and 98% respectively. On top of the performance metrics, DeepLog also comes with harder-to-quantify advantages such as ease of deployment and the capability to pin-point anomalies at log level in real time.

Chapter 2

Dense Neural Networks Ensemble for the UNR-IDD Dataset

In light of my own practical experimentation with delineating an intrusion detection mechanism, I opted to fulfil the objective of exploring big data-driven anomaly-based IDSs through the creation of two distinct prototypes via supervised learning techniques. One of these prototypes is presented in this chapter, while the other will be discussed in Chapter 3.

It is relevant to stay mindful of the fact that neither one of these machines I trained and tested is a fully-fledged AIDS per se, but rather a demo programme whose sole purpose is dedicated to exploration of the cybersecurity concept that this paper is promoting. As a result, although they may contribute to the existing literature in the field (a modest portion of which was recalled in the previous chapter), they do not provide any upstanding technology that a real-world enterprise should use to protect itself from cybercrimes.

Having said that, Chapter 2 is devoted to illustrating an ensemble of dense, feedforward neural networks applied to the UNR-IDD dataset, as one is about to discover within the following sections.

2.1 The UNR-IDD Dataset

Compared to related research papers that are studying various implementations of intrusion detection approaches over illustrious but old, if not even obsolete, datasets (like NSL-KDD, an improved extension of 1999's KDD), I decided to work with the recently released UNR-IDD dataset [7]. The University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD) was published in 2023, and it prominently highlights its ambition to innovate in the area of open-sourced datasets that are consequently gaining traction for ML-based NIDSs by pro-

viding adequate modelling of tail classes. This NIDS dataset also pushes against the tendency of traditional datasets to primarily aggregate network flow-level information (i.e., source and destination IP addresses, number of transferred packets or bytes, port numbers, protocol types, and the flow duration). Although useful when it comes to analysing large volumes of network traffic, classical datasets are prone to overlooking fine-grained details of individual packets and, nevertheless, are deeply dependent on the initial network configuration, lacking the capability of the obtained NIDS to be transferred to different configurations.

The proclaimed strength of the UNR-IDD dataset, as explained in [7], is brought about by its detachment from the flow observation trend and the adoption of network port statistics, meaning that its data is compounded of recorded port metrics and delta port statistics to give a sense of magnitude deviations in time. The desired effect of this method is prompt irregularities identification. Furthermore, great attention is given to better representation of the entire spectrum of classes, including the tail ones, as a generous number of complete samples are attributed to each class, so an NIDS can produce high F-scores (the harmonic means between precisions and recalls).

Placing both sides of the scale, namely customary datasets and UNR-IDD, in a fair comparative light, one should also portray the latter one as relatively small, a trait that may appear convenient when looking over the training periods but inconvenient when considering the overall training outcome. As an illustration, the research's favourite dataset, NSL-KDD, puts together over 125K data entries; in the meantime, UNR-IDD only possesses around 37K. Even so, as the consensus reached by the dataset's documentation [7] and the current study's result pronounce, the aforesaid drawback is compensated by the proportions that each network traffic category occupies in the dataset. Expressing it in a manner that is more direct and clear, for agreeable intrusion detection results, one can afford to avoid flooding a dataset with records and data points as long as they guarantee adequate representation of each class regarded. Finally, yet importantly, while UNR-IDD achieves its demonstrative ideal, the real-world applicability of a ML-supervised model trained on this dataset would require a multitude of additional attack classes.

2.1.1 Data Collection

An authentically simulated and controlled environment was configured by the University of Nevada by means of an SDN (*Software-Defined Networking*) controller along with a network topology generator, Mininet. The resultant topology embodies 10 hosts and 12 switches.

A Python script employing IPerf and the Mininet API was then written to simulate lifelike network flows with the instrumentality of TCP-UDP data streams and dummy payloads. Equal importance was invested in each and every traffic category (the legitimate flows and the

anomalous traffic types) by making it certain that all of the considered scenarios are simulated in a relevant number of flows of the same 5-second duration so that minimum variation and sufficient representation are enrolled. The source-destination host pair was always selected at random.

A custom programme uses messages sent between the controller and switches to regularly (once every 5 seconds) collect statistics from switches, compute delta port statistics, store the information in a map, and record it in JSON files, according to [7].

2.1.2 Data Features

Every single switch inside the SDN was being monitored, and statistics (metrics and magnitudes) were collected at the port level as the network flows were being simulated at the set time interval of 5 seconds. In consequence, a data point within the dataset reflects a log entry corresponding to one port of one SDN switch at a moment in time. In addition, the delta values were computed as a difference between a port's collected metrics after each time interval. Ultimately, this dataset also presents statistics related to the simulated network's flow table.

The captured data features can be grouped into three families, as per [7], with the mention that "Rx" refers to "receiver" and "Tx" to "transmitter".

1. *Port Statistics:* Received Packet, Received Bytes, Sent Packets, Sent Bytes, Port alive Duration, Packets Rx Dropped, Packet Tx Dropped, Packet Rx Errors, Packet Tx Errors.

2. *Delta Port Statistics:* Delta Received Packet, Delta Received Bytes, Delta Sent Packets, Delta Sent Bytes, Delta Port alive Duration, Delta Packets Rx Dropped, Delta Packet Tx Dropped, Delta Packet Rx Errors, Delta Packet Tx Errors.

3. *Flow Entry and Flow Table Statistics:* Connection Point, Total Load/Rate, Total Load-/Latest, Unknown Load/Rate, Unknown Load/Latest, Time seen, TableID, Active-FlowEntries, PacketsLookedUp, PacketsMatched, MaxSize.

2.1.3 Labels

The UNR-IDD dataset creators adopted two sets of labels for the data points encapsulated in their collection in order for a ML solution to be trainable for both binary and multi-label classification.

(a) *Binary Classification*

Binary classification seeks to distinguish between intrusions and typical behavioural circumstances. It evaluates whether the surveilled network is experiencing an attack or not

without providing any details on the attack's properties by simply labelling the data point as either "Normal" or "Attack".

(b) *Multi-label Classification*

Multi-label classification, on the other hand, has in view a vaster assessment of the encountered behaviour by envisaging an entire set of attacks instead of the unilateral "Attack" branch. The labels it enfolds besides the elementary "Normal" one are listed below:

- "TCP-SYN" - It refers to the TCP-SYN Flood, a subcategory of DDoS (Distributed-Denial-of-Service) attacks, i.e., a movement orchestrated to overwhelm a target network by rendering it unavailable to authorised users. When conducting a TCP-SYN Flood attack, malicious actors target particular hosts by launching an excessive number of TCP handshakes. In contrast to a typical TCP handshake, which entails a three-step procedure to guarantee appropriate connection formation, in this attack, the attackers send the initial SYN (synchronise) request without waiting for a response from the target node. On the target device, this causes a buildup of open but pending connections, eating up precious system resources. The target's resources are reduced as a result of the battle to handle the inflow of incomplete connections, which can cause performance degradation or even a full shutdown. The TCP-SYN Flood attack makes use of the TCP protocol's innate susceptibility to resource exhaustion, making it a potent weapon for crashing websites and online services.

- "PortScan" - Port scanning usually acts as a prelude to more targeted malicious activity, as it empowers cybercriminals on the road to sharpen their tactics and increase their success rates. It is basically a reconnaissance technique that attackers apply to acquire information about the network services, versions, and configuration (sometimes even security protections, like firewalls and IDSs) of a target host device. Throughout this procedure, several ports on the target host are regularly scanned in an effort to connect or search for open ports; the exposed ones supply valuable intelligence about the target and make it possible for attackers to further craft attacks that take advantage of the identified vulnerabilities.

- "Overflow" - This one annotates the flow table overflow, a strategic assault on a network's switches and routers that undermines their operational integrity. By exploiting known vulnerabilities, adversaries become apt to compromise the vital network elements' operability. This happens when these devices' flow tables are used to their fullest extent, as they are the ones that administer the correct guiding and forwarding of the network packets. Attackers saturate the flow tables with a bombardment of fraudulent flow entries and rules, which causes a bottleneck in the handling of

real network data. The incident prevents legitimate flow entries and rules from being set up correctly, which prevents the device from processing and routing traffic as intended. As a result, the overtaken switch or router suffers a devastating loss of performance, which causes network outages or, in severe cases, the inaccessibility of the entire service. By using the device's built-in defence mechanisms against itself, this assault takes advantage of the design flaws to create chaos and intoxicate the device's core operation.

- "Blackhole" - Just like the previous attack, the blackhole directly affects switches and routers in a network, intending to interrupt the regular flow of its traffic. The adversary misleads the aforementioned components into discarding the arriving packets that would normally be transmitted to the next predetermined destination. A "blackhole" (void) is consequently conjured inside the network path due to the switches and routers intercepting but ignoring the network traffic and refusing to forward it. This communication flow interruption disrupts the expected packet movement, which, in the best-case scenario, results only in delays but has the potential to culminate in data loss or critical flow breakdowns over whole network segments, sabotaging the system's reliability and integrity.

- "Diversion" - Also directed at the network's switches and routers for flow manipulation aspirations, the traffic diversion attack hopes to reroute packets away from their intended destination. The packets' journey duration is thereby extended, leaving space for subtle Man-in-the-Middle interventions to capture network communications. While the packets traverse altered routes, the traffic may be both delayed (the mild possible consequence) and disrupted towards a circuit that upholds the position of the attacker (somewhere between the source and the destination) who intercepts, eavesdrops, or even alters the transmitted content (the detrimental possible consequence), endangering the privacy and the integrity of the network system.

2.2 Deep Learning on UNR-IDD

This section covers my personal interaction with the dataset. The process involves meticulously going over the features of the dataset, electing the ones that have the greatest significance, and getting the data ready for analysis. I travel through the steps taken to prepare and process the data for future machine learning experiments by cleaning, normalising, and transforming it. The goal of these experiments is to recognise patterns, abnormalities, and forecasting trends in an effort to produce actionable insights for efficient intrusion detection.

