Trustable machine learning for intrusion detection systems

Robin Duraz

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Informatique*

Par

Robin Duraz

Trustable Machine Learning for Intrusion Detection Systems

Thèse présentée et soutenue à IMT Atlantique, Brest, le 29 Novembre 2024
Thèse financée et préparée au sein de la Chaire de Cyberdéfense des Systèmes Navals
Unité de recherche : Lab-STICC
Thèse N° : 2024IMTA0433

Rapporteurs avant soutenance :

Eric TOTEL	Professeur, Télécom Sud Paris
Philippe OWEZARSKI	Directeur de recherche CNRS, LAAS-CNRS

Composition du Jury :

Président :	Philippe OWEZARSKI	Directeur de recherche CNRS, LAAS-CNRS
Examinateurs :	Eric TOTEL	Professeur, Télécom Sud Paris
	Isabelle CHRISMENT	Professeure, TELECOM Nancy
	Yufei HAN	Directeur de recherche, INRIA Rennes
Dir. de thèse :	Sandrine VATON	Professeure, IMT Atlantique
Co-encadrants :	David ESPES	Professeur, UBO
	Julien FRANCQ	Ingénieur de recherche, Naval Group (Ollioules)

Invité(s) :

Olivier BETTAN	Ingénieur de recherche, Thalès R&D

Acknowledgement

First, I would like to thank my supervisors, Sandrine, David and Julien, who supported me during this thesis. The freedom they provided allowed me to conduct this research in the way I wanted, allowed me to spend time on subjects that were difficult to tackle. Their inputs on my work also allowed me to continuously improve their quality.

I would like to thank the jury members for their comments on the manuscript and the interesting discussion during the defense. Their comments also allowed me to discover potentially new areas of improvements for my work. I would like to further thank Philippe Owezarski that managed to review my thesis despite the relatively short notice.

I also would like to thank my colleagues at both the Chaire of Naval Cyberdefense, David, Jessica, Sebastien, Quentin, Paul, Clet, Douraid, Pierre and Maxence, and IMT Atlantique, Sanaa, Chahrazed, Coraline, Armelle, Ihab, Gaelic, Gwendal, Ouail, Pierre, Stanislas, Alexandre, Illyas and all the others for their warm welcome. Their presence allowed me to relax in the difficult times, especially in the difficult exercise of writing this manuscript.

Finally, I would like to thank my family for being present, for their continuous support and for all the help they provided during my life, without which I probably would not have gone so far.

Résumé en français

Cette thèse explore les outils d'apprentissage automatique appliqués à la détection d'intrusions, et plus particulièrement comment rendre ces outils utiles sur le terrain pour l'aide à la décision. Elle explore également les outils supplémentaires dont on a besoin pour leur faire confiance pour les tâches qui leur sont confiées. La détection d'intrusion étant actuellement confiée à des outils créés et utilisés par des experts en cybersécurité, l'utilisation d'outils créés par des experts en apprentissage automatique pour des experts en cybersécurité demande des réflexions supplémentaires. La cybersécurité ayant des contraintes fortes, et une mauvaise décision pouvant entraîner des conséquences plus préjudiciables que l'attaque à prévenir, il est important d'être certain de l'action à réaliser en cas d'alerte relevée par un tel outil.

Les avantages principaux des outils d'apprentissage automatique sont leur haute performance, ainsi que leur potentiel pour détecter des attaques jusqu'alors inconnues (aussi appelées attaques Zero-day). Cependant, les méthodes utilisées pour obtenir de hautes performance de détection sont souvent basées sur des algorithmes dits «boîtes noires». De ce fait, l'utilisation de ces outils implique la nécessité de comprendre comment ils fonctionnent pour prendre les décisions appropriées. De plus, leur apparente haute performance repose sur des métriques d'apprentissage automatique, qui ne sont pas forcément adaptées à l'application à la cybersécurité.

Ainsi, cette thèse se penche sur la façon d'adapter correctement ces outils au domaine de la cybersécurité pour mieux leur faire confiance, ce qui passe par la définition de nouvelles métrique plus adaptées au domaine, à l'exploration et utilisation des outils d'explication, ainsi qu'au perfectionnement de ces outils pour mieux détecter des attaques inconnues.

Contributions

Les travaux portant sur la définition de nouvelles métriques adaptées au domaine de la cybersécurité ont donné lieu à deux publications, une première dans une conférence internationale présentant les nouvelles métriques, ainsi que leur utilisation pour différentes

techniques d'apprentissage automatique sur différents jeux de données publics, tandis que la seconde étend l'utilisation de ces métriques pour prendre en compte les spécificités du système à protéger, ainsi qu'à améliorer l'entraînement des techniques d'apprentissage.

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2023. Cyber Informedness: A New Metric using CVSS to Increase Trust in Intrusion Detection Systems. In Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference (EICC '23). Association for Computing Machinery, New York, NY, USA, 53–58. https://doi.org/10.1145/3590777.3590786

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2024. Using CVSS scores can make more informed and more adapted Intrusion Detection Systems. In Journal of Universal Computer Science Special Issue Fighting Cybersecurity Risks from a Multidisciplinary Perspective, vol. 30, no. 9 (2024), 1244-1264, submitted: 27/9/2023, accepted: 30/5/2024, appeared: 14/9/2024, https://doi.org/10.3897/jucs.131659

Ensuite, une contribution a été apportée sur l'utilisation de méthodes externes d'explication des méthodes d'apprentissage automatique, afin d'expliquer les décisions, et l'utilisation de ces outils d'explication pour corriger des erreurs et obtenir de meilleurs performances.

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2023. Explainability-based Metrics to Help Cyber Operators Find and Correct Misclassified Cyberattacks. In Proceedings of the 2023 on Explainable and Safety Bounded, Fidelitous, Machine Learning for Networking (SAFE '23). Association for Computing Machinery, New York, NY, USA, 9–15. https://doi.org/10.1145/3630050.3630177

Pour finir, une contribution a été apportée sur l'amélioration de la capacité des méthodes d'apprentissage automatiques à détecter des attaques auparavant inconnues (et donc très difficilement détectables par les outils existants). Cette contribution a été publiée dans une conférence internationale.

Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2024. SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization. In the 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria. https://doi.org/10.1145/3664476.3664505

Finalement, cette thèse est structurée en cinq chapitres. Le premier offre une introduction au sujet, et présente les différents éléments nécessaires à la compréhension du sujet. Le second chapitre met en évidence les déficiences des métriques d'évaluation actuelles et présente de nouvelles métriques adaptée au domaine de la cybersécurité, ainsi que les potentiels bénéfices liés à l'utilisation de ces métriques. Le troisième chapitre discute des méthodes externes d'explication des algorithmes d'apprentissage automatique, de leur utilisation pour expliquer une décision, ainsi que la façon dont elles peuvent être utilisées pour améliorer les performances de détection. Le cinquième chapitre développe les techniques employées dans cette thèse pour augmenter la capacité d'un outil de détection d'intrusion à détecter des attaques inconnues. Finalement, une conclusion aux travaux de thèse ainsi que des perspectives sont présentés pour conclure les travaux, ainsi que présenter des voies d'amélioration.

Apprentissage automatique appliqué à la détection d'intrusions

Avec l'arrivée de l'internet dans les années 90, les premières cyberattaques ont pu avoir de bien plus gros impacts. Les méthodes d'apprentissage automatiques ont été développées en parallèle des méthodes basées sur des signatures qui reposent sur des experts définissant des règles de détection. Tandis que les méthodes basées sur signature ont été utilisées commercialement, les méthodes d'apprentissage sont majoritairement restées au stade de la recherche.

Les méthodes d'apprentissage peuvent être séparées en trois catégories principales, suivant l'utilisation du jeu de données d'entraînement :

— Les méthodes supervisées, qui nécessitent des étiquettes pour chaque donnée, afin de correctement séparer les données appartenant à différentes classes.

— Les méthodes non supervisées, qui ne nécessitent pas d'étiquettes et qui permettent de compartimenter les données en utilisant des notions de similarité.
— Les méthodes semi-supervisées, qui ne nécessitent que quelques étiquettes et utilisent différentes méthodes pour séparer les données en utilisant à la fois les données étiquetées, et celles qui ne le sont pas.

Pour entraîner ces méthodes, des jeux de données publics sont en général employés, la majorité disposant d'étiquettes. Durant cette thèse, différents jeux de données ont été utilisés :

— Des jeux de données réseaux : CIC-IDS2017, UNSW-NB15 et DAPT2020.
— Un jeu de données industriel : WADI.

La qualité des jeux de données étant primordiale, et à défaut d'en constituer un soi-même, les travaux réalisés ont été basés sur les jeux de données les moins problématiques possible.

Différentes métriques d'évaluations peuvent être utilisées, chacune présentant des avantages et inconvénients. Dans le cadre de cette thèse, les métriques retenues l'ont été afin d'obtenir une vision la plus poussée possible des performances «réelles» d'une méthode d'apprentissage automatique.

Finalement, l'augmentation de la confiance en ces outils passe par différentes améliorations : une performance ayant du sens pour les utilisateurs de ces outils, la capacité d'expliquer comment ces outils arrivent à une décision, ainsi que leur capacité à faire ce dont les outils actuels ne sont pas capables.

Intégration de connaissances en cybersécurité à travers les scores CVSS

Dans différents domaines d'application des méthodes d'apprentissage automatiques, des métriques d'évaluation spécifiques ont été développées pour correctement évaluer la performance, comme BLEU et ROUGE pour le traitement du langage. Cependant, ce n'a pas été le cas pour la détection d'intrusion.

Différentes possibilités existent pour remédier à cela, et il a été choisi pour cette thèse d'intégrer les scores CVSS (pour Common Vulnerability Scoring System) pour créer de nouvelles métriques d'évaluation. Les scores CVSS sont utilisées par la communauté pour noter la sévérité des vulnérabilités nouvellement détectées, et sont très largement utilisées. De plus, ce sont des scores numériques, relativement simples à intégrer à des métriques.