2.2.1 Feature Study and Data Preparation

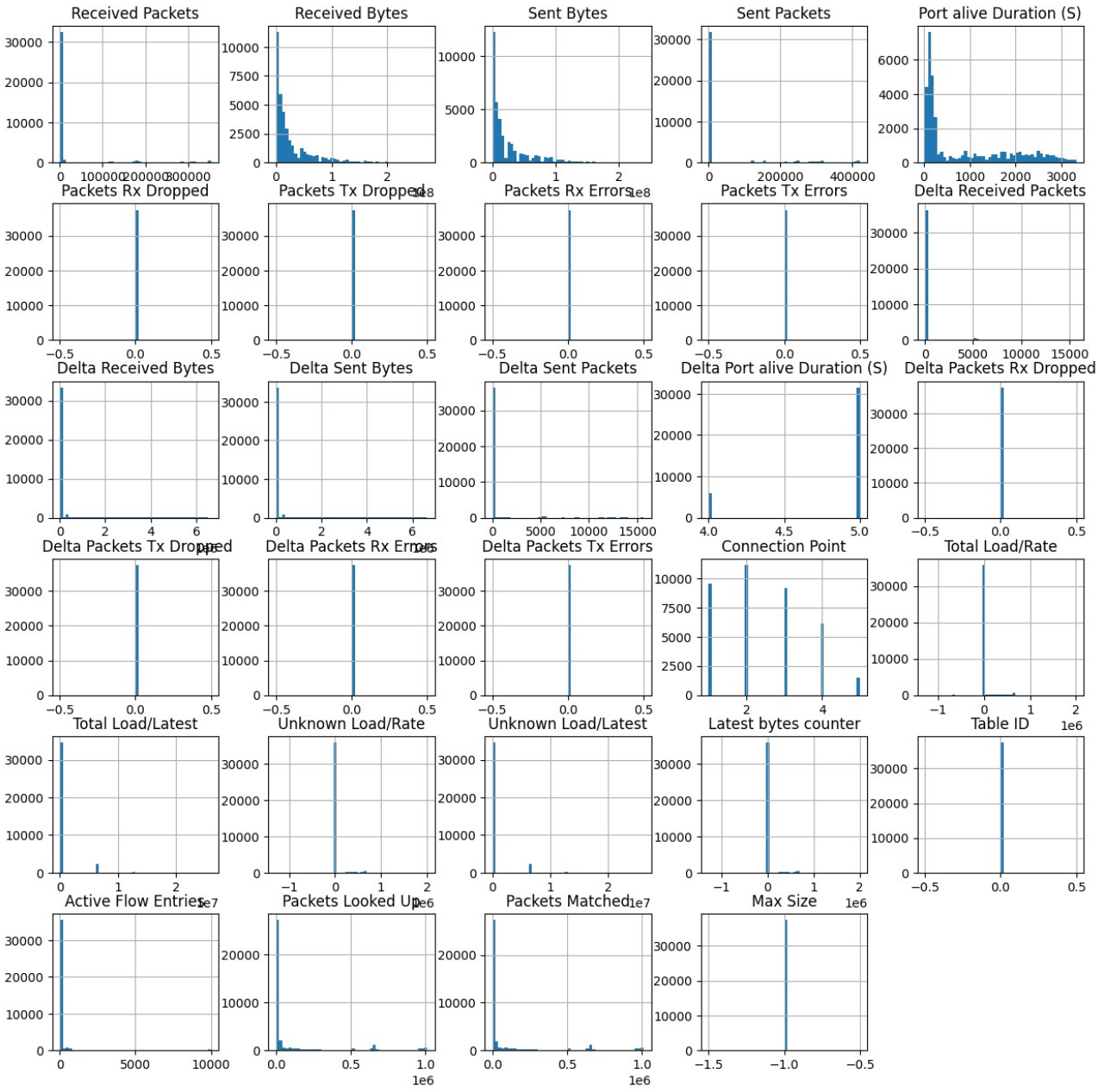

Fig. 2.1: Histogram Analysis of Features in the UNR-IDD Dataset

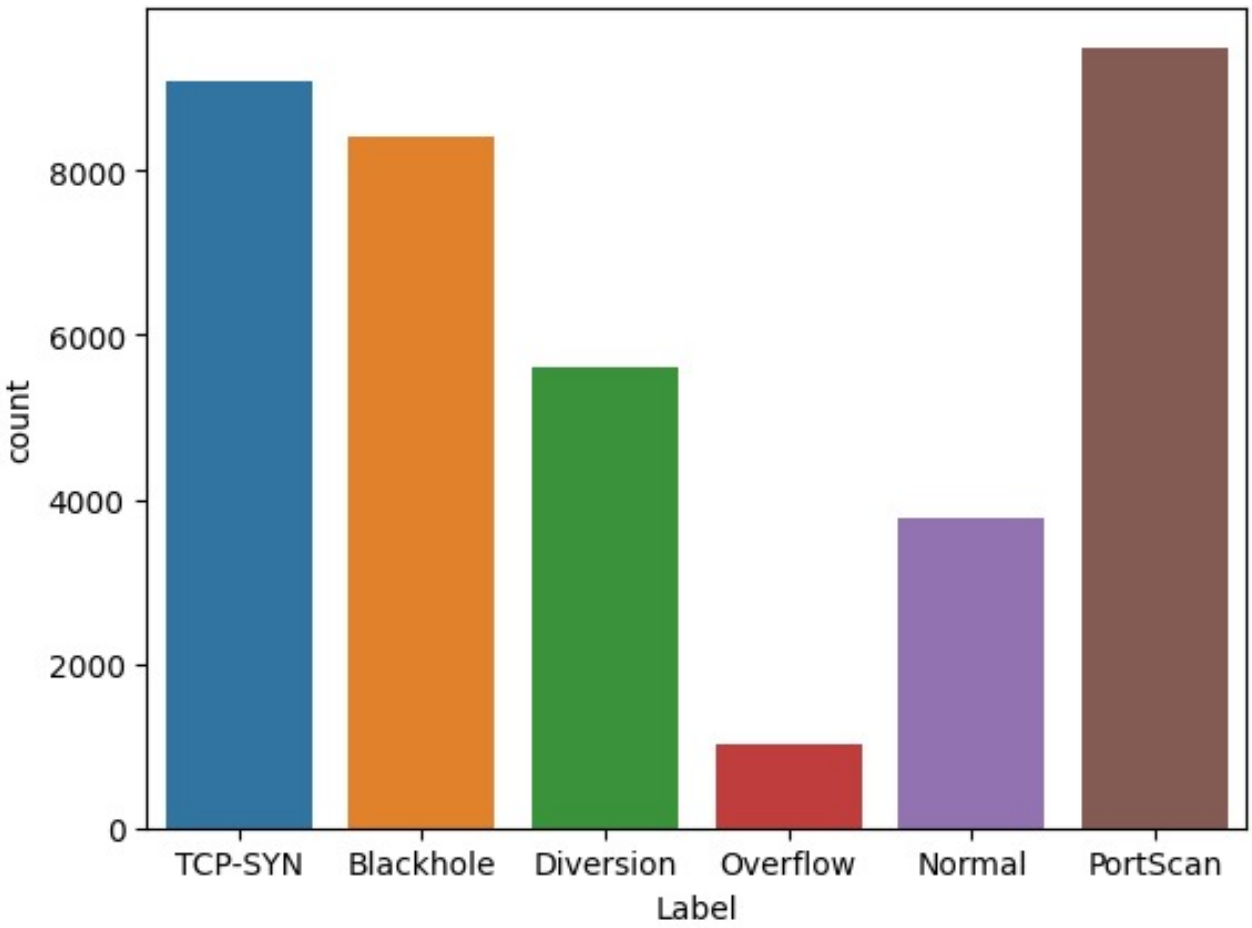

Fig. 2.2: Distribution of Attack Types in the UNR-IDD Dataset

Figure 2.1 displays the histogram of each feature the dataset was conceived with and helps determine their degree of relevance. By consulting these histograms, I ascertained the list of data features having a standard deviation of zero and thereafter eliminated them from the schema. These redundant features are: Packets Rx Dropped, Packets Tx Dropped, Packets Rx Errors, Packets Tx Errors, Delta Packets Rx Dropped, Delta Packets Tx Dropped, Delta Packets Rx Errors, Delta Packets Tx Errors, Table ID, Max Size.

To assess the relevance of the remained features, a variant of the permutation feature importance technique was used. After training the model that will be introduced in the upcoming section, the testing data was fed into the model, calculating its accuracy. Afterwards, one by one, each of the features in the testing data was set to zero, with the resulting accuracy being subtracted from the original one. These values can be seen in Figure 2.3. Whether this algorithm fully quantifies the relevance of each feature or not is still up for debate. It is, however, a good method of proving that only relevant features were selected.

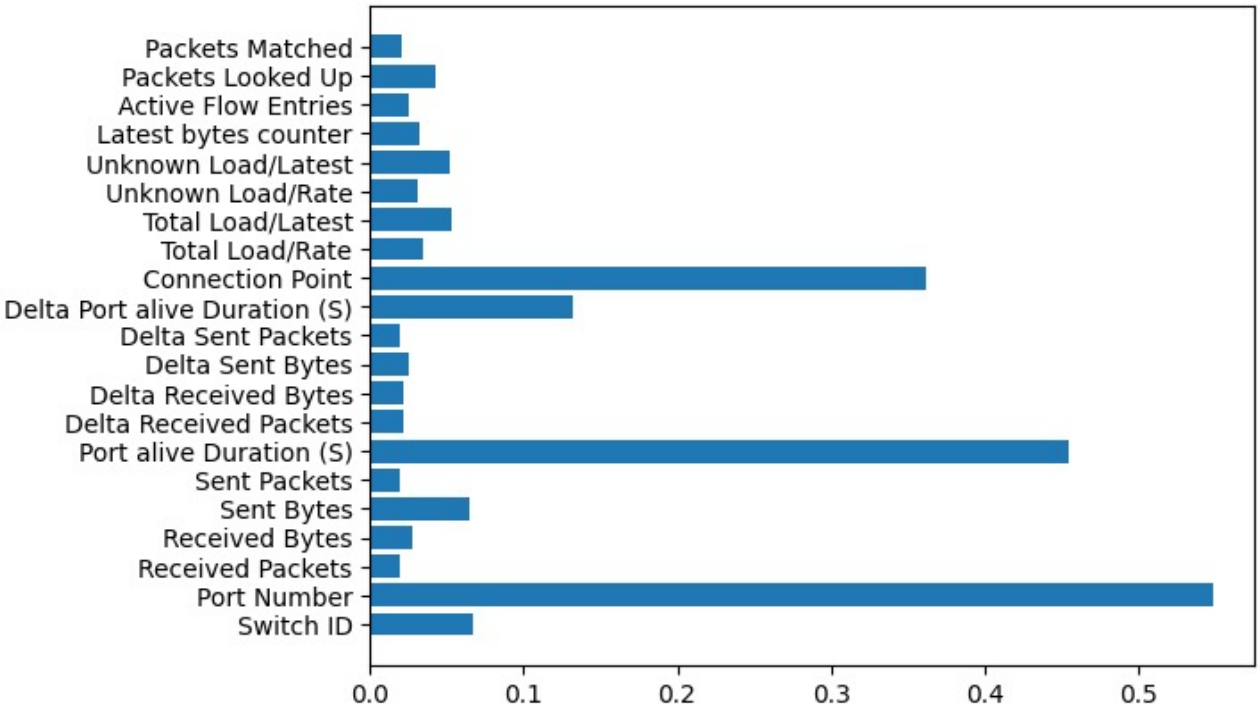

Fig. 2.3: Relevance of the Selected Features (UNR-IDD)

With regards to the aspect of data preparation, two focal transformations were executed:

- The Switch ID feature, originally stored in OpenFlow format within the dataset (as a hexadecimal representation of switches' MAC addresses that the SDN environment manages, e.g., "of:000000000000000c"), was converted to a decimal number ranging from 1 to 12.
- The Port Number feature, which initially appeared in the format "Port#:i", $i = \overline{1,4}$, was first converted to the numerical value i and later, during further experimentation, delegated as the object of One-Hot Encoding, implying that it was transformed into 4 binary columns: Port$i$, $i = \overline{1,4}$ (for one data point, the column corresponding to the delegated port contained 1, while the others were filled with 0). An improvement was observed in the overall accuracy after one-hot encoding compared to the previous simple processing. This improvement can be justified by the neural networks disregarding the numerical order of the switch ports.

Heading to data normalisation, first and foremost, it must be observed that data is of two sorts: numerical and categorical. The categorical values are those in the Connection Point and Switch ID columns as well as the one-hot encoded Port Number. Each of the neural networks in the ensemble that remains to be described contains a normalisation layer that shifts and scales numerical inputs into a distribution centred around 0 with a standard deviation of 1.

2.2.2 Model Architecture

For the dataset under consideration, I propose an ensemble model composed of three neural networks and a Random Forest classifier. The ambition of this exercise is to successfully classify the testing data points into one of the six labels enumerated in the section before, as well as to evaluate the possibility of detecting previously unseen attacks by excluding one of the anomaly classes from the training set in order to emulate a zero-day attack.

1. *Neural Network - Model 1*

   This DNN (*Dense Neural Networks*) consists of an output layer of 6 neurons (one for each class) with the Softmax activation function, an input layer of 18 neurons (one for each numerical input feature), feeding into a normalisation layer, accompanied by a concatenation layer which combines the normalised numerical features with the 6 categorical inputs, and a densely connected hidden structure as follows: Layer 1 (512 neurons) - 40% Dropout - Layer 2 (256 neurons) - 30% Dropout - Layer 3 (128 neurons) - 20% Dropout - Layer 4 (64 neurons), with each neuron being activated by the ReLU (*Rectified Linear Unit*) function. This network is optimised by the Adam optimiser, with a learning rate of 0.001. The model's loss is evaluated by the categorical cross-entropy function.