Ainsi, trois nouvelles métriques ont été définies : Miss Cost, qui réprésente le coût des attaques manquées, False Alarm Cost, qui représente le coût des comportements normaux faussement considérés comme des attaques, et Cyber Informedness qui agrège les deux métriques précédentes. Des expériences ont été réalisées pour valider l'utilité de ces métriques, et les comparer aux métriques existantes.

Un intérêt supplémentaire des nouvelles métriques repose sur leur capacité à prendre en compte la spécificité d'un système via la modification des scores CVSS par des paramètres environnements (besoin du système à protéger). De ce fait, il est possible de comparer l'adaptation des différents outils de détection d'intrusions au système à protéger.

Finalement, en intégrant les scores CVSS à la fonction de perte d'algorithmes de réseau de neurones d'une façon analogue à la définition de nouvelles métriques, il est possible dès l'entraînement d'adapter un outil de détection d'intrusions pour qu'il devienne plus performant sur un système spécifique à protéger.

Explicabilité pour les outils de détection d'intrusion : Quoi, pourquoi et comment ?

Comme les outils de détection d'intrusions utilisent majoritairement des algorithmes dits «boîtes noires», il est nécessaire d'être capable d'expliquer comment ils arrivent à une décision, pour que l'utilisateur puisse prendre une décision éclairée. Les explications étant dépendantes des connaissances de l'utilisateur, il est souvent difficile de fournir des explications utiles.

Parmi les différentes méthodes d'explication existantes, les travaux de cette thèse se sont concentrés sur les outils d'explication dits externes, et surtout sur les deux méthodes les plus employées : LIME et SHAP. Une expérimentation approfondie a montré des manques quant à leur capacité à fournir des explications utiles. Cependant, ces méthodes peuvent être utilisées pour identifier des comportements suspects ou trouver et corriger des erreurs présentes dans le jeu de données.

Finalement, dans un but d'évaluation de la qualité de ces méthodes d'explications, des métriques ont été implémentées. Grâce à ces métriques, une méthode d'identification et de correction des erreurs de prédiction a été développée. Cette méthode peut simplement identifier et demander une correction par un expert humain, entraînant une hausse de la charge de travail, ou bien corriger automatiquement, pour une quantité de correction inférieure.

Des expérimentations sur trois jeux de données ont montré la capacité de cette méthode à identifier et potentiellement corriger des erreurs de prédictions, particulièrement pour des classes d'attaques très mal détectées.

Détection et classification d'attaques Zero-day

Un des principaux avantages des méthodes d'apprentissage automatique par rapport aux méthodes basées sur les signatures sont leur potentielle capacité à détecter des attaques auparavant inconnues. Cependant, cela repose en général sur des méthodes d'apprentissage non supervisées, qui sont soit incapables de différencier différentes classes, soit perdent en performance de détection, particulièrement pour des attaques connues (qui disposent d'étiquettes).

Les méthodes semi-supervisées permettent de potentiellement remédier à ce problème, mais ont une performance encore insuffisante, particulièrement dans le cas où il pourrait y avoir plusieurs nouvelles attaques inconnues. De ce fait, les travaux développés dans cette thèse se basent sur le paradigme de l'apprentissage par contraste, fonctionnant avec des réseaux de neurones, qui a révélé une performance très élevée en classification d'images, même en l'absence d'étiquettes. Cependant, les méthodes employées sont difficilement applicables au contexte et aux données de détection d'intrusions. De ce fait, le problème a été contourné en basant la méthode sur de l'apprentissage par contraste supervisé. Du fait de la supervision, les performances de détection sont très élevées pour les classes étiquetées, mais sont assez faibles pour les classes inconnues. Pour remédier à cela, différentes techniques de régularisation ont été utilisées pour améliorer la capacité à généraliser et détecter des attaques inconnues. Trois méthodes ont été utilisées : dropout, communément employée avec des réseaux de neurones, Von Neumann Entropy, qui est une technique permettant de mieux répartir l'information apprise, et Sepmix, une méthode développée dans le cadre de cette thèse pour pallier à l'impossibilité d'apprendre des classes rares en générant de nouvelles données fictives.

Les résultats obtenus par la méthode développée ont montré une capacité à détecter des attaques connues similaire aux méthodes supervisées, ainsi qu'une capacité intéressante à détecter des attaques inconnues, même quand il y en a plusieurs.

Conclusions et travaux futurs

Cette thèse a exploré les outils de détection d'intrusions utilisant des techniques d'apprentissage automatique, et essayé d'apporter différentes contributions pour augmenter la confiance que l'on peut accorder à ces outils dans une utilisation en situation réelle. De nouvelles métriques ont été développées pour évaluer d'une façon qui a plus de sens pour les utilisateurs finaux. Des méthodes d'explicabilité ont été utilisées pour expliquer le processus de décision de ces techniques, et à défaut d'explications utiles, ont été utilisées pour trouver et corriger des erreurs dans les décisions données. Finalement, une méthode basée sur l'apprentissage par contraste et utilisant différentes techniques de régularisation a été développée pour augmenter la capacité à détecter des attaques inconnues.

Pour les travaux futurs, il est premièrement envisagé de combiner les différents travaux réalisés pour créer une solution plus complete. Ensuite, il est envisagé d'explorer des techniques d'apprentissage automatique plus complexes, comme les réseaux de neurones en graphe, qui peuvent améliorer les performances, ainsi qu'apporter de nouvelles possibilités quant à la génération d'explication de la décision.

Finalement, il convient de garder à l'esprit différentes limitations de ces outils pour mitiger leur impact, comme la dépendance au jeux de données et leurs qualités, ainsi que la résistance de l'outil en lui-même aux différentes attaques qui auraient pour but de l'empoisonner ou le tromper.

Table of Contents

Acronyms

AE Auto-Encoder.

AI Artificial Intelligence.

APT Advanced Persistent Threat.

AUC Area Under the ROC Curve.

BI Bookmaker Informedness.

CE Cross-Entropy loss.

CE Contrastive Encoder.

CI Cyber-Informedness.

CIA Confidentiality, Integrity and Availability.

CL Contrastive Learning.

CNN Convolutional Neural Network.

CVSS Common Vulnerability Scoring System.

DDoS Distributed Denial of Service.

DL Deep Learning.

DNN Deep Neural Network.

DoS Denial of Service.

DT Decision Tree.

FAC False Alarm Cost.

GAN Generative Adversarial Network.

GNN Graph Neural Network.

ICS Industrial Control System.

IDS Intrusion Detection System.

IF Isolation Forest.

IoT Internet of Things.

IoU Intersection over Union.

IT Information Technology.

LLM Large Language Model.

LOF Local Outlier Factor.

LR Linear Regression.

LSTM Long-Short Term Memory.

MC Miss Cost.

MCC Matthews Correlation Coefficient.

ML Machine Learning.

NLP Natural Language Processing.

NN Neural Network.

OC-SVM One-Class Support Vector Machine.

OOD Out-of-Distribution.

OSL Open-Set Learning.

OT Operational Technology.

OWL Open-World Learning.

PLC Programmable Logic Controller.

RF Random Forest.

RNN Recurrent Neural Network.

SVM Support Vector Machine.

VAE Variational Auto-Encoder.

VNE Von Neumann Entropy.

XAI Explainable AI.

ZDA Zero-Day Attack.

Chapter 1

Introduction

1.1 Machine Learning and Cybersecurity

Both Machine Learning (ML) and cyberattacks are almost as old as computers. The idea of machines able to learn can be traced back to 1943 [117] with the first explanation of what will become a Neural Network (NN). Meanwhile, the first famous cyberattack can be traced back to 1971 with the Creeper, a self-replicating virus that used ARPANET, the parent of Internet, to infect multiple machines.

In the 1990s and 2000s, while Information Technology (IT) systems quickly became integrated to networks and even connected to the Internet to more easily collect and share data, Operational Technology (OT) systems remained largely self-contained and disconnected from networks. As a result, two different approaches to cybersecurity existed for, respectively, IT and OT systems. For IT systems that were connected through networks, approaches such as network segmentation, the principle of least privilege, or Intrusion Detection Systems (IDSs) were applied or developed to reduce the risk. Meanwhile, for OT systems, security relied mainly on the fact that equipment was disconnected from networks and used proprietary software.

The interest in using ML for cybersecurity, especially for detecting intrusions increased since 1999, with the 1999 DARPA Intrusion Detection Evaluation [1]. This also created the first public dataset that could be used to train ML algorithms in detecting intrusions, later giving birth to the KDD'99 dataset [88]. The increased amount of data collected, because equipment is connected through networks, also participated in promoting the use of Big Data methods, such as ML.

Since then, the digital pervaded our society, with everything from our financial system and healthcare machinery to our vehicles relying on some kind of automation or even autonomous and interconnected systems. While the relatively short life-cycle and ease of

1. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset. Last accessed 2024-06-05.

access of IT equipment allowed for frequent updates in both hardware and software to cater to cybersecurity's needs, OT equipment was and is still used in contexts where it might remain in use for a decade or even decades, e.g., power plants, naval ships, etc. During the last decade, however, OT equipment became increasingly connected through networks with a convergence of IT and OT through the development of Internet of Things (IoT) and Big Data technologies. This change was further accelerated during the COVID-19 pandemic where connection through Internet allowed for maintenance and control with remote access. In this context, both IT and OT can benefit from similar protection mechanisms, e.g., network segmentation, IDSs, with the additional requirements that OT needs to consider the increased vulnerability of old equipment.

In parallel, ML regained interest since the 2010s with the arrival of Deep Learning (DL) models, able to recognize images as well as, if not better than humans. Since then, multiple new areas of research have been developed, notably with Natural Language Processing, and more recently the arrival of powerful generative models such as ChatGPT.

In order to properly benefit from ML in intrusion detection, there are several important questions and factors that need to be taken into account. These questions and factors will be developed in the next section.

1.2 Intrusion Detection

It would of course be ideal to have perfectly secure systems, and security by design certainly contributes [12]. However, while it is possible to increase security, it might prove impossible to perfectly secure systems. As such, IDSs are important components to reduce risks by detecting cyberattacks before they can impact or compromise the targeted systems.

Traditional approaches are generally based on signatures, i.e., manually designed rules, that are able to properly detect known cyberattacks with a low number of false alarms. For example, Snort [141] allows to define network rules that match characteristics of communication between devices, e.g., ports, protocols, authentication attempts, etc., and raise alerts. They require much work to maintain because every new attack needs to be investigated to extract relevant patterns and design detection rules. A study in 2014 [76] showed that using standard rule sets also leads to relatively low detection of intrusion. Although the rule set used was able to detect close to 20% of zero-day attacks, i.e., attacks the rules were not designed for, the detection rate of known attacks was only close

to 50% [2]. In [145], specific Snort rules were used and achieved a very high detection rate of attacks, while having no false alarms [3]. Although correctly configured signature-based methods can achieve a high detection rate, with possibly no false alarms, they definitely struggle to detect attacks their rules are not created for. Furthermore, as the number of rules increases, rules also increase in complexity, which makes them harder and harder to update. Therefore, other approaches could be used concurrently to complement them.

ML approaches are more diverse in their capabilities and show promises in detecting both known cyberattacks and zero-day attacks. Such approaches are enabled by and rely on collecting data to train ML algorithms. These algorithms can globally be differentiated according to their requirements on labeled data, i.e., data that possesses information about what it represents. The high diversity of possible algorithms provides various capabilities, such as zero-day detection with unsupervised methods (does not require labeled data), or a high detection rate of known attacks with supervised methods (requires labeled data). Semi-supervised methods are positioned between both and aim at correctly detecting known attacks while having the ability to detect Zero-Day Attacks (ZDAs).

Despite the possible advantages of ML-based methods for intrusion detection, they also need to fulfill several requirements to be properly used in an operational context. While an autonomous anomaly-based IDS might seem exciting, it is difficult to fully rely on it, especially because it is operating in a critical context and could make mistakes difficult to recover from. As such, it is better to, at least for now, envision anomaly-based IDSs as a decision-making support for human operators.

Therefore, anomaly-based IDSs should first and foremost be able to provide relevant information when an alert is raised. One obvious information is the type or a categorization of the alert raised, to facilitate investigations. For ML, this would be enabled by training models in a multi-class setting, whereby it would be able to recognize and differentiate between normal traffic and multiple cyberattacks.

Another important factor is the amount of trust that can be given to such an IDS. Since their training prediction processes are often quite obscure, it is relatively difficult to understand which criteria triggered an alert. In a context where detection is not perfect and false alarms can be frequent (a particular disadvantage of ML methods), not being able to trust the IDS can reduce its effectiveness by wasting the operators' time, or worse, give a false sense of security. This is particularly aggravated by the fact that attacks are

2. The author warns that the rule set used is possibly less effective than the average Snort rule set.
3. This high performance compared to [76] can also be due to improvements in Snort over the years.

often relatively rare, and thus most alerts that need to be investigated by human experts could actually be false alarms.

Finally, information systems all have different security needs, and have to be protected from cyberattacks with varying impacts. As such, anomaly-based IDSs should not be too generic and should be able to consider both requirements of a system and the gravity of cyberattacks. It would make it more effective by being able to prioritize what really endangers the systems it is supposed to protect.

1.3 Positioning

In this thesis, the subject of ML for IDSs is explored, with an emphasis on building IDSs to help human decision-making. As stated previously, autonomous systems are not yet desirable, both for the risk of failure, as well as practical or legislative reasons about responsibility in case of failure. As such, the typical (and somewhat ideal) use-case envisioned is presented in Figure 1.1.

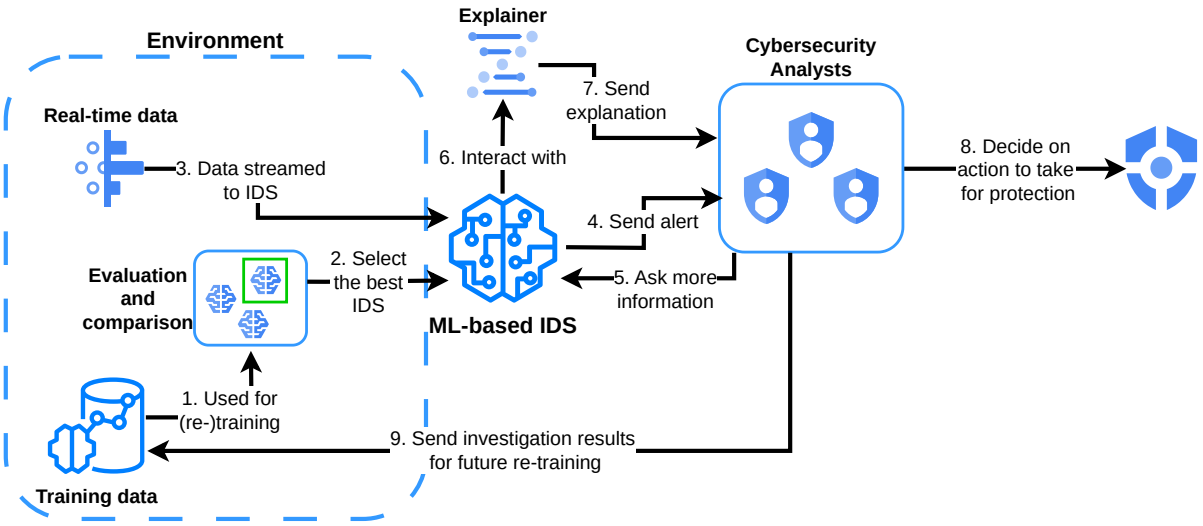


Figure 1.1 – Typical use-case for an IDS based on ML to help decision-making.

An initial dataset possessing labels for normal traffic and different cyberattacks is used to train different IDSs. Once evaluated and compared, the best IDS is chosen. Once operational, real-world data is streamed to the IDS that will try to detect cyberattacks, then raise an alert when a cyberattack is detected. Cybersecurity experts can choose to trust the alert and act on it, ask for more details, or manually investigate the alert. If

asked for more details, an IDS should ideally be able to provide explanations about the cause of its prediction to either support investigations or actions taken by human experts. Once specifics of the alert are validated, actions to protect the system need to be taken, and the alert can be labeled and integrated to the training dataset for potential future re-training of the IDS.

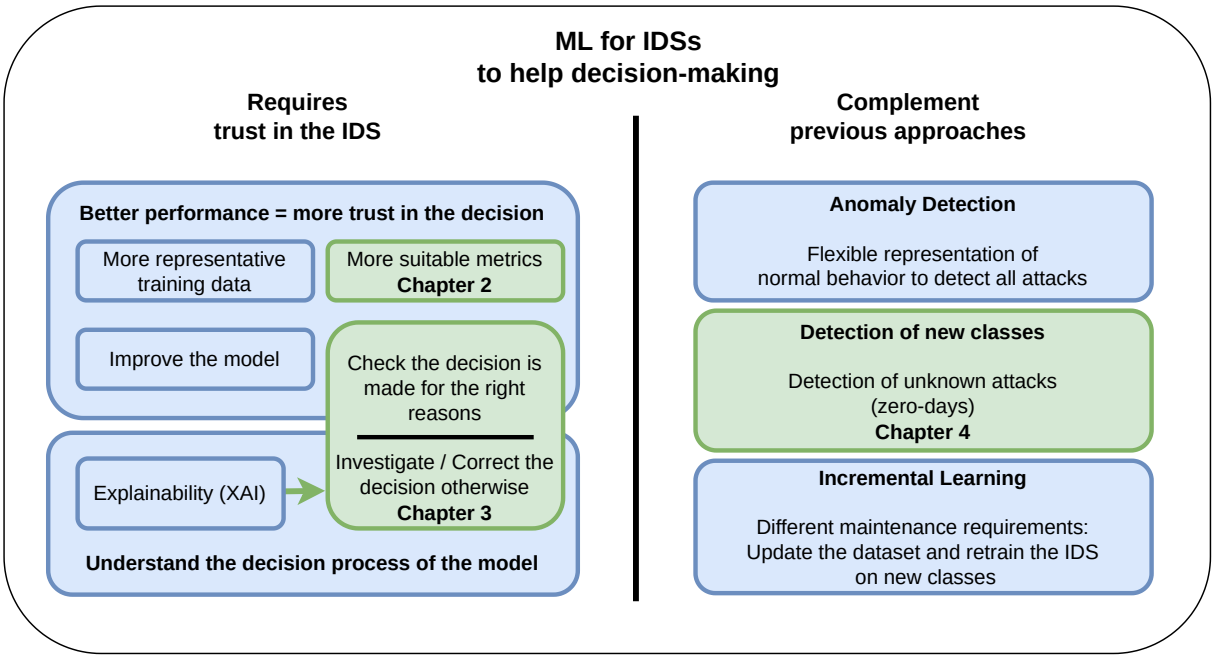


Figure 1.2 – Positioning of this thesis on the topic of ML for IDSs. Contributions made in this thesis are highlighted in green.

In order to properly help in decision-making, human experts need to trust the decisions of the IDS. As shown in Figure 1.2, two different axes can be researched to increase trust in the IDS:

— **Impact performance.** This can be done by improving on ML methods to achieve better detection and less false alarms, as has mostly been the case yet. Or it can be done by making performance more representative of real-world scenarios, by using data representative of the real world, or by using metrics better suited to the intrusion detection problem.