2. *Neural Network - Model 2*

   This network is constructed with 18 neurons and 6 neurons in the two input layers, separating the numerical and categorical features. The numerical features are passed through a normalisation layer and then concatenated with the categorical ones. This network's output layer is made up of 6 neurons, activated by the Softmax function. The difference between this model and the previous one resides mainly in the hidden architecture, which now has the following configuration: Layer 1 (256 neurons) - 40% Dropout - Layer 2 (128 neurons) - 20% Dropout - Layer 3 (64 neurons), with the activation function ReLU. The categorical cross-entropy function is used to calculate the model's loss, with the Adam optimiser adjusting the network's parameters during training. The chosen learning rate is 0.0001.

3. *Neural Network - Model 3*

   This last network has the same input, normalisation, concatenation, and output layer structures as the previous ones, with a hidden structure consisting of: Layer 1 (64 neurons), 20% Dropout - Layer 2 (64 neurons) - Layer 3 (64 neurons) - Layer 4 (64 neurons) - Layer 5 (64 neurons) - 10% Dropout - Layer 6 (64 neurons), all having the ReLU acti-

vation. The chosen loss function is categorical cross-entropy, and the chosen optimiser is Adam, with a learning rate parameter of 0.0001.

4. *Random Forest Classifier*

   To complete and also diversify the ensemble, a different type of classifier was picked, none other than Random Forest. It was designed to have 250 decision trees, and it ensures that the resolution is not drawn solely by neural networks.

Immersing into the actual training and testing of the four models brought up above, UNR-IDD was split by this structure: 80% training data, 10% validation data, and 10% testing data in the case of the three neural networks; the validation data was appended to the training data in the case of Random Forest, producing a data distribution of 90% training with 10% testing.

For all these DNNs, the batch size was chosen to be 32, but the number of epochs differs from one model to another: the first neural network was set up with 180 epochs, the second neural network was configured to go through 300 epochs, and the third one was implemented with 250 epochs. The reason for the proportionally small number of epochs in the first DNN is given by the higher learning rate that this one was designed to have.

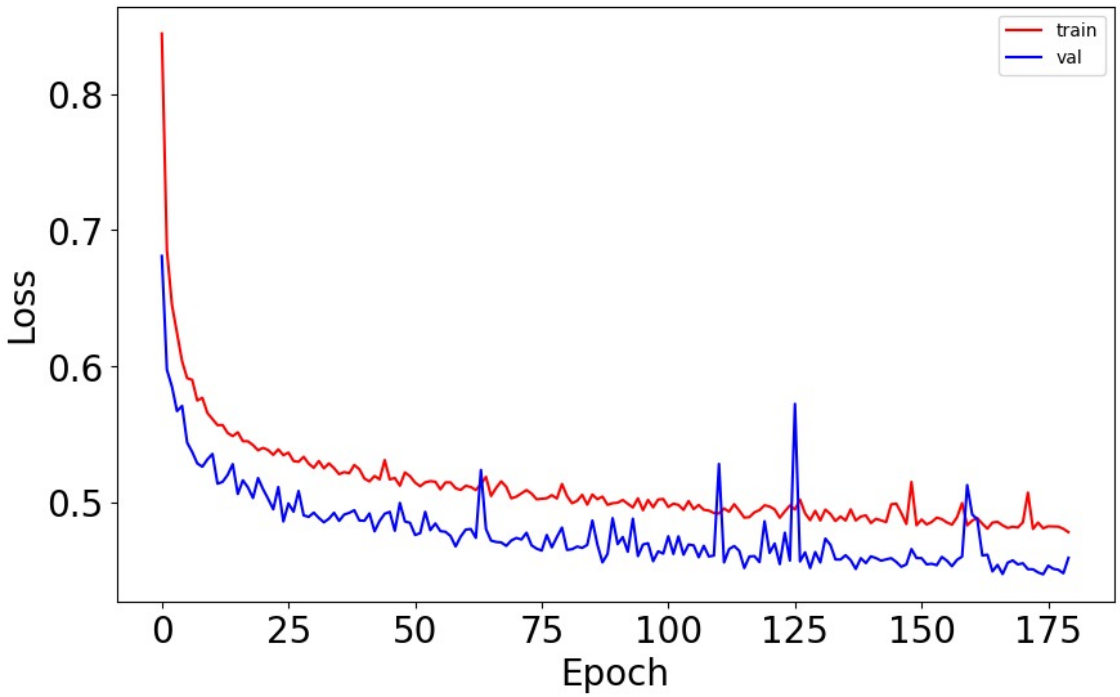

Fig. 2.4: Loss Function Graph for Neural Network - Model 1 (UNR-IDD)

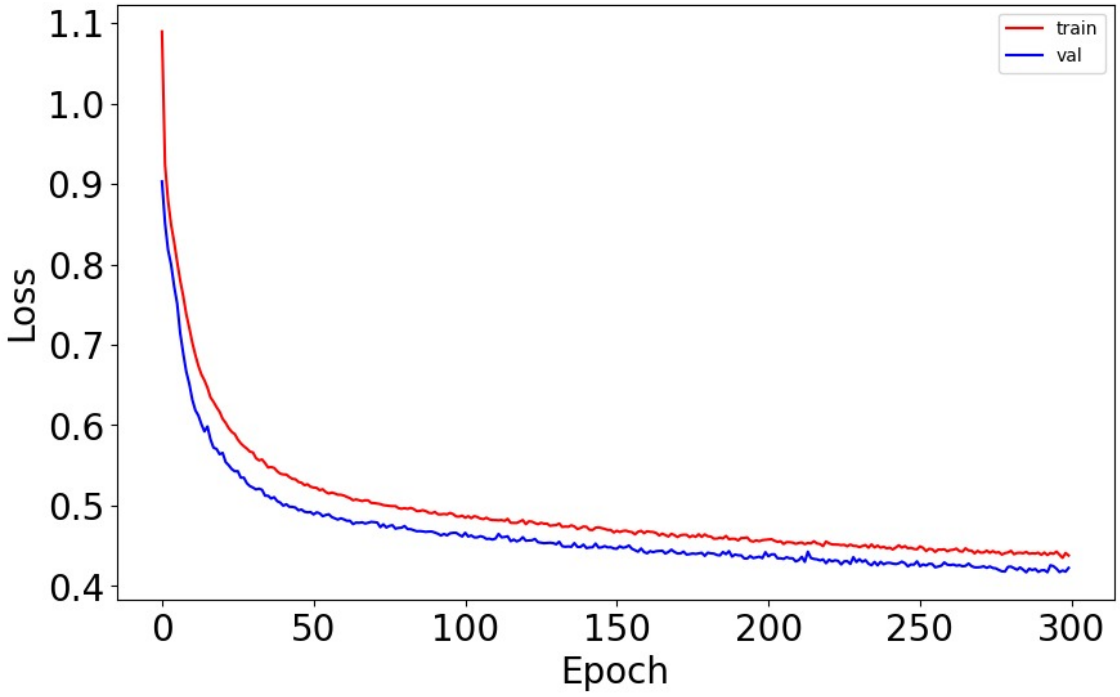

Fig. 2.5: Loss Function Graph for Neural Network - Model 2 (UNR-IDD)

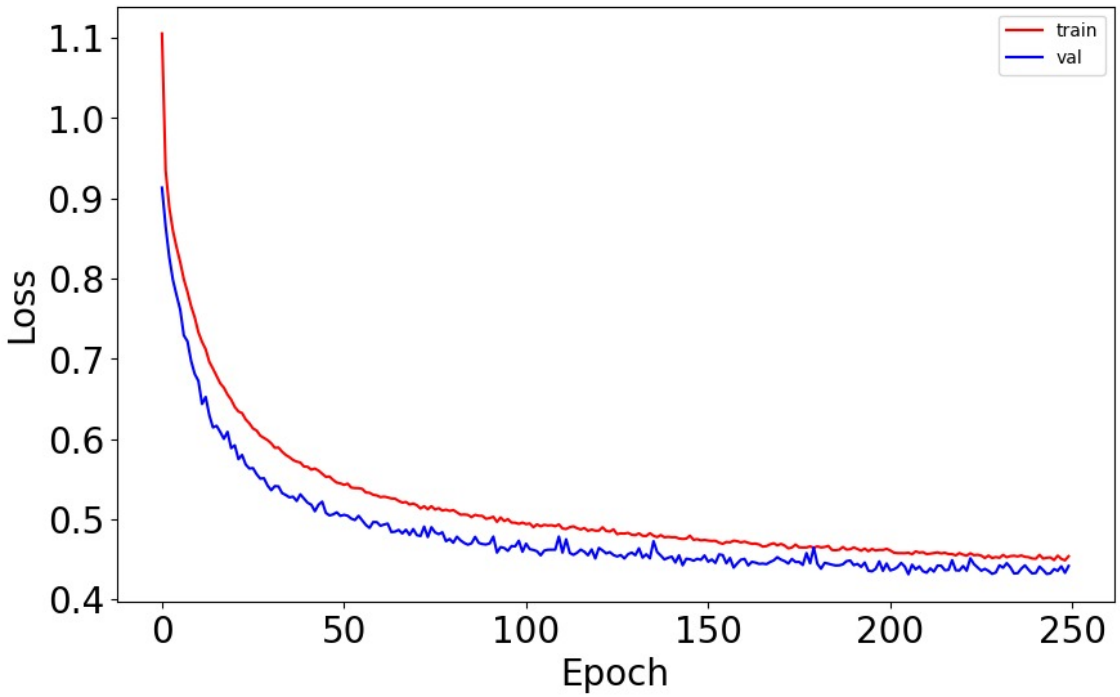

Fig. 2.6: Loss Function Graph for Neural Network - Model 3 (UNR-IDD)

The training process of each neural network is shown by plotting the categorical cross-entropy function on the training and validation sets in each epoch. These graphs (Figures 2.4, 2.5, and 2.6) serve as an indicator for neural network overfitting, with the training loss dropping below the validation loss if the model starts memorising the training data without learning generalised information, therefore having a worse performance when presented with new data.

2.2.3 Results and Conclusions

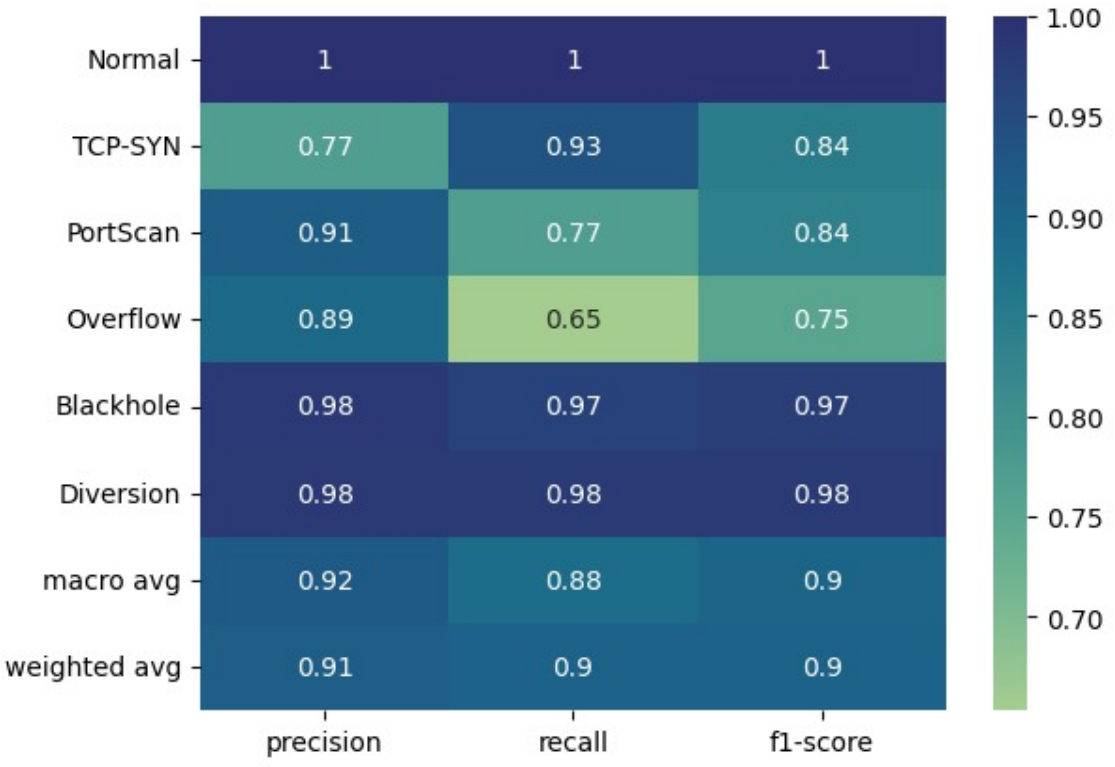

Fig. 2.7: Classification Report for Neural Network - Model 1 (UNR-IDD)

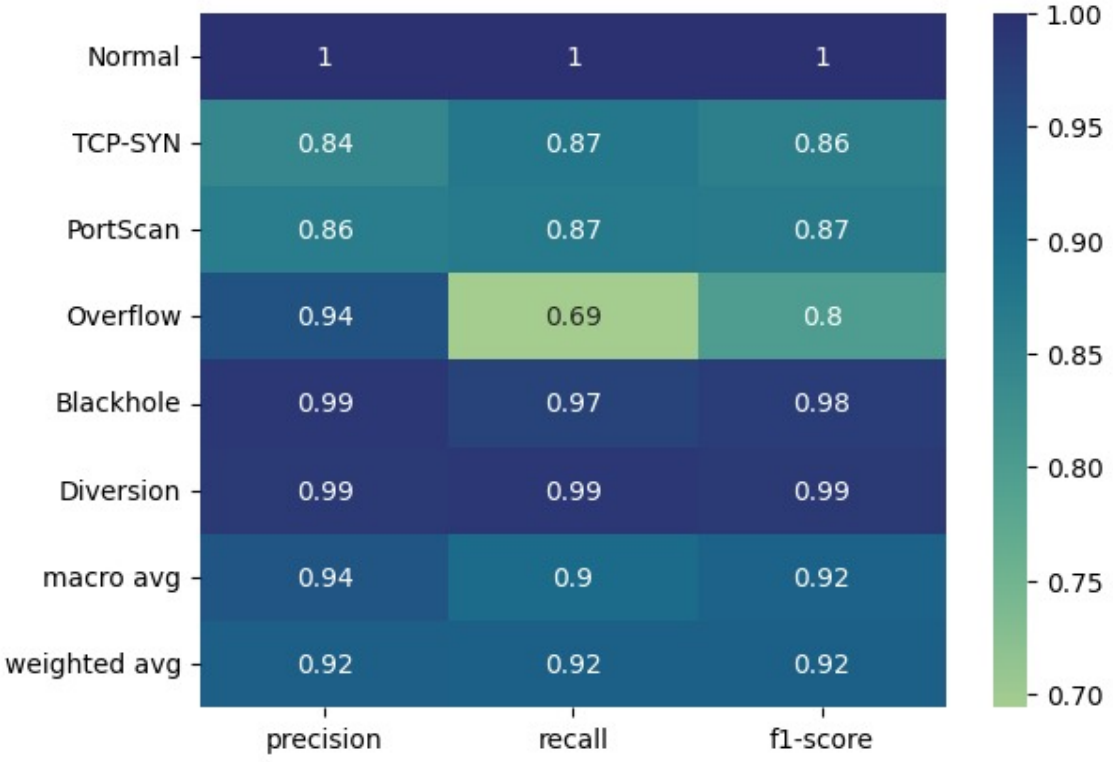

Fig. 2.8: Classification Report for Neural Network - Model 2 (UNR-IDD)

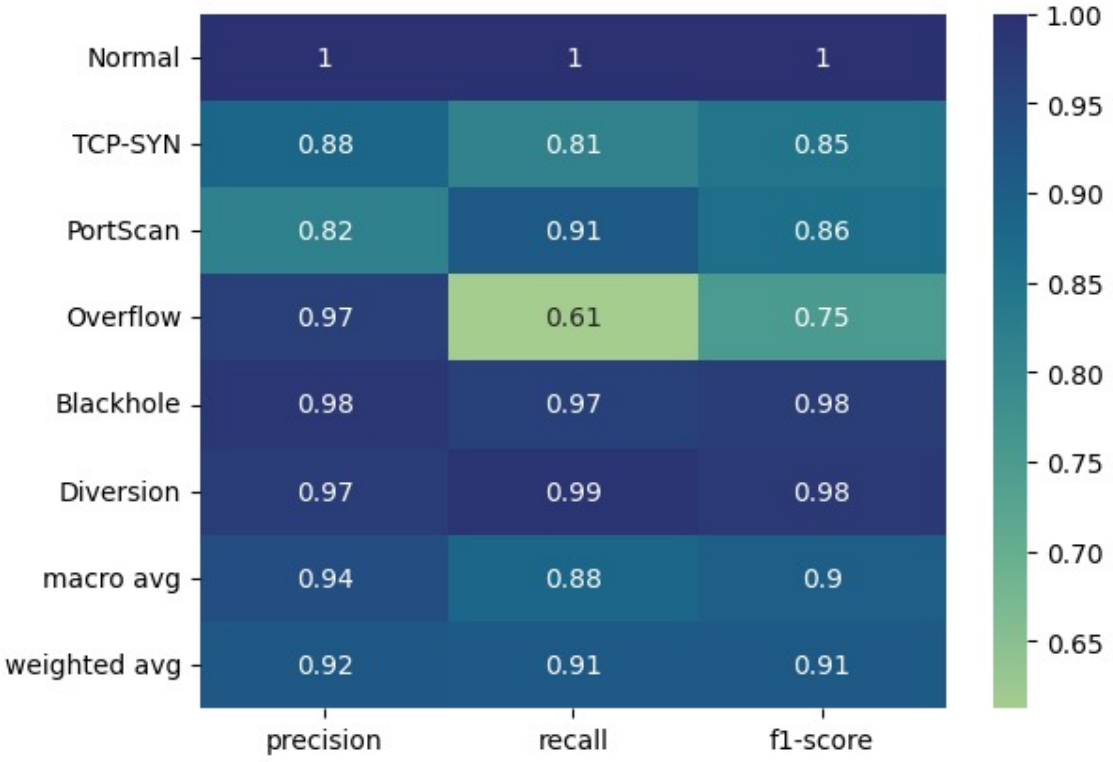

Fig. 2.9: Classification Report for Neural Network - Model 3 (UNR-IDD)

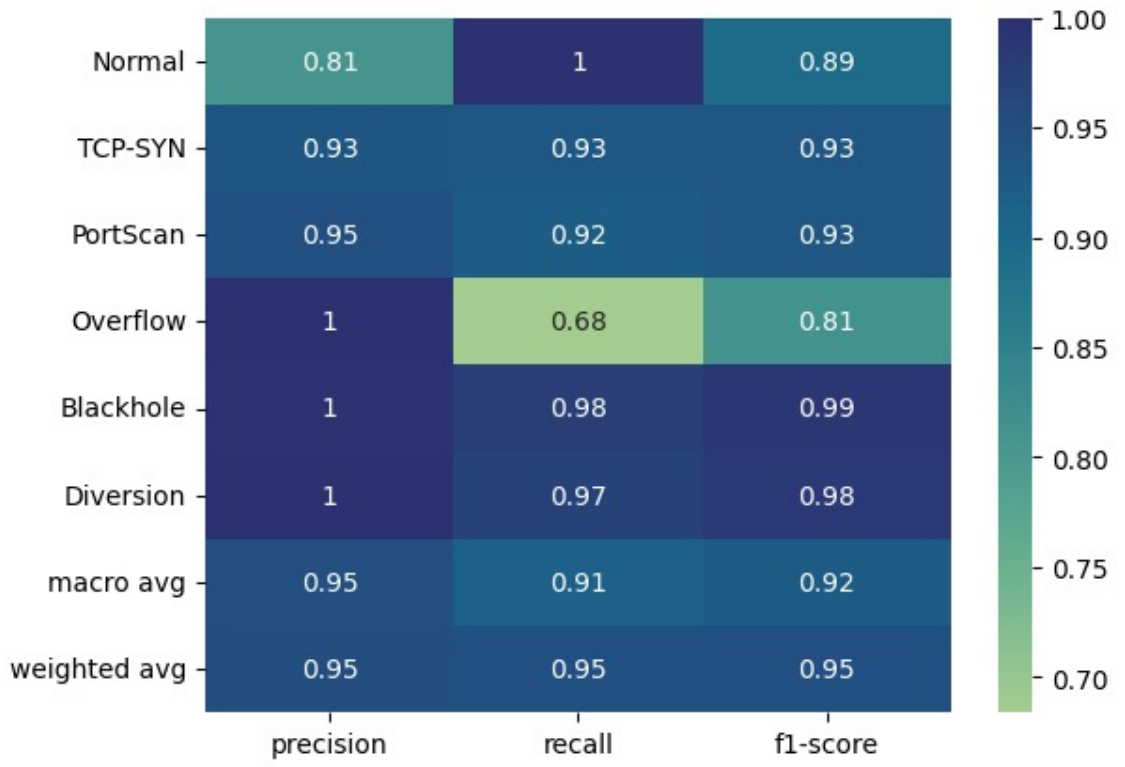

Fig. 2.10: Classification Report for Random Forest (UNR-IDD)

When it comes to the performance of each model on the testing data, the reports above illustrate how the neural networks and the Random Forest classifier perform on each of the six classes. One conclusion that may be drawn from the individual reports is that the third DNN model, which has the highest number of hidden layers, performs better on the Overflow attack class, while the wider models (those with fewer but larger layers) derive a stricter definition of the PortScan class with higher precision. The most noteworthy difference is in their performance on the normal traffic class, with the neural networks showing perfect results on the training set while the Random Forest model obtains around 80% precision. For practical purposes, any confusion related to the normal traffic class makes the model undesirable due to the costly consequences generated by these misclassifications. However, since the Random Forest model displays better results than the DNNs across the attack classes, it cannot be dismissed.

With the above results in mind, an ensemble voting method was implemented in an attempt to improve each model's classification capabilities. The chosen voting weights were 0.1 for the first DNN, 0.3 for the second DNN, 0.2 for the third DNN, and 0.4 for Random Forest. These values, reached after trial and error testing, create an ensemble that perfectly classifies normal traffic while showing equal or better results than each individual model on the attack classes, thus resulting in a 96% precision, 96% recall, and 96% F-score on the testing data. While better neural architectures and more precise voting weights may be obtained by making use of genetic algorithms, thereby obtaining better results, the computational costs are hardly

justified.