— **Understand the decision process of the IDS.** This can be done by using ML methods easily understandable by humans, e.g., Decision Trees, or using external tools, e.g., from Explainable AI (XAI), to explain the decision process.

In this thesis, since the basis of a proper comparison between approaches are a high

quality dataset and suitable metrics, we first focused on developing a metric leveraging cybersecurity information to perform a more informed comparison. We further contributed to the trust aspect by leveraging XAI to verify that ML models make sufficiently informed predictions, and potentially correct these predictions in case of uncertainty. Finally, we used ML approaches to develop an area where signature-based approaches are less efficient: the detection of ZDAs.

1.4 Manuscript Structure and Contributions

Although the topic will not be developed further in this thesis, a first contribution was made highlighting the first results obtained concerning ML and visualization tools for cyberattack detection:

> Robin Duraz, David Espes, Julien Francq, Sandrine Vaton. Machine Learning and Visualization tools for Cyberattack Detection. RESSI 2022 : Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information, May 2022, Chambon-sur-Lac, France. ⟨hal-03647627⟩

Chapter 1 expands on existing research on intrusion detection, on what exists for anomaly-based IDSs as well as possible objectives and goals beyond intrusion detection. It presents the different methods and requirements to increase trust in an IDS built with ML. It also briefly introduces ML techniques that are quite novel in other ML applications and could be adapted for IDSs. Finally, it elaborates on areas that complement signature-based approaches, where ML methods present significant advantages.

Chapter 2 is about improving the integration of cybersecurity knowledge into anomaly-based IDSs. It is possible to make ML methods more adapted through metrics using Common Vulnerability Scoring System (CVSS) scores. They can take into account both differences in impact of attacks, as well as specific characteristics of the systems to protect. Integrating CVSS scores into the evaluation process and training process led to two contributions:

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2023. Cyber Informedness: A New Metric using CVSS to Increase Trust in Intrusion Detection Systems. In Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference (EICC '23). Association for Computing Machinery, New York, NY, USA, 53–58. https://doi.org/10.1145/3590777.3590786

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2024. Using CVSS scores can make more informed and more adapted Intrusion Detection Systems. In Journal of Universal Computer Science Special Issue Fighting Cybersecurity Risks from a Multidisciplinary Perspective, vol. 30, no. 9 (2024), 1244-1264, submitted: 27/9/2023, accepted: 30/5/2024, appeared: 14/9/2024, https://doi.org/10.3897/jucs.131659

Chapter 3 is about making anomaly-based IDSs more trustable through the usage of XAI. Although current XAI methods are unable to provide to humans a clear explanation of an IDS's decision, they can still be applied as an additional check on predictions to increase their reliability, or possibly correct them in case of uncertainty. For the latter case, this can even increase performance of the IDS by correcting false alarms or detecting previously missed cyberattacks. Such a process could either be an additional tool requiring intervention of a human expert for maximum performance or be fully automated for a lower workload. This led to the following contribution:

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2023. Explainability-based Metrics to Help Cyber Operators Find and Correct Misclassified Cyberattacks. In Proceedings of the 2023 on Explainable and Safety Bounded, Fidelitous, Machine Learning for Networking (SAFE '23). Association for Computing Machinery, New York, NY, USA, 9–15. https://doi.org/10.1145/3630050.3630177

Chapter 4 is about how ML methods offers the possibility of detecting ZDAs. While this is a possibility, this remains a very difficult task. A method based on Contrastive Learning (CL) (an unsupervised ML paradigm relying on similarity with created virtual

data) and multiple regularization techniques (to prevent overfitting and create more useful virtual data) is developed, to increase the ability to detect new unknown classes, and potentially leverage it to better detect both known and unknown classes. The following approach also presents the advantage of not being fully dependent on labels and can thus learn from larger volumes of unlabeled data, either in an initial training phase or in an incremental learning process. This led to the following contribution:

> Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2024. SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization. In the 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria. https://doi.org/10.1145/3664476.3664505

Finally, the last chapter concludes this thesis by summarizing the different contributions and providing an outlook to the intrusion detection problem. It details some difficulties that still exist in the field, as well as potential areas of research to solve these difficulties.

Chapter 2

Machine learning applied to intrusion detection

2.1 Introduction

The widespread adoption of the Internet in the late 1990s and early 2000s led to the creation and diffusion on a massive scale of the first cyberattacks. Since then, multiple methods have been developed to detect cyberattacks in advance and prevent them from causing damage. Machine Learning (ML) increased popularity as a solution for intrusion detection since the appearance of the first publicly available dataset, the 1999 DARPA dataset [1] and the subsequent KDD'99 [88] dataset.

ML methods have been researched alongside more traditional signature-based methods relying on human experts defining rules, such as Snort [141], Suricata [1] or Zeek [2]. While signature-based methods were quickly integrated into commercial solutions, ML methods have mostly remained at the stage of a research topic, with only a few commercial solutions such as Vectra [3] or Darktrace [4].

One of the motivations behind the use of ML-based intrusion detection methods is the apparent lack of detection of Zero-Day Attacks (ZDAs) by traditional signature-based approaches. ZDAs refer to previously unseen and unidentified cyberattacks that have not been cataloged by vulnerability databases, e.g., CVE [5] or CWE [6]. The term is however often used to refer more commonly to unknown attacks. With traditional signature-based methods, this is generally used to refer to attacks the rules are not designed for. For ML-based methods, this generally refers to attacks that are not present in the training

1. https://suricata.io/. Last accessed 2024-08-25.
2. https://zeek.org/. Last accessed 2024-08-25.
3. https://www.vectra.ai/. Last accessed 2024-08-25.
4. https://darktrace.com/. Last accessed 2024-08-25.
5. https://www.cve.org/. Last accessed 2024-08-25.
6. https://cwe.mitre.org/. Last accessed 2024-08-25.

data. An additional motivation is the need for frequent updates required by signature-based methods, while ML-based methods only need to be retrained with an updated dataset without particular additional work besides labeling the new part of the dataset (if required).

A study in [76] suggests that signature-based methods could actually detect ZDAs, but with limited success: a Snort rule set was able to detect 17% of attacks the rule sets were not designed for. However, the same paper considers 8% detection of ZDAs a more conservative estimate and shows that only 54% of the attacks the rules were designed for were detected, which ML methods are definitely able to beat.

In order to train ML algorithms and select the best performing one, two critical components are required: a dataset and evaluation metrics. On one hand, the dataset, preferably of high quality (size, diversity, realistic, etc.), will provide the foundation to train ML algorithms and condition their ability to detect the expected cyberattacks. On the other hand, evaluation metrics will allow comparison between multiple trained ML algorithms according to some criteria.

In this chapter, we explore the basics of ML for intrusion detection. We introduce the different paradigms in ML and their advantages and disadvantages with regard to intrusion detection. We also detail the publicly available cybersecurity datasets, their strengths and weaknesses, as well as the possible metrics that can be used to compare different Intrusion Detection Systems (IDSs). We also show how ML for IDSs being a relatively recent research topic, there is often a lack of standards for an effective comparison between approaches. We highlight the evolution of this research topic, from being focused on performance to integrating other concerns and goals. Finally, we highlight the limits and gaps in current methodologies and the areas we decided to contribute in.

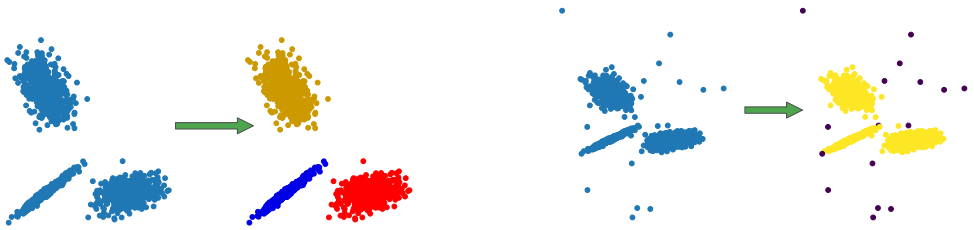
2.2 Basics of machine learning for intrusion detection

In ML, there are three main components used to solve any problem: the algorithm that will learn, the dataset that will be used by the algorithm to learn and the metrics that will be used to evaluate the algorithm.

2.2.1 Machine learning methods

ML methods can be divided in three main categories, according to their requirements of labeled data: unsupervised, semi-supervised and supervised. Unsupervised methods do not require any labels, while supervised methods only work with labels. Semi-supervised methods are more flexible and work with both labeled and unlabeled data.

In the context of intrusion detection, unsupervised methods can be separated in two main categories: clustering and anomaly detection. Clustering allows to group data points that share some common characteristics into clusters, and thus can categorize data, as shown in Figure 2.1a. It is particularly useful to discover links or similarities between data points in the absence of any information about the data. However, it relies on human investigation, or the presence of some labeled data to be able to give meaning to the different clusters. On the other hand, anomaly detection tries to learn a normal behavior that we differentiate from anomalies (also called outliers) in the data. This allows to detect ZDAs, or even faults. However, while this can detect anomalies, it is, as clustering is, unable to give meaning to the detected anomalies and is even unable to differentiate them. As shown in Figure 2.1b, different anomalies can be very different, yet are all considered in a single anomaly category, so further investigations are required.



(a) Clustering [7] (b) Anomaly detection [8]

Figure 2.1 – Unsupervised method examples.

In intrusion detection, one of the difficulties in using unsupervised approaches is that they generally operate under the assumption that the data contains mostly normal traffic, with possibly some unwanted contamination, i.e., attacks that were not detected and thus removed from the supposed normal training data. There are, however, no guarantees about the amount of anomalies present in the training data. The more these anomalies

8. The k-means algorithm was used.
8. A Local Outlier Factor (LOF) algorithm was used.

are present during training, the less effective the approach will be in actually detecting anomalies.