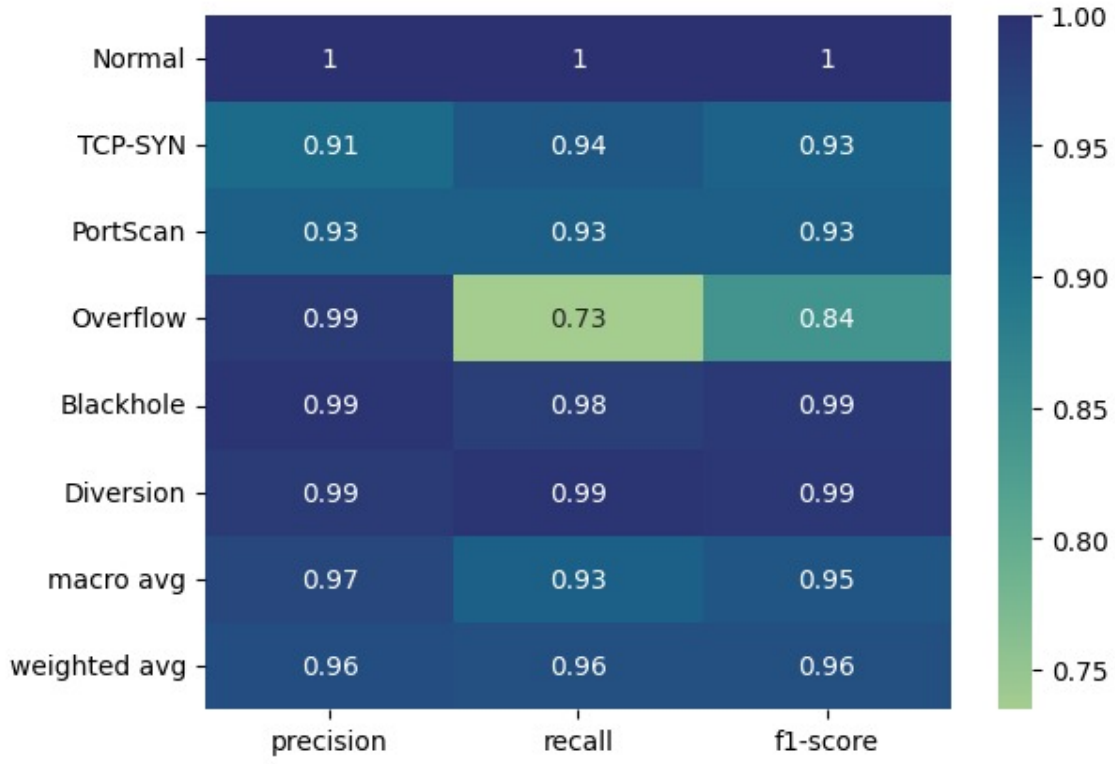

Fig. 2.11: Classification Report for the Ensemble (UNR-IDD)

This experiment also explores the possibility of detecting zero-day attacks in a simulated scenario by taking advantage of the binary labelling that UNR-IDD was endowed with. This classification is done only with a modified version of the second neural network: the 6-neuron layer is now activated by the ReLU function, becoming a hidden layer, and an extra layer with one neuron activated by the sigmoid function is added as the new output representing the probability of the data point being an attack. The loss function used is the binary cross-entropy function. The training dataset was modified to remove all examples of the Blackhole attack, resulting in the neural network seeing the aforementioned attack during the testing phase for the first time. After only 20 training epochs, the model is able to perfectly classify the testing data into the two classes, normal and attack.

While no effective method of labelling an unknown attack as "new" was found, the network shows a clear understanding of normal traffic, removing the dangerous possibility of a zero-day attack being classified as normal traffic.

Chapter 3

Comparison Between SIDS and AIDS on the CIC-IDS2017 Dataset

This chapter directs its attention towards actually contrasting the outcomes of a trivial instantiation of AIDS, which I developed, with those of an SIDS. For the scope of this study, I have selected Snort [2] as the representative SIDS due to its pre-existing repository of signatures. Since both approaches had to be tested on the same dataset, I needed to select one that would offer PCAP (*Packet Capture*) files to be supported by Snort, and this is when I stumbled upon CIC-IDS2017 [25].

3.1 The CIC-IDS2017 Dataset

With the same goal as the University of Nevada in mind of addressing the issues posed by outdated and unreliable datasets that have been dominating the computer and network security field since 1998, the Canadian Institute for Cybersecurity developed a dataset that confronts the familiar insufficiencies: the lack of traffic diversity and volumes, the undersupply of attack variants, features, and metadata, and the elusive packet information and payload. CIC-IDS2017 is designed to bring an innovative outlook to the properties and criteria that should be met by a modern dataset in order to sustain contemporary intrusion detection necessities. These properties were formulated in [13] as an epilogue to an extensive research of the studies conducted until 2016 and they total up to eleven, namely: attack diversity (variety of attack types), anonymity (sensitive data modelling for privacy protection), available protocols (broad spectrum of communication protocols), complete capture (integral network activity recording), complete interaction (integral exchange between entities recording), complete network configuration (ability to mimic diverse real-world network topologies), complete traffic (record of all layers in the network traffic), feature set (large set of extracted features), heterogeneity (in terms of software, devices, operating systems etc.), labelling (accurate and detailed

labels attached to the instances), and metadata (for transparency and reproductibility).

3.1.1 Data Collection

The methodology that the authors appealed to when conceiving the testbed architecture lays in crafting a controlled environment, an amalgamation of real-world systems and simulated traffic. This environment was meticulously created to imitate common network interactions while also foreseeing potential attacks. It was compartmentalised into multiple subnets and structured to emulate standard enterprise networks, encompassing routers, switches, and other archetypal network components to guide and oversee data flow. The authors made sure that both conventional and emerging attack vectors were included in the dataset by fusing real benign traffic with adversarial patterns generated by modern penetration testing tools.

The described dynamics were brought to life by imposing two separate actors: the Victim-Network, equipped with all the nowadays imperative protection mechanisms and network components in the following formula: three servers (one DC&DNS and two web servers), a firewall, two switches, a mirror port on the main switch (capturing the entire traffic circulation within the network), and ten hosts joined through a domain controller and active directory and invested with various editions of Windows, multiple Linux distributions, or even Mac OS; and the Attack-Network, a simpler infrastructure containing one router, one switch, and four hosts with Kali and Windows 8.1 operating systems.

The attacks were emulated using the best and most widely available tools for each over the course of four days, with an extra day reserved for benign traffic. The network traffic features were extracted using an open-source flow simulator developed in-house, which can obtain 80 features from a PCAP file, including but not limited to source IP, source port, and destination IP. The generated flow was then labelled using the time schedule for the attacks. On this dataset of 80 features, a Random Forest classifier was used by the authors to determine their relevance.

3.1.2 Data Features

The dataset includes 78 attributes, from rudimentary network metrics (like packet size) to more intricate indicators. Special emphasis was placed on curating a set of features that provide an encompassing view of the nature of both benign and malicious traffic. The flow level features, such as flow bytes, flow packets, and ACK flag count, are supplemented by derived features, including, but not limited to, mean and standard deviation columns calculated for various features.

3.1.3 Labels

The labelling system for the CIC-IDS2017 dataset is solely multi-class and, as the authors put into the spotlight, covers a plethora of attacks, gathering a number of 15 distinct labels, including the benign traffic category, which is a desirable trait of any IDS-intended dataset. Adversely, the highly disproportionate, and in some places lacklustre, representation of each attack may severely impact the correlation between the experimental performance of IDS and their real capabilities.

It is also worth mentioning that the CIC-IDS2017 dataset's approach of abundant labels to the detriment of each class' representation sharply contrasts with the approach ruling over the UNR-IDD dataset from Chapter 2, which attributed greater importance to the classes' representation itself than to their amount. This fact becomes evident upon juxtaposing Figure 2.2 with Figure 3.1.

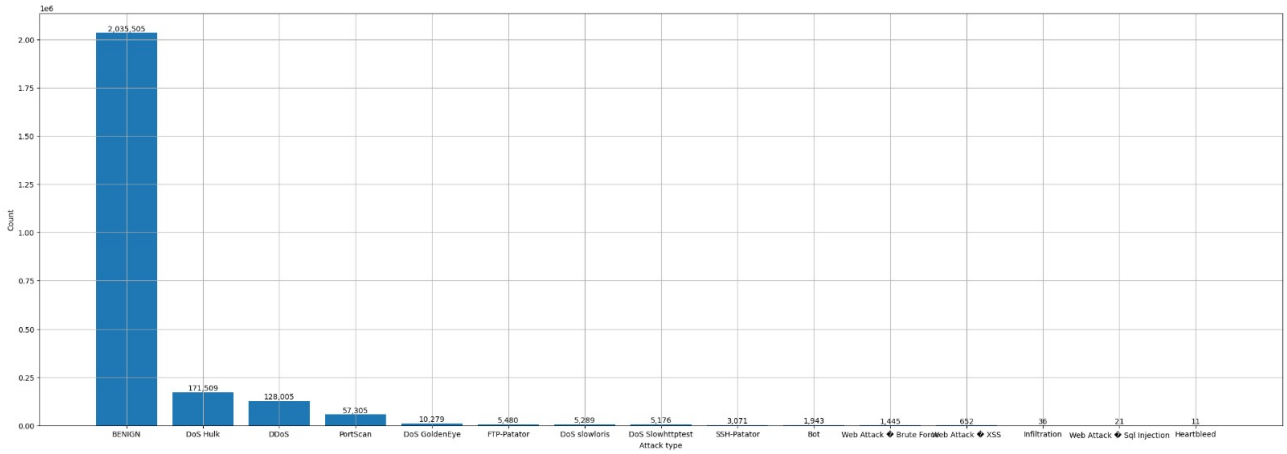

Fig. 3.1: Distribution of Attack Types in the CIC-IDS2017 Dataset

In light of the foregoing, besides the class denoting normal traffic and labelled "Benign", the attacks that were inserted in the dataset furnished the following labels:

- "DoS Hulk" - HTTP Unbearable Load King, a Denial of Service attack tool, is administered to overwhelm web servers with HTTP requests in massive quantities, thereby automating the process of launching HTTP-based DoS operations and exhausting the server's resources (memory, processing power, network bandwidth etc.) to prevent its usage by legitimate users. In other words, the Hulk tool makes it difficult for the target server to distinguish between legal and malicious traffic by generating a huge number of arbitrary HTTP GET requests, each with a different URL and set of query parameters.

- "DoS GoldenEye" - Similarly to Hulk, the GoldenEye tool aims to flood a web server with HTTP requests and render it unavailable to the allowed clients. However, more prominence is granted to the customizability of these requests rather than its simplicity and ease of use. The HTTP method can be any, GET, POST, or HEAD, each with randomised headers and content.
- "DoS slowloris" - This is a denial of service attack that aims to consume a server's resources by keeping a large number of concurrent connections open. The attack opens a large number of connections and sends requests as slowly as possible, ideally one byte at a time. This exploit presents a challenge to security systems as it does not bear the classical footprint of large spikes in network traffic.
- "DoS Slowhttptest" - Slowhttptest is the name of a security testing tool that allows specialists to perform different variations of the slowloris attack described above. Its main selling point is the ability to create very specific and controlled slowloris attacks with highly tunable parameters while measuring the subject's behaviour and performance under different slowloris scenarios.
- "DDoS" - A Distributed Denial of Service attack is an attempt to overwhelm a target network or service with traffic, making it unable to handle traffic from normal users, by using a large number of coordinated attackers, hence the distributed factor.
- "PortScan" - This one was also described in Chapter 2. To return to the earlier topic with a fresh perspective, port scanning enables hackers to hone their techniques and boost their success rates. It is fundamentally a reconnaissance technique used by attackers to learn about the network services, versions, and configuration, and occasionally even security measures like firewalls and intrusion detection systems, of a target host device, frequently preceding more malicious activities. Several ports on the target host are periodically scanned throughout this process in an effort to detect open ports; the found ports provide useful information about the target and allow attackers to further formulate attacks that utilise the identified vulnerabilities.
- "FTP-Patator" - This patator refers to an FTP brute forcer which guesses the target FTP server's password via trial and error. It is a flexible, multi-threaded tool which uses persistent connections to test several passwords until the server disconnects.
- "SSH-Patator" - Just like its FTP counterpart, this tool attempts to brute force passwords, but on an SSH authentication.
- "Web Attack Brute Force" - This attack attempts to obtain a user's login page credentials. While many techniques are employed to increase the number of passwords attempted and at the same time dodge rate limits, this attack is fundamentally based on the fact that most passwords are weak and easily guessable.
- "Web Attack Sql Injection" - When performing an SQL injection, the attacker attempts to manipulate a web application to execute malicious SQL code by inputting the code into the app's text input fields. If the input text is not sanitised of escape characters,

it may be delivered to an SQL server, which would treat the input as code instead of a character string and run it.