On the other hand, when it comes to being able to detect and differentiate between multiple cyberattacks, approaches relying on labeled data are much more efficient. Supervised methods are leveraging labeled data to learn patterns and create decision boundaries between different classes, as in Figure 2.2. It can be either binary or multi-class, depending on the number of labels existing in the data. Supervised methods, contrary to unsupervised ones, benefit from the increased presence of attacks in the data, given they are properly labeled.

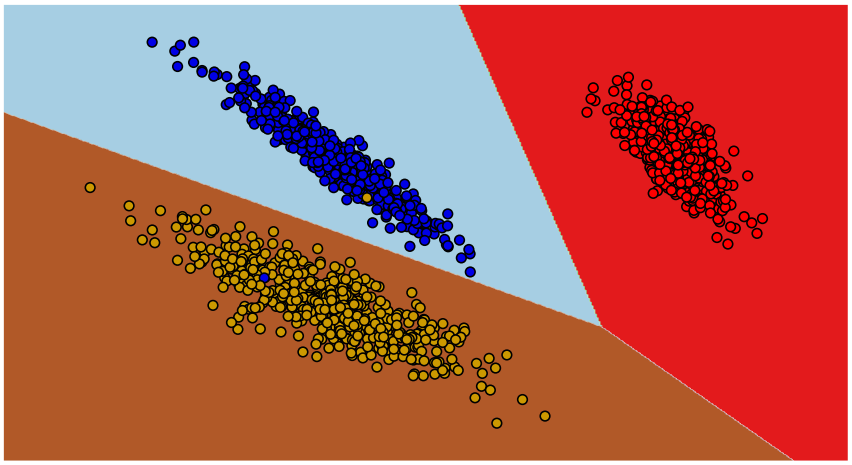


Figure 2.2 – Supervised method example, with a Logistic Regression algorithm.

Most current methods in state-of-the-art research fall in the category of supervised methods, e.g., Neural Networks (NNs), Decision Trees (DTs), Random Forests (RFs), Support Vector Machines (SVMs), both because their performance is higher, and because most currently available datasets do possess labels. Unfortunately, while supervised methods can exhibit a much better detection of known cyberattacks, they are constrained to detecting attacks they learned during training and are inherently unable to detect ZDAs.

Finally, semi-supervised methods position themselves between unsupervised and supervised methods on detection of both known cyberattacks and ZDAs. As shown in Figure 2.3, they are able to leverage unlabeled data to further refine their representation of the different classes obtained through labeled data. In the context of intrusion detection,

semi-supervised methods have been used in two different approaches. The first refers to anomaly detection by using labeled data to ensure that training data contains only normal traffic, or to control the contamination of the dataset [159, 8]. They are considered semi-supervised since they possess information about the normal class while having no information about the different attack classes. The second regroups all semi-supervised methods that rely on both labeled and unlabeled data to achieve a detection ability on known attacks similar to supervised methods while having the flexibility to recognize data that is not part of the known classes. In the latter case, approaches can either consider them as a single anomaly which might require further investigation, or are also able to differentiate them.

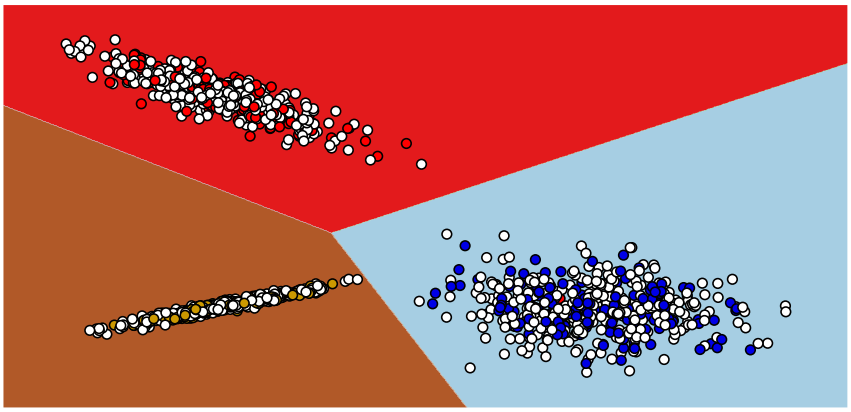


Figure 2.3 – Semi-supervised method example. In white are unlabeled data points, leveraged for training.

Among the three proposed categories, semi-supervised methods seem the most promising. They are, however, quite novel and applying them to intrusion detection still needs further research.

Finally, regardless of the chosen approach, an IDS is generally only as good as its training data allows it to, and it is thus essential to use high quality data for training.

2.2.2 Representative Datasets

Datasets in intrusion detection, contrary to many other ML applications, are difficult to create or aggregate for multiple reasons. First, they are supposed to represent a constantly

evolving environment. This implies that created datasets can quickly become obsolete as the environment changes, because traffic focuses on new applications, or new types of digital devices are now able to communicate both between themselves and with older devices. Secondly, there is a significant concern over privacy or confidentiality of the data, which essentially prevents from aggregating a dataset from real traffic for public research without anonymization of the data. Unfortunately, this process generally deletes valuable information to train IDSs. Finally, there can be changes in protocols or technologies, such as the increasing proportion of traffic that is encrypted or the switch to OPC UA [9], a unifying framework, for many Operational Technology (OT) systems. All of these reasons converge to make the creation of publicly available datasets for intrusion detection research a topic of its own.

Following the research trends in intrusion detection, datasets can be separated in different categories: network datasets [164, 123, 148], malware datasets, industrial control datasets [6, 86], Internet of Things (IoT) datasets [122, 94]. Nevertheless, in the context of intrusion detection for IoT or Federated Learning (a paradigm to collaboratively train multiple IDSs), network datasets are often used, possibly alongside a more specific dataset. In all cases, datasets represent traffic between machines, in an environment where both normal behavior and cyberattacks can be observed. However, they also possess differences, especially in their content and their feature sets. For example, network datasets generally deal with encrypted traffic and thus are composed of statistical features extracted from traffic, while many industrial datasets are composed of data directly collected from sensors.

Since this thesis was realized at the Chaire of Naval Cyberdefense, an emphasis is put on data that would reflect behavior of equipment found in a naval context, e.g., on ships, while trying to keep a general approach. Therefore, we will focus in this thesis on network datasets, because Information Technology (IT) is increasingly present in the naval context, and industrial control datasets, because most equipment present on ships, i.e., sensors, actuators, etc., are similarly used in other industrial fields. Mentions of other datasets, e.g., IoT datasets, can however be expected since intrusion detection approaches focused on other types of datasets are often relatively similar.

Network datasets

Network datasets are focused on network data, i.e., traffic using mostly TCP or UDP protocols to transmit data. TCP and UDP are used to encapsulate a variety of protocols,

9. https://opcfoundation.org/about/opc-technologies/opc-ua/

e.g., HTTP(S), SSH, FTP, ICMP, IMAP, etc. Therefore, there is a high diversity of normal behaviors, which can make the learning of what represents "a normal behavior" a difficult task.

The first datasets used by researchers were the DARPA'99 [1] and KDD'99 [88] datasets. With research on these datasets, problems surfaced, mainly about the redundancy of records in KDD'99. This led to the creation of the NSL-KDD dataset [164] where this was mitigated.

While both KDD'99 [118, 164, 43, 166] and NSL-KDD [123] have been heavily criticized [10], mostly because of their age that makes their traffic more unrealistic nowadays, they remain the most used datasets for the evaluation of IDSs. In fact, more than two thirds of the papers surveyed in [72] used one of these two datasets. Therefore, while problematic, they remain often used to compare approaches.

Since these two datasets were considered problematic, more effort has been made in the last decade to provide datasets of higher quality. In [150, 22, 149], guidelines or criteria are given to ensure quality of the simulated datasets. They can be summarized as:

— **Realistic and diverse traffic.** It includes having a variety of equipment in the network, different user behavior simulated using various protocols, as well as a diversity of cyberattacks.

— **Complete traffic.** Information that could be anonymized, such as IPs or packet content should be retained and not modified. Tools used to extract features should create a feature set as exhaustive as possible for researchers.

— **Documentation.** The creation process and necessary information should be as detailed as possible to be able to conduct proper research.

One critically important parameter is the feature set that will be used to describe the collected data. In most cases, IT network data is directly collected using tcpdump. It can then be used directly in the form of packets, or transformed using a software such as Argus [11] or CICFlowMeter [12] to provide statistical data aggregated by flows, i.e., an entire connection between two machines. Common features are generally basic information (ports, IPs, etc.) and statistics computed over the connection, such as packets per second, number of flags (ACK, SYN, URG, etc.), mean size of packets, etc. This level of granularity is often better suited to represent attacks because a combination of multiple payloads is

10. Authors of NSL-KDD themselves admit in [164] that their dataset still suffers from some of the problems raised in [118].

11. https://openargus.org/

12. https://github.com/ahlashkari/CICFlowMeter

what enables an attack, thus considering a single packet as an attack is often unsuitable. It also has the benefit of working even with encrypted traffic [137], because statistics focus on headers and not payload content. However, this obviously lowers the ability of an IDS to detect attacks that would be differentiated using packets' payloads.

Since ML-based IDSs trained on a dataset need to be evaluated, the emphasis is often made on labeled datasets that make this evaluation, and hence the comparison between IDSs, much simpler. Therefore, the accepted solution to create high quality datasets in the last decade has been to simulate traffic from as realistic as possible environments. In this way, labeling the data is relatively trivial, either by using a combination of IPs and ports used by attacking machines if they are separate from machines generating normal traffic, or by using timestamps. Although IPs and ports are often used in current signature-based IDSs, they should be removed or carefully handled in the case of ML-based IDSs to not generate any unwanted bias. Furthermore, timestamps should be remove altogether, because they can also induce a bias and do not inform on the characteristics of an attack.

In the last few years, most research on IDSs has fortunately focused on more recent datasets, such as UNSW-NB15 [123], CIC-IDS2017 or CSE-CIC-IDS2018 [148]. Even more recently, the DAPT2020 dataset [125] shows attacks methodologies much closer to current attack patterns, i.e., Advanced Persistent Threats (APTs), although the dataset is relatively small. An overview of available datasets before 2020 is provided in [60, 140], showing their strengths and weaknesses. While multiple datasets exist now, UNSW-NB15 and CIC-IDS2017 have been the most used datasets in the last few years in the field of intrusion detection. Finally, while much progress has been made, there are still existing challenges. In [72], authors show that actual coverage of known attacks is still lacking in existing datasets and that datasets are fixed and unsuited to evolve as new attacks emerge. Simulated traffic also introduces biases, and should at least be compared to real world traffic for a better simulation.

Industrial control datasets

Industrial control datasets present some similarities, but also some differences with regard to network datasets focused on IT equipment. They represent traffic between different equipment in an Industrial Control System (ICS), e.g., sensors, actuators, Programmable Logic Controllers (PLCs). There is also a variety of different protocols used, and possibly much more than for network datasets. However, communication on these systems is generally much more simple than for IT networks, and also often unencrypted. Therefore,

packet captures are generally used as-is, and data can be composed of both sensor values and traffic exchanged with specific protocols, such as Modbus.

More than a decade ago, ICSs were mostly isolated or using obscure proprietary software, so securing them from intrusion was of much lower concern. However, since the first cyberattack involving ICSs, the infamous Stuxnet [55], was publicized, interest in the field increased. Cyberattacks happening later, such as Industroyer [37] or Triton [44], further increased the incentive in building IDSs for ICSs.

Among the most used datasets in the literature [16], research often focuses on the SWaT [86] and WADI [6] datasets. While SWaT is composed of both sensor information and Modbus packets, WADI only possesses sensor information.

ICS datasets are generally not aggregated into flows. This is mainly for two reasons: traffic is unencrypted (so packet content is generally more useful than flow statistics), and sensor data can be assimilated to time series (the time dependency would be removed by aggregating flows), which can be taken advantage of by models leveraging time dependencies, such as Recurrent Neural Networks (RNNs).

Importance of dataset details and processing for proper comparisons

The dataset used for training an IDS based on ML algorithms is a very important aspect. Publicly available datasets are generally collected in different forms, separated in multiple files and can even be available in multiple versions. Such a variety often causes differences in results, not because of different approaches, but because the dataset used was different, even though it fundamentally came from the same data.

A well-known example is the NSL-KDD dataset that is available with 8 different files, representing 4 different datasets in two formats, i.e., csv and arff:

— KDDTrain+ which represents the full train set, only with binary labels for the arff file, and with multi-class labels and data point's difficulty for the csv file [13].

— KDDTrain+_20Percent which represents a subset of the train set with only 20% of the data.

— KDDTest+ which represents the full train set, only with binary labels for the arff file, and with multi-class labels and data point's difficulty for the csv file.

— KDDTest-21, which excluded records of the test set with a difficulty of 21.

13. Class difficulty comes from the NSL-KDD article [164] where authors tested 21 ML algorithms on the created datasets. Difficulty level shows how many classifiers were not able to classify correctly an instance.

Although the NSL-KDD dataset is still one of the two most used datasets, its usage often varies, which impedes comparison between approaches. The choice of a different train or test dataset necessarily leads to different expected results, and comparison to approaches using other datasets is not adequate. Furthermore, validation methodology also varies and while some approaches use one of the train sets and one of the test sets, other approaches might perform cross-validation on the train set to evaluate their approach's performance. However, one of the specificities of the NSL-KDD dataset is that some attacks are only present in the test set, which makes their detection much harder. Therefore, it is arguably counterproductive to compare approaches performing cross-validation with approaches using one of the test sets, both producing results in clearly different ranges.

A similar problem can happen with both the UNSW-NB15 and CIC-IDS2017 datasets that are much bigger. In both cases, subsets of the datasets are sometimes used instead of the full datasets. It might be done to reduce computation time and costs or it might be done by removing classes to better comply with stated objectives, e.g., only focus on Denial of Service (DoS) attacks in CIC-IDS2017. This is also more common for the UNSW-NB15 dataset that directly provides a subset of both training and test sets.

Fortunately, such differences in dataset usage tend to disappear as research progresses and did not happen as much in the last few years as it did before. For the purpose of this thesis, the NSL-KDD, UNSW-NB15, CIC-IDS2017, DAPT2020 and WADI datasets were used. In Table 2.1, details about classes, and total number of samples are available to ensure proper usage of the datasets. Still, separating datasets into train and test sets randomly could lead to different class proportions in the two sets, especially for rare attacks. Therefore, the best method is to perform a stratified split that respects class proportions.

Finally, while the dataset and how it is used has a huge impact on performance, this performance is evaluated using metrics often taken from other ML applications. This can create differences because intrusion detection datasets are generally much more imbalanced than, for example, image datasets. Therefore, the choice of the metrics and how to average them is often more crucial in intrusion detection problems.

2.2.3 Metrics for intrusion detection

In order to evaluate the performance of ML-based IDSs, metrics are required. While it is possible to evaluate unsupervised ML models with scores, e.g., silhouette, it is generally

Table 2.1: Datasets details

Dataset	Number of instances per class	Total
UNSW-NB15[a]	Normal: 2218761, Generic: 215481, Exploits: 44525, Fuzzers: 24246, DoS: 16353, Reconnaissance: 13987, Analysis: 2677, Backdoor: 2329, Shellcode: 1511, Worms: 174	2540047
CIC-IDS2017[b]	Benign: 2273097, DoS Hulk: 231073, Portscan: 158930, DDoS: 128027, DoS GoldenEye: 10293, FTP-Patator: 7938, SSH-Patator: 5897, DoS Slowloris: 5796, DoS Slowhttptest: 5499, Botnet: 1966, Web Attack Brute Force: 1507, Web Attack XSS[1]: 652, Infiltration: 36, Web Attack SQL Injection: 21, Heartbleed: 11	2830743
DAPT2020[c]	**Normal**: 63712 **Reconnaissance*** – Network Scan: 7614, Account Discovery: 124, Directory BruteForce: 1503, Web Vulnerability Scan: 2574, Account BruteForce: 94 **Establish Foothold*** – SQL Injection: 55, Directory Bruteforce: 8467, Account Bruteforce: 47, Account Discovery: 12, Malware Download: 2, Network Scan: 2, CSRF[2]: 7, Command Injection: 12 **Lateral Movement*** – Network Scan: 117, Backdoor: 20, Account Discovery: 2272, SQL Injection: 29, Privilege Escalation: 13 **Data Exfiltration*** – Network Scan: 9, Data Exfiltration: 6	86691
WADI[d]	Normal: 947347, Attack_3–4: 1742, Attack_10: 1620, Attack_1: 1502, Attack_5: 852, Attack_6: 808, Attack_9: 700, Attack_8: 672, Attack_7: 632, Attack_2: 592, Attack_13: 578, Attack_14: 204, Attack_15: 89	957338

* These represent DAPT2020's four attack stages. Not in bold are each stage's activities. Stages are representative of different steps of an APT, while activities are different cyberattacks used in a stage.
[1] Cross-Site Scripting.
[2] Cross-Site Request Forgery.

[a] Downloaded from https://research.unsw.edu.au/projects/unsw-nb15-dataset
[b] Downloaded from https://www.kaggle.com/cicdataset/cicids2017
[c] Downloaded from https://www.kaggle.com/datasets/sowmyamyneni/dapt2020
[d] Downloaded via the form in https://itrust.sutd.edu.sg/itrust-labs_datasets/

more objective to rely on metrics used in supervised ML. Furthermore, most datasets used in current research are now labeled, which also increases the incentive to evaluate using these metrics from supervised ML.

One important aspect influencing the definition and use of metrics is the formulation of the problem as a binary problem or as a multi-class problem. In either case, the most complete representation of an IDS's performance is the full confusion matrix (Table 2.2 for the binary case example), which can be extended from the binary to the multi-class case.

Table 2.2: Binary confusion matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN)	**Positive (P)**
	Negative	False Positive (FP)	True Negative (TN)	**Negative (N)**
		Predicted Positive (PP)	**Predicted Negative (PN)**	**Total**

Metrics commonly used in ML, e.g., Accuracy (Equation (2.1)), Precision (Equation (2.2)), Recall (Equation (2.3)) and F1-Score (Equation (2.4)), are generally derived from the confusion matrix.

Some of these metrics are more or less useful depending on how the problem was formulated. For example, the Area Under the ROC Curve (AUC), that measures the likelihood of a model to be more confident in a correct prediction than an incorrect one, is one of the more informative evaluation metrics for binary problems, whereas it is not appropriate to use for multi-class problems because it relies on binary decisions. Additionally, it is ill-suited to cases when there is a disparity in the cost of false positives and false negatives. In intrusion detection, false alarms are generally considered much less costly than missed attacks, unless they happen too often, in which case the situation is no longer manageable by human operators and too many false alarms becomes very costly.

Some of these metrics have also been used with a different name in intrusion detection to better represent what they mean in the context of IDSs: Recall (or True Positive Rate) is often called Detection Rate and represents the proportion of attacks that are correctly detected, while False Positive Rate (Equation (2.5)) is often called False Alarm Rate, and

as its name suggests, represents the proportion of false alarms.