- "Web Attack XSS" - Cross-site scripting (XSS) attacks try to deliver a malicious JavaScript payload to be executed by an unsuspecting user. This exploit usually gives the user a modified link to a trusted web page, injecting the payload into the web page and allowing the attacker to record keystrokes or extract information that the logged-in user has access to.
- "Bot" - This label refers to the use of the Botnet Ares tool in order to remotely command multiple compromised computers by connecting them to a command and control server, which can upload and download files or instruct the bots to run shell commands and Python scripts.
- "Infiltration" - By exploiting vulnerable software, infiltration attacks trick the victim into executing scripts from inside a network, essentially giving the attacker backdoor access to the network. In the case of the CIC-IDS2017 dataset, this was simulated by the victim machines downloading a file from Dropbox or from an infected USB flash drive, which would execute an Nmap PortScan on the entire victim network.
- "Heartbleed" - Heartbleed is a vulnerability discovered in the OpenSSL cryptographic library, more specifically in its implementation of the TLS heartbeat functionality used to keep a connection alive. Attackers are able to send a maliciously crafted heartbeat request to a vulnerable server and receive a chunk of the server's memory content in response, potentially including sensitive data.

3.2 Deep Learning on CIC-IDS2017

This section follows the structure adopted in the previous chapter, delineating my personal experience with building a Python script that would process the dataset, serving results and insights from which valuable conclusions can be drawn. Akin to Chapter 2, my intention with this work is not to deliver an actionable application per se but rather to aid the journey towards constructing a competent IDS.

3.2.1 Feature Study and Data Preparation

To assess the relative significance of the 78 individual features within the dataset, Figure 3.2 displays their corresponding histograms. Given the augmented number of features, it was judicious to exclude the 10 data attributes with a standard deviation of less than 0.1 from the schema, streamlining the analysis process.

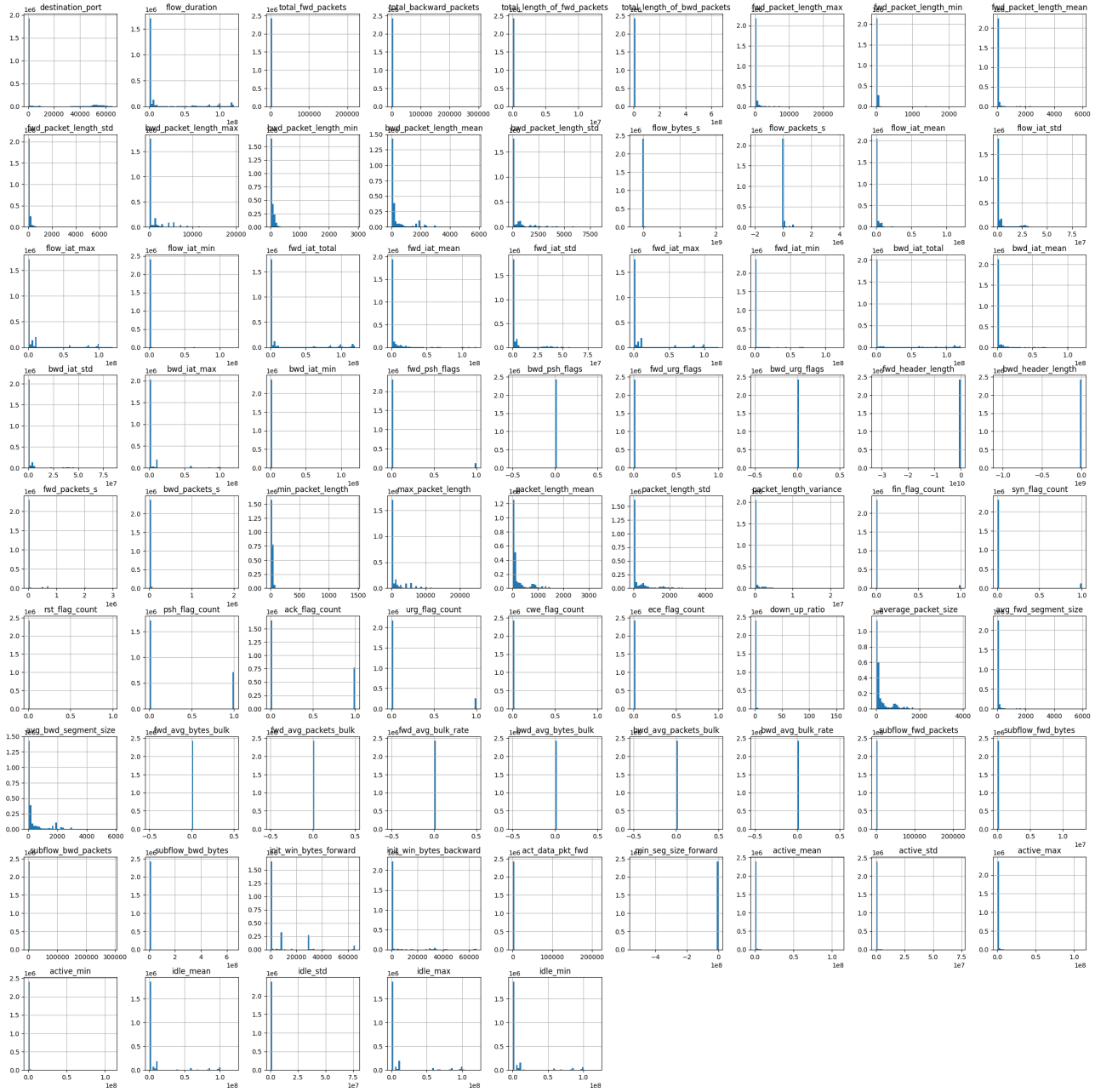

Fig. 3.2: Histogram Analysis of Features in the CIC-IDS2017 Dataset

To further reduce redundant information with regards to the data characteristics, exceedingly correlated features also had to be disposed of. This was done by computing a correlation matrix and subsequently dropping features with a correlation coefficient greater than 0.98. This led to 18 features being discarded.

Given the voluminous assortment of labels present within the dataset, several of which delineate analogous types of cyberattacks, albeit instantiated via divergent tools, it was deemed imperative to undertake a systematic relabelling of the dataset. The redefined labels have been scrupulously chosen as follows: Benign, PortScan, DoS/DDoS (encompassing labels such as

DoS Hulk, Goldeneye, slowloris, slowhttptest, DDoS, and Heartbleed), Brute Force (derived from the FTP and SSH Patator designations), Botnet ARES (subsuming the 'Bot' label), Web Attack (incorporating the Web Attack Brute Force, SQL Injection, and XSS designations), and Infiltration. This procedure confers the added advantage of producing better-represented labels, thereby bolstering the machine learning endeavour.

Pursuing the task of class representation, the predominant challenge to address is the vast disparity between instances of benign traffic and attack traffic. As depicted in Figure 3.1, the majority of the dataset consists of data points categorised as benign. Preliminary tests have demonstrated commendable performance for the benign class. Consequently, the decision to under-sample this class was made based on the inference that, beyond a certain threshold of data points, a machine learning model would derive minimal or no additional insight regarding the class. The benign data points were adjusted to reflect 50,000 fewer entries than the aggregate of all attack classes, forging an almost equitable distribution of the dataset between regular traffic and attacks, albeit with a slight tilt towards malicious traffic. With this modified dataset, a machine learning algorithm would place greater emphasis on comprehending the nuances of the attack classes, as it would no longer be rewarded with over 90% accuracy just for distinguishing the benign class from the others. The distribution of labels in the resulting dataset is displayed in Figure 3.3.

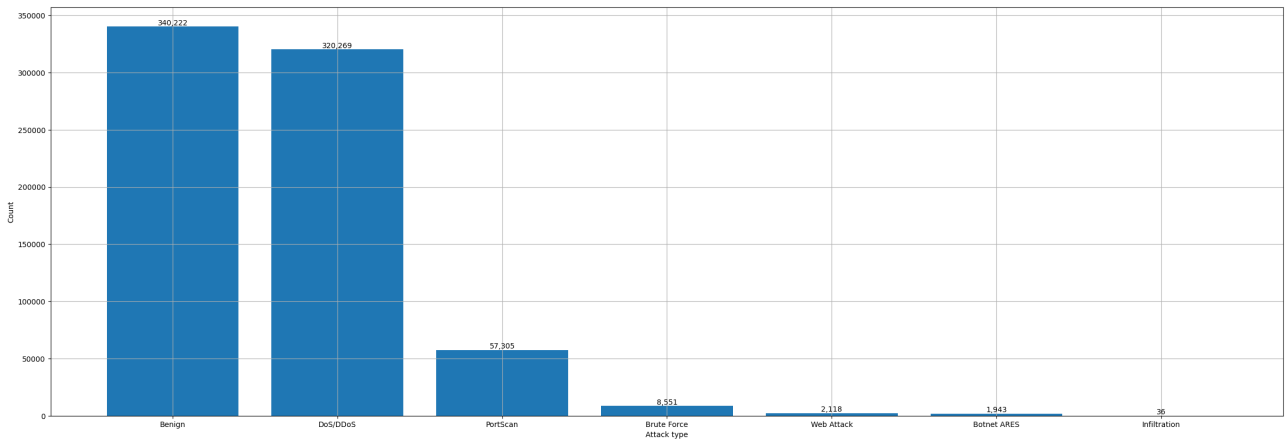

Fig. 3.3: Distribution of Attack Types in the Under-Sampled CIC-IDS2017 Dataset

Similarly to the feature selection previously performed on the UNR-IDD dataset, the final step for picking the right features to work with would be quantifying their relevance. Using the same technique as described in Chapter 2, a neural network, which will be analysed in the following section, was trained and tested on the dataset. Afterwards, each feature was set to zero, and the model would be evaluated on the resulting dataset. This approach determines the relevance of each feature as the difference between the model's original loss and the loss

obtained with the particular feature set to zero across the entire testing dataset. The 13 features that showed zero relevance, according to this algorithm, were removed from the dataset. While this approach does not guarantee a perfect quantification of each feature's importance, it does demonstrate that each selected feature has some degree of impact on the model's performance.

Finally, to prepare the dataset for use in the model, its columns were transformed into uniform distributions between 0 and 1. The uniform distribution was chosen over the normal distribution with the goal of better handling the large number of data points. The uniform distribution lessens the importance of single-feature outliers, especially those with the benign label, therefore encouraging the model to derive classification criteria based on a greater number of features, creating a more robust IDS that would ideally not rush to label an example as an attack based solely on an abnormal measurement for one of its features. This quantified relevance of each of the 36 remaining feature is depicted in Figure 3.4.

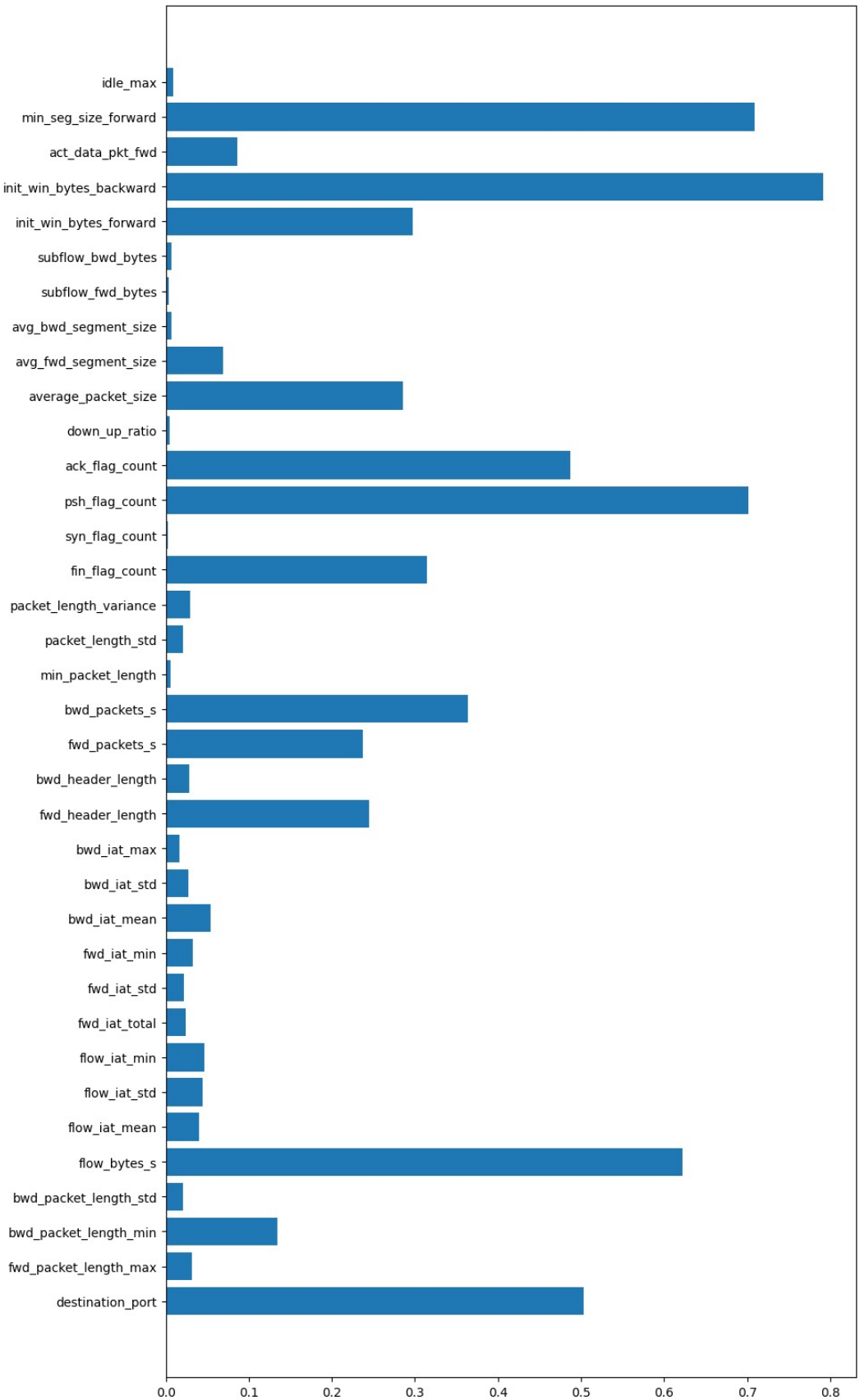

Fig. 3.4: Relevance of the Selected Features (CIC-IDS2017)

3.2.2 Model Architecture

This time, the model fabricated to simulate a small-scale AIDS comprises a self-contained dense neural network, and it examines exclusively the potential of ML in categorising learned attack patterns, in order to prepare the ground for an equitable comparison with the performance of an SIDS. Hence, within the context of this specific undertaking, the exploration of machine learning techniques for detecting zero-day attacks falls outside the intended scope.

The above-mentioned neural network was designed to incorporate an input layer made of 36 neurons, one for each preserved data feature, along with an output layer of 7 neurons, one for each label in the new set proposed earlier and activated by Softmax. The hidden anatomy of the DNN takes the following shape: Layer 1 (200 neurons) - 60% Dropout - Layer 2 (150 neurons) - Layer 3 (100 neurons) - Layer 4 (50 neurons) - Layer 5 (25 neurons) - 20% Dropout - Layer 6 (180 neurons), equipped with the ReLU activation function. The model's loss is calculated using the categorical cross-entropy function, and the Adam optimiser is the one designated to update the network's parameters during training. The learning rate specified is 0.0001.

The CIC-IDS2017 dataset was divided using the following framework to engage in the practical training and testing of the previously stated model: 56% training data, 14% validation data, and 30% testing data. Additionally, the model runs over 55 epochs, and the entry data is split into batches of 64 points.

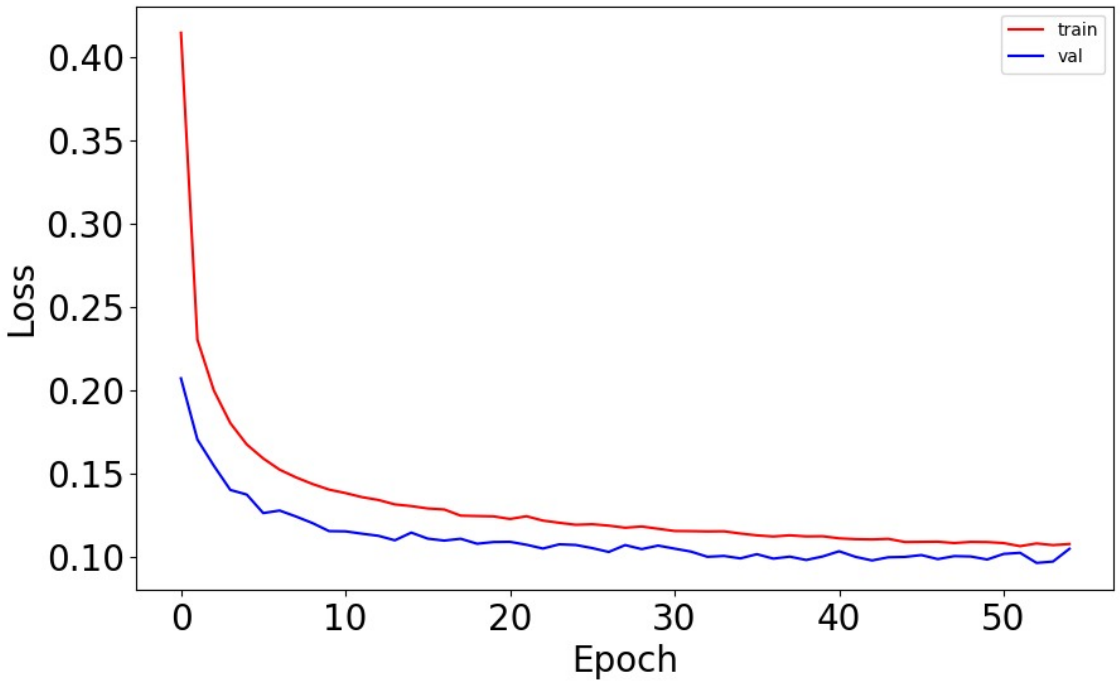

Fig. 3.5: Loss Function Graph (CIC-IDS2017)

Figure 3.5 exhibits the value of the loss function throughout the training process, with the difference between the training and validation losses excluding the case of overfitting.

3.2.3 Results and Conclusions

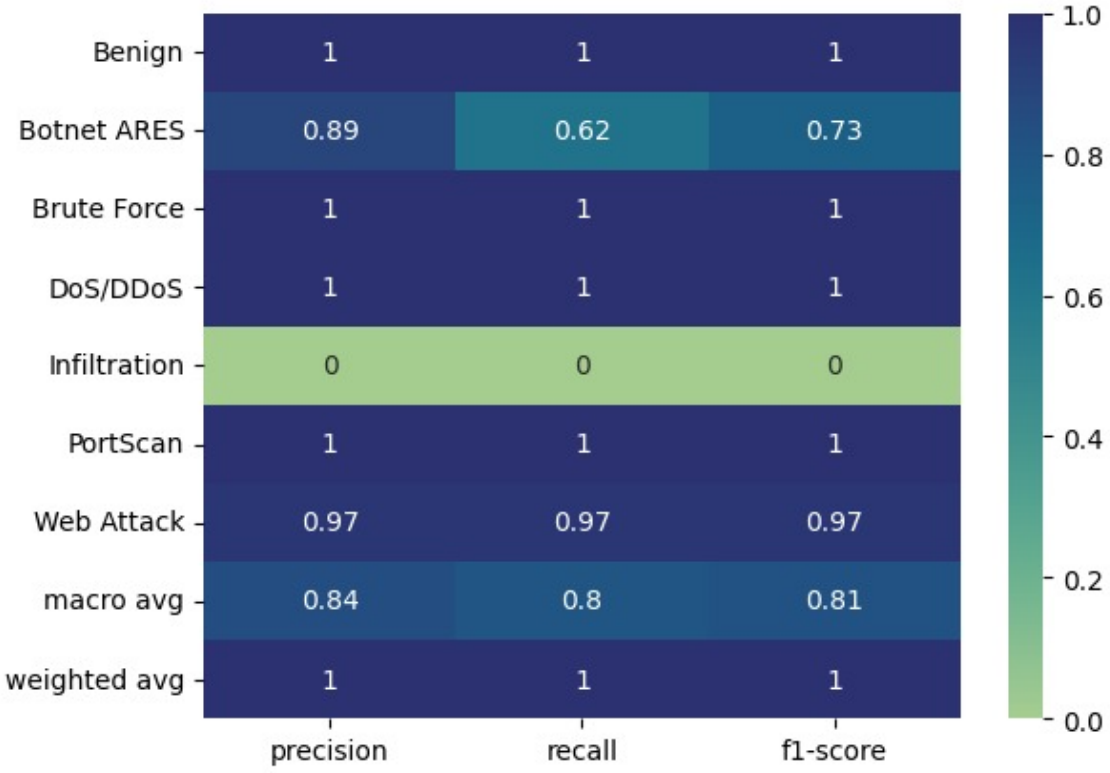

Fig. 3.6: Classification Report (CIC-IDS2017)

Running the aforementioned model on the testing dataset, an accuracy of 99.73% was achieved. To dive into the model's results and draw practical conclusions, an analysis of the classification report in Figure 3.6 is required, which presents the precision, recall, and f1-score for each class. Just from a quick glance, the neural network's abysmal performance on the Infiltration class is noticed. This result perfectly displays the main downside of machine learning models: their dependency on the volume of training data. Since the infiltration class contained only 36 data points, split into 25 training and 11 testing examples, it becomes clear that the model did not have enough examples to understand the attack's pattern. When considering the entire report, the results are strongly correlated with the representation of each class, with the two labels that have the lowest number of data points showing the lowest performance. Despite the lacklustre performance on the Infiltration and Botnet ARES classes, the model shows potential practical use due to its ability to properly differentiate between the rest of the attacks.

3.3 Snort on CIC-IDS2017

The current section completes the last one in the hopes of bringing forth a comparative analysis of a signature-based intrusion detection system, in the role of which Snort [2] was designated, relative to the machine learning anomaly-based intrusion detection system, which is embodied by the model characterised above.

3.3.1 Experiments

To start this experiment, the five PCAP files provided by the CIC-IDS2017 dataset have been broken down into smaller PCAPs, one for each 60-second interval. These resulting PCAPs are evaluated by Snort 2.9, equipped with its registered user rule package. While precise numbers for Snort's false negatives are not attainable, attack time intervals and approximate calculations related to the dataset's label counts have been used to evaluate the SIDS's performance on each class.

The first of the 5 PCAPs, which only contains normal traffic, was used to both tune the rule set and test its accuracy. Since this data is considered benign, most alerts raised, such as those regarding TCP packets being sent too slowly, have been disabled. However, any results that directly classify this traffic into one of the attack classes present in the dataset have been left untouched. Therefore, on Day 1, a number of 316 PortScan and DDoS alerts (false positives) have been raised for the 529919 data points, bringing this Snort setup's accuracy on benign traffic down to an approximate 99%. Despite the unclear correlation between an alert and the number of data points in the dataset, this estimate is enough to offer a general view of the SIDS's performance.