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.1}$$

$$Precision = \frac{TP}{PP} \tag{2.2}$$

$$Recall = \frac{TP}{P} \tag{2.3}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.4}$$

$$False\ Positive\ Rate = \frac{FP}{N} \tag{2.5}$$

While these metrics are extensively used in different ML applications, there are significant drawbacks to using most of them for intrusion detection. One of the major pitfalls described in [14] is caused by the high imbalance between classes in intrusion detection problems and was further highlighted in [154, 64, 83]. This imbalance will silence signal coming from less represented classes and might create undue confidence in the IDS. This is also further accentuated in multi-class intrusion detection where some attacks are very rare. Unfortunately, while most metrics are not adapted to handle a high imbalance, which is detrimental to the evaluation, this high imbalance is an inherent characteristic of realistic intrusion detection problems. Therefore, there are two solutions to this problem: find more adapted metrics or balance the training dataset. In either case, the test set should conserve an imbalance representative of real world traffic, and thus using metrics resistant to imbalance is a must.

Furthermore, when the problem is formulated as a multi-class problem, the impact of the imbalance increases. For every metric, or even the confusion matrix (when normalized), there are pitfalls and low performance can possibly be hidden. Because presenting results with too many classes can quickly become difficult, common metrics are often averaged following one of two methods: micro-averaging or macro-averaging. Micro-averaging factors in class proportion to give more importance to classes that are more prevalent, which can hide low performance on rare classes. Macro-averaging, on the other hand, averages without class proportions which can result in giving too much importance to really rare events. Considering four classes A, B, C and D with respectively 10000, 1000,

20 and 10 instances, macro-averaging results of a given metric f (results using metric f for class A are denoted f_A) can be obtained as in Equation (2.6), while micro-averaging results can be obtained as in Equation (2.7). While the confusion matrix does not hide or overestimate performance, it is often normalized, which in this case, can easily hide misclassification of overrepresented classes.

$$macro\_f = \frac{f_A + f_B + f_C + f_D}{4} \tag{2.6}$$

$$micro\_f = \frac{10000 \times f_A + 1000 \times f_B + 20 \times f_C + 10 \times f_D}{10000 + 1000 + 20 + 10} \tag{2.7}$$

Table 2.3a shows the confusion matrix of a toy example with four classes that can be used to illustrate how the effect of imbalance is further amplified in the case of intrusion detection problems. In this context, Class A could represent Normal traffic, Class B could represent an attack generating a lot of traffic, such as DoS, and Class C and D could be rarer attacks, or attacks generating a low amount of traffic, e.g., Heartbleed, SQL injections, infiltration, exfiltration, etc.

When used to present results, confusion matrices are generally normalized using class support, as in Table 2.3b. By using class support to normalize, the given confusion matrix actually focuses on detection rate. Given the imbalance in the data, results show the ability to find all attacks, at the risk of having too many false alarms. For example, Class D that is always detected shows a detection rate of 1. However, what is not properly shown is that because of the higher number of Class A incorrectly classified as Class D, the proportion of Class D predictions that are correct is low. In this case, only one in eleven Class D predictions is correct.

On the contrary, if we want to focus more on false alarm rate, it is better to normalize using predictions' support, as in Table 2.3c. In this case, false alarms will be better highlighted, but results might not show if rarer attacks are missed. For example, Class C shows a performance of 1 in the diagonal, because all Class C predictions were correct. However, half of Class C samples were missed and classified as class A, which is not properly shown.

While it was shown that the choice about the normalization impacts which information is highlighted by confusion matrices, they are more rarely used than other metrics, mainly because the visibility of information decreases as the number of classes increases. As in other application domains, the most encountered metrics are Accuracy, Precision, Recall

(a) Example of a multi-class confusion matrix

		Predicted class			
		Class A	**Class B**	**Class C**	**Class D**
Actual class	**Class A**	9800	100	0	100
	Class B	0	1000	0	0
	Class C	10	0	10	0
	Class D	0	0	0	10

(b) Multi-class confusion matrix normalized by actual class

	Class A	**Class B**	**Class C**	**Class D**
Class A	0.98	0.01	0	0.01
Class B	0	1	0	0
Class C	0.5	0	0.5	0
Class D	0	0	0	1

(c) Multi-class confusion matrix normalized by predicted class

Class A	**Class B**	**Class C**	**Class D**
0.998	0.09	0	0.91
0	0.91	0	0
0.001	0	1	0
0.001	0	0	0.09

Table 2.3 – Confusion matrices on a 4-class toy example.

and F1-Score. These metrics are often micro-averaged to give single values comprising all classes for each metric, which is fine with balanced datasets. However, with imbalanced datasets, results are heavily biased towards showing performance on the most prevalent classes. Table 2.4 illustrates how micro-averaged or macro-averaged metrics can, respectively, hide low performance or overestimate it in case of imbalanced datasets. In this case, low detection of Class C as well as the high amount of false alarms for Class D are almost completely hidden with micro-averaged metrics. On the contrary, they are highlighted too much with macro-averaged metrics. This raises questions about which method to use, and might be chosen depending on the goal pursued. If the objective is to focus on better performance on rare classes, it might be advisable to perform macro-averaging.

Table 2.4: Micro and Macro-averaged metrics, Accuracy = 0.981

	Precision	**Recall**	**F1-Score**
Micro-averaged	0.990	0.981	0.984
Macro-averaged	0.750	0.870	0.694

Unfortunately, research has not yet settled on a comprehensive metric to handle imbalance in the data. In 2017, [38] advised to use Matthews Correlation Coefficient (MCC) (Equation (2.8)) to account for imbalance. It is argued that since MCC considers TP, TN, FP and FN, it is the most complete metric and can account for imbalance. On the contrary, it is shown in [189] that although MCC might be more complete, it is not very resistant to imbalance. In this case, the geometric mean of TPR and TNR or the Bookmaker Informedness (BI) (Equation (2.9)) might be better adapted. It is also argued that BI, similarly to MCC, captures information on both the positive and negative spectrum while simultaneously resisting to imbalance, and is therefore a better metric. However, these conclusions stand only for binary problems and are much less accurate in multi-class problems.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.8}$$

$$BI = \frac{TP}{P} + \frac{TN}{N} - 1 \tag{2.9}$$

Since there is no perfect metric currently available, the generally accepted solution is to represent results using multiple metrics, as suggested in [154]. Better yet, a metric or metrics adapted to the cybersecurity landscape should be developed to properly handle the imbalanced data in intrusion detection problems, as well as the different costs (human costs for investigation and financial costs for discontinued services and remediation), related to missing attacks or raising too many false alarms.

Finally, one of the major deficiencies of current metrics is that they are generic, and cannot account for the importance of different classes. In cybersecurity, different attacks are undoubtedly not equal, and detecting a currently ongoing exfiltration of information is much more critical than detecting a port scan.

2.3 Evolution of ML methods and their application to IDSs

ML methods applied to intrusion detection can be divided into two main categories depending on the objective: globally separate attacks from normal traffic, or be able to categorize traffic into different classes comprising normal traffic and different cyberattacks. In the former case, anomaly detection approaches or binary classification approaches are employed. In the latter case, multi-class classification approaches are used. From an operational context, anomaly detection tries to inform of a suspicious behavior, while relegating the task of investigating what is happening to human operators. This could potentially help find ZDAs, or even faults in some equipment, but requires extensive investigation efforts from human operators. On the contrary, multi-class classification approaches raise alerts when identifying known malicious behavior. This can greatly decrease the workload of human operators, since they only have to verify that the alert raised was correct. For the purpose of this thesis, more emphasis will be put on multi-class classification approaches, because we consider that using methods based on anomaly detection would lead to unmanageable workloads. Indeed, human operators are generally unable to handle all alerts considering real-time constraints (an alert about an ongoing cyberattack has to be verified quickly), so limiting the scope of investigations by providing information about the potential cause of an alert is preferable.

In anomaly detection like in multi-class classification, there have been multiple methods employed over the years, which can be categorized into "shallow" models and Deep Learning (DL) models. "Shallow" models have been used since research started on behavioral approaches for intrusion detection. The difference between the two comes from the fact that "shallow" models do not change the representation on the data, while DL methods will create new latent representations of the data. These new representations are potentially richer, and features better disentangled, which can improve detection performance. Therefore, DL models perform a kind of automated feature engineering, removing the need to do it manually.

2.3.1 Anomaly Detection

Anomaly detection is used in many different applications, from detecting illicit financial transactions, to identifying faults in some equipment. In [142], an anomaly is defined

as "an observation that deviates considerably from some concept of normality". In the context of intrusion detection, these approaches are useful when only normal traffic is defined and we want to be able to differentiate it from abnormal behavior, corresponding to potential cyberattacks. Although training data can be "corrupted" with a small prevalence of some attacks, it is generally assumed that training data is composed only of normal traffic.

While methods employed in binary-classification are generally the same as multi-class classification, the evolution of unsupervised methods also followed a similar trend to that of multi-class classification methods. The first methods used were shallow models, e.g., One-Class SVM, Isolation Forests, Local Outlier Factor, and can prove useful in a context where no information is available about classes and their low computation requirements makes them highly scalable IDSs [135].

DL methods were then increasingly used, with Auto-Encoders (AEs) being the most used method. They are specific NNs that are built with an encoder part that maps data to a latent space of lower dimension, and tries to recreate the original data through a decoder part. The reconstruction difference is then used to detect anomalies. Although these models are more efficient at detecting cyberattacks than their "shallow" counterparts [125], they are more often used as feature reduction techniques [180, 108] because of their ability to create lower dimensional representations while losing less information than other techniques like Principal Component Analysis.

Finally, generative models were developed to better model the underlying distribution of the data, with Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs). In intrusion detection, both techniques model the distribution of normal traffic, and while VAEs generally make assumptions about the distribution, such as being a mixture of gaussians, GANs generally do not make any assumption. While these techniques show a higher detection rate of anomalies than previous approaches [130, 178, 129, 46], they are also often employed because of their ability to model the underlying distribution of the data to generate synthetic traffic. As will be shown in Section 2.4.1, these techniques can be used to solve the imbalance problem.

However, as already argued in this thesis, we believe anomaly detection methods are unsuitable to help in decision-making because too much information is lacking and their false alarm rate is notably higher than multi-class classification methods [81].

2.3.2 Multi-class classification methods

As mentioned previously, we believe that multi-class classification approaches present more advantages that anomaly detection and binary classification methods, so we will focus on multi-class classification approaches in this thesis.

"Shallow" Learning

"Shallow" models refer to models that are not part of the DL category, e.g., tree-based classifiers, SVMs, regression methods, etc. While some models are relatively simple in how they work, they can still achieve a high detection rate. They are also generally much faster to train, and most also scale well as data increases. However, it is generally accepted that, as data and complexity increases, they tend to perform worse than DL techniques.

The most often encountered "shallow" learning models are:

— Tree-based classifiers. Decision trees [30] and Random Forests [29] are the most common. More advanced tree-based models using Boosting such as XGBoost [35] are sometimes encountered.
— Support Vector Machines [41].
— Regression methods, e.g., linear regression, logistic regression [161].
— k-Nearest Neighbors [42].
— Bayesian approaches, e.g., Naïve Bayes [93].
— Clustering methods, e.g., k-means [115], DBSCAN [53].

In the context of IDSs In 2010, [95] showed that approaches mainly centered around the previously mentioned approaches. Furthermore, it showed that ensemble methods (combinations of multiple ML models) were already used and often better than single classifiers. Finally, it already highlighted concerns that still remain to this day: handling of large traffic volumes for real time analysis and detection of rare attacks. While there has not been much improvements in the ML models employed, research tried to enhance their performance with careful data pre-processing, feature selection [102, 80, 165, 24, 5] or class balancing with approaches [165] such a SMOTE [34] or ADASYN [68].

While most of these methods tend to either not perform well or not scale well as complexity and size increases, tree-based classifiers, and particularly those based on tree ensembles, can often achieve performance similar to most DL approaches on existing public datasets.

Deep Learning

Although research on DL techniques started much earlier, it was actually coined as such in 2015 in [98]. These techniques are based on the concept of multiple neuron layers separated by non-linear activation functions, where the model is trained using back-propagation. Breakthroughs around the mid-2010s, mainly in image processing applications with Convolutional Neural Networks (CNNs), really popularized the research topic.

In the last decade, with the rapid growth in hardware capabilities, ML has observed many breakthroughs in multiple domains. This enabled the usage of more complex models, among which DL models are at the forefront. The amount of internal parameters in different DL models increased from the tens of thousands, to the millions or even billions. Depending on various factors and problem formulations, different ML models are more often employed, e.g., CNNs for image recognition, transformers for language, etc. These models are however focused on tasks that can be put into two main categories: binary or multi-class classification, and anomaly detection.

In the context of IDSs While they were already used before, the late 2010s also marked a shift from regular ML (shallow models) to more DL methods. In [74], authors provided a taxonomy of existing methods to build IDSs where they asserted that DL methods reach a higher performance than shallow models. In [56], authors expanded on DL for IDSs and showed that the most used DL methods are Deep Neural Networks (DNNs), RNNs [19], CNNs and restricted Boltzmann machines.

At the same time, because of the accepted limits of existing datasets, more research tried to evaluate approaches on multiple datasets, such as in [170, 54]. This trend is still continuing with most research papers now using at least two to three datasets for evaluation purposes.

While CNNs have revolutionized computer vision, it provides less benefits in the context of intrusion detection. Convolutions can capture higher level information but there is no spatial dependency between traffic features. Although 2D CNNs could be used to take into account temporal information, RNNs are naturally more adapted and probably a better choice in this case. Nevertheless, 1D CNNs have been used often in intrusion detection [183, 186, 70] and show a relatively high performance. Since they are able to perform as well as other methods while using convolutions on often unrelated features, this brings back to the question of the adequacy of the data and metrics used for evaluation.

While some variations exist, the core concept of being able to forget or remember

information, potentially long-term, remain the same. As such, Long-Short Term Memorys (LSTMs) [73] are particularly interesting in intrusion detection applications where data takes the form of time series, e.g., in ICS applications with sensor data [10]. However, although often employed on data in the form of flows, they does not seem to bring any benefits compared to more regular NNs [40]. Another potential use of LSTM networks is on data in the form of packets, where temporal dependency does exist, although research on this topic remains sparse.

More recently, with the same objective, approaches leveraging transformers, architectures based on the concept of attention, were tested in intrusion detection [174, 108], including on data in the form of packets [66]. Approaches using CNN-LSTM or CNN-BiLSTM were also used for the same purpose [157, 84, 2, 65, 167, 176, 61]. However, as argued previously, the use of CNN, LSTM, attention and transformers, or a model combining multiple concepts, still remain weakly adapted to intrusion detection when data takes the form of flows. While performance is similar or sometimes better than other approaches, this is often due to a bias in the methodology used, e.g., a specific usage of the dataset, or simply because of the inadequacy of existing metrics for heavily imbalanced datasets.

While different methods have been used in the context of intrusion, there has rarely been models that are able to leverage characteristics of the intrusion detection problems or even to leverage network or cybersecurity knowledge.

Integration of network topology

When using ML for intrusion detection, it is important to remove information that might potentially bias the ML model towards a specific behavior or make the detection task too simple. As such, features such as IP addresses, sometimes also ports, are often removed from the feature set, because they might hinder the ability of models to generalize. Datasets are often created with specific machines performing the role of attackers, which leads IP addresses to be a sufficient descriptor to predict a cyberattack in many cases. A training ML model would then only focus on this particular feature instead of learning to recognize different attacks. However, while IP addresses might induce a strong bias, they are still relevant information. For example, a machine in a local network communicating with a machine outside the network might be a strong signal for suspicious behavior.

Similarly, port number might be important information. For example, an IDS raising an alert of an unauthorized SSH connection attempt when port 22 is not used should

obviously be incorrect. Some features are thus in an awkward spot where they might bias or prevent the IDS from learning correctly, but still represent important information.

Graph Neural Networks (GNNs) provide a possibility to take into account network topology without biasing the IDS. Besides their ability to consider network topology, GNNs present various advantages depending on how they are constructed:

— Attributed graphs are the most simple GNNs useful in intrusion detection. Nodes or edges can possess information about machine state or communications, which could be richer than only flow features.

— Spatio-temporal graphs can preserve temporal information, which allows for a better detection of sudden changes in behavior, as in [27].

— Dynamic graphs are especially useful to represent networks that evolve as new machines are added or new communications are initiated.

GNNs are generally trained by updating the embedding of nodes, considering the information contained in edges, in attributes, and using a node's neighborhood. While this can provide rich representations, this can also be difficult to scale. As such, an approach restraining the neighborhood of nodes was developed [110]. However, while this shows improvements, it is still unclear if GNNs would be able to scale well enough in real-world applications.

Most of the research on GNNs in intrusion detection has focused on static graphs [32, 134, 128, 185, 63, 7] where the topology is fixed during training and the same is used for testing. While static graphs are useful to understand past attacks behavior, they are unsuitable for real-time applications where topology might evolve. As such, dynamic graphs where the topology can also evolve will have a better potential to detect attacks in real time [188]. In [100], both static and dynamic graphs (topology is different between training and testing) are used and dynamic graphs show a similar performance to static graphs. However, more research needs to be done on the topic to ensure the ability of dynamic graphs to change along with topology while retaining a high detection rate.

2.4 Increase trust in the IDS

Various factors can influence trust in the IDS. Two of the most obvious factors are the positive expectations of the model, as well as the ability to understand the decision [168, 169]. Positive expectations of the model in the context of IDSs is influenced by its ability to correctly detect attacks, while not raising too many false alarms. On the other hand,

the ability of a human to understand the decision greatly depends on the ML model used. Some models are already understandable, e.g., Decision Trees, but most are difficult or impossible to understand, e.g., Neural Networks, ensemble methods, etc.

Most of the work to improve performance of IDSs in the context of intrusion detection has focused on improving the ML models used or trying to remediate to some difficulties encountered in intrusion detection problems, e.g., the imbalance in the data.

2.4.1 Improve performance by reducing the imbalance problem

Intrusion detection is a research topic where classes are notably imbalanced. While normal traffic is predominant, the imbalance increases greatly when considering multi-class problems since many cyberattacks do not produce a lot of traffic and are thus very rare. Most ML models generally struggle to properly recognize rare classes because their training mechanisms are more influenced by the higher amount of errors made on more prevalent classes. As such, techniques to reduce this imbalance, whether partially or completely, have been used to build IDSs. It can be as simple as random undersampling or oversampling, or undersampling of the majority class, as in [66]. However, it can also be done through other more advanced techniques.

One such technique is SMOTE [34] that will synthesize samples of underrepresented classes. In [68], authors try to improve upon SMOTE by proposing a method using a weighted distribution to focus more on synthesizing samples of classes that are difficult to learn. Both of these techniques follow a simple generation method: given a sample and its neighbor, it will generate a new sample between both points using some λ (randomly chosen between 0 and 1) parameter.

Both of these methods have been used in [84, 57, 182, 162, 167, 183, 108, 163, 61] and resulted in an increase in performance. In [15], various combinations of random oversampling and undersampling, SMOTE and ADASYN, are used and show improvements over regular training.

Additionally, probabilistic estimators used in anomaly detection are able to generate data from the distributions they learned. Two such examples are VAEs and GANs. VAEs are auto-encoders that learn the distribution of the latent space obtained through encoding and reconstruction of the original data. There are generally assumptions made on this distribution, such as being a mixture of gaussian distributions. This enables them to leverage the learned distribution to generate new samples, but has to follow the assumptions made. GANs were introduced in [62] as a new method to estimate generative models via

an adversarial