To evaluate Snort's capability to detect attacks, the first test was run on FTP and SSH Brute Force traffic. Since Snort does not ship by default with rules to detect these brute force attempts, two custom rules were implemented. While a one-to-one correlation between the dataset's entries and Snort's attack alerts does not exist, comparisons between the timestamp of each alert and the interval when the attacks were simulated on the testbed configuration show a good detection of FTP Brute Force attacks but a failure to detect those performed on the SSH port.

When tested on the packet captures from the third day, when DoS attacks were simulated, Snort proved to be a reliable asset in detecting all types of DoS attempts, clearly indicating the period when the malicious traffic was in place.

The fourth day came with its own challenges, in the form of web attacks in the morning

and infiltration attempts in the afternoon. The three web attack types were easily observed by Snort, with streams of meaningful alert messages being thrown during the period when the network was subjected to these attacks. During the infiltration attempts, however, Snort was unable to offer any actual insight into the security threat at hand. While spikes in alerts do indicate potentially bad traffic, "oversize request URI directory", the most common message during these intervals, simply refers to requests to longer URLs, which can be considered just a coincidence resulting from this exact network setup and not a direct correlation to an infiltration attempt's signature.

The fifth and final morning provides examples of the traffic created by the presence of a botnet, while in the afternoon a number of port scans are executed on the victim network. With the latter presenting no problem to the system, it is the botnet that warrants further discussion due to its presence being clearly observable but labelled by Snort as a DoS attempt. Therefore, the conclusion to be drawn is that, while a rule dedicated to botnets is not present, its creation would allow for easy detection of the threat since deciphering its signature is a small step when its presence is already very clearly observed.

3.3.2 Conclusions

Even in a sub-optimal scenario, with limited information about the network configuration and no possibilities for live testing, Snort is able to alert the presence of most threats. There is sufficient proof to indicate that, in an ideal application, Snort would be able to identify attacks with near-perfect accuracy. Despite the alert logs being polluted with PortScan false positives, a thorough analysis of the logs by a security professional would make them aware of the aforementioned malicious attempts against a network. Improving on the already good publicly available general use rule sets by adding more rules and fine-tuning the existing ones during live testing, one would obtain a reliable tool for network intrusion detection.

General Observations and Conclusions

When big data-driven approaches are concerned with improving intrusion detection, the journey this paper paves through the field of entrusting machine learning with the responsibility of detecting intrusion and potential security threats leads to the acknowledgement of several notable aspects that demand recognition. Each chapter delves into a specific perspective, with the literature review diving into previous achievements, milestones, and approaches; the demonstrations on the UNR-IDD dataset touching the possibilities of detecting zero-day attacks while exploring a carefully curated dataset; and the CIC-IDS2017 experiments painting a comparison between machine learning solutions and traditional rule-based systems on a dataset resembling a real-world use case. Following this examination, conclusions with regards to the overarching ideas observed throughout will now be elected.

Taking into account both the reviewed literature and my own experiments, the first thing that grabs one's attention should be the high accuracy proven by machine learning intrusion detection solutions, with all of them showing above 90% accuracy and a couple of them even going past the 99% milestone. Hence, it is clear that learning algorithms are worth considering for practical application in intrusion detection. For all that, whenever these algorithms are at stake, their effectiveness is strongly dependent on the training dataset.

An essential point observed during my experiments is the impact of the dataset's construction and characteristics on a model's performance. As previously discussed, the main difference between the two datasets is in the emphasis on class representation placed by UNR-IDD, with CIC-IDS2017 conversely focusing on providing a larger volume of data resembling a real-world scenario. As a result, let us demarcate the distinction between these two:

- UNR-IDD is a smaller dataset, with the attack scenarios representing an unrealistically large percentage of the traffic, for the purpose of aiding machine learning endeavours with a better representation of each label. Additionally, since it aims to be of use only to data science tasks, it does not provide resources particular to the field of cybersecurity, such as PCAP files, but offers carefully selected features with the goal of general applicability in mind.

- CIC-IDS2017 is a larger dataset, with millions of data points, depicting a scenario closer to the one encountered in a real application. As a consequence, benign traffic represents the large majority of the recorded data, with the attack time intervals adding up to only a small portion of the network's uptime. With the emphasis placed on practical cybersecurity usage, the dataset's numerous features are extracted from packet captures and may not bear importance for a machine learning algorithm. Also, PCAP files are annexed to the dataset, bringing various intrusion detection methods into the realm of possibilities.

To paint a complete picture of the relevance that picking the right dataset has, a discussion needs to be made about the labelling systems proposed by the two datasets at hand:

- UNR-IDD brings both binary and well-represented multi-class labelling formats to the table. Returning to the experiment I conducted on the dataset, a noteworthy observation would be that machine learning algorithms struggle to label a zero-day attack as such, so it should not be expected of them to handle the classification of unlearned incidents in a multi-label context. For this reason, binary labels have proven useful, as the proposed model can still tell a zero-day attack apart from normal traffic.
- CIC-IDS2017 brings another aspect of the labelling system into the spotlight, namely the degree of importance that the dimension of each class has. As such, a wider array of documented attacks does not necessarily lead to the development of a better intrusion detection system, unless the increase in diversity is backed up by proper representation of the attacks.

Tying the above observations together, it is crystal clear that building an adequate dataset is a cornerstone of machine learning IDS, not a mere preliminary step. An inadequate or biased dataset can compromise the generalisation capabilities of ML models, making them highly susceptible to false positives or negatives. As analysed by the existing literature [10], wrong classifications generally result in high costs and range from requiring a security specialist to review false alarms to the steep consequences of treating a cybersecurity threat as normal traffic.

Comparing the collection of anomaly-based IDS, which encompasses machine learning approaches, to traditional signature-based IDS leads to a case-by-case analysis weighing the benefits and disadvantages of each system, as neither shows superior results in all scenarios. Considering just how important a role the dataset plays in the success of an ML algorithm, any situation in which the construction of the dataset cannot be given the attention and resources it requires would warrant the use of a SIDS instead. In contrast, designing an AIDS would be justified when the necessary data is obtainable, as in-depth knowledge of an attack's inner

workings is not necessary and the added potential of stopping zero-day threats in their tracks is appealing to any network. While advanced cybersecurity knowledge is a pre-requisite to constructing a SIDS, with great effort being put into meticulous rule writing and testing, an AIDS can surpass that need by deriving its knowledge from large volumes of data, shifting the focus to a data science task.

Bibliography and References

[1] http://https://www.gartner.com/ - internationally renowned research and consultancy agency that offers perceptions, evaluations, and suggestions to assist firms in making wise choices in a range of technological and commercial matters.

[2] https://www.snort.org/.

[3] Adebowale Ajayi and Idowu S.A. "Comparative study of selected data mining algorithms used for intrusion detection". In: (July 2013).

[4] Rebecca Gurley Bace. *"Intrusion Detection"*. Macmillan Technical Publishing, 2000. ISBN: 1-57870-185-6.

[5] Guy Bruneau. *The History and Evolution of Intrusion Detection*. https://www.sans.org/white-papers/344/. 2021.

[6] Chuck Cartledge. *How many vs are there in big data?* 2016.

[7] Tapadhir Das. *UNR-IDD Intrusion Detection Dataset*. https://www.kaggle.com/datasets/tapadhirdas/unridd-intrusion-detection-dataset. About: https://www.tapadhirdas.com/unr-idd-dataset. 2023.

[8] Hervé Debar, M. Becker, and D. Siboni. "A Neural Network Component for an Intrusion Detection System". In: June 1992, pp. 240–250. ISBN: 0-8186-2825-1. DOI: 10.1109/RISP.1992.213257.

[9] A. K. Dewdney. *"Computer Recreations: Of Worms, Viruses and Core War"*. Scientific American. Mar. 1989.

[10] Luís Dias and Miguel Correia. "Big Data Analytics for Intrusion Detection: An Overview". In: *Handbook of Research on Machine and Deep Learning Applications for Cyber Security* Chapter 14 (July 2019), pp. 292–317.

[11] Min Du et al. "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1285–1298. ISBN: 9781450349468. DOI: 10.1145/3133956.3134015. URL: https://doi.org/10.1145/3133956.3134015.

[12] Nabila Farnaaz and M.A. Jabbar. "Random Forest Modeling for Network Intrusion Detection System". In: *Procedia Computer Science* 89 (2016). Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India, pp. 213–217. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.06.047. URL: https://www.sciencedirect.com/science/article/pii/S1877050916311127.

[13] Amirhossein Gharib et al. "An Evaluation Framework for Intrusion Detection Dataset". In: Dec. 2016, pp. 1–6. DOI: 10.1109/ICISSEC.2016.7885840.

[14] Manasi Gyanchandani, JL Rana, and RN Yadav. "Taxonomy of anomaly based intrusion detection system: a review". In: *International Journal of Scientific and Research Publications* 2.12 (2012), pp. 1–13.

[15] *Internet security threat report 2017*. Tech. rep. vol. 22. https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf. Symantec, Apr. 2017.

[16] Ansam Khraisat et al. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2.1 (2019), p. 20.

[17] Yinhui Li et al. "An efficient intrusion detection system based on support vector machines and gradually feature removal method". In: *Expert Systems with Applications* 39.1 (2012), pp. 424–430. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.07.032. URL: https://www.sciencedirect.com/science/article/pii/S0957417411009948.

[18] Hung-Jen Liao et al. "Intrusion detection system: A comprehensive review". In: *Journal of Network and Computer Applications* 36.1 (2013), pp. 16–24. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2012.09.004. URL: https://www.sciencedirect.com/science/article/pii/S1084804512001944.

[19] Julien Maury. "How Hackers Evade Detection". In: *eSecurityPlanet* (Apr. 2022). https://www.esecurityplanet.com/threats/how-hackers-evade-detection/.

[20] Yisroel Mirsky et al. *Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection*. 2018. arXiv: 1802.09089 [cs.CR].

[21] Chirag Modi et al. "A survey of intrusion detection techniques in Cloud". In: *Journal of Network and Computer Applications* 36.1 (2013), pp. 42–57. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2012.05.003. URL: https://www.sciencedirect.com/science/article/pii/S1084804512001178.

[22] Saurabh Mukherjee and Neelam Sharma. "Intrusion Detection using Naive Bayes Classifier with Feature Reduction". In: *Procedia Technology* 4 (2012). 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012, pp. 119–128. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2012.05.017. URL: https://www.sciencedirect.com/science/article/pii/S2212017312002964.

[23] Shanmugavadivu R and Dr.N.Nagarajan. "Network Intrusion Detection System using Fuzzy Logic". In: *Indian Journal of Computer Science and Engineering* 2 (Feb. 2011).

[24] Secureworks. *The Evolution of Intrusion Detection & Prevention.* https://www.secureworks.com/blog/the-evolution-of-intrusion-detection-prevention. 2017.

[25] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *International Conference on Information Systems Security and Privacy.* Dataset link: https://www.unb.ca/cic/datasets/ids-2017.html. 2018. URL: https://api.semanticscholar.org/CorpusID:4707749.

[26] G. Vigna and R. A. Kemmerer. "Intrusion Detection: A Brief History and Overview (Supplement to Computer Magazine)". In: *Computer* 35.04 (Apr. 2002), pp. 27–30. ISSN: 1558-0814. DOI: 10.1109/MC.2002.10036.

[27] Ting-Fang Yen et al. "Beehive: Large-Scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks". In: *Proceedings of the 29th Annual Computer Security Applications Conference.* ACSAC '13. New Orleans, Louisiana, USA: Association for Computing Machinery, 2013, pp. 199–208. ISBN: 9781450320153. DOI: 10.1145/2523649.2523670. URL: https://doi.org/10.1145/2523649.2523670.

[28] Richard Zuech, Taghi M. Khoshgoftaar, and Randall Wald. "Intrusion detection and Big Heterogeneous Data: a Survey". In: *Journal of Big Data* 2 (Feb. 2015), pp. 90–107. ISSN: 2196-1115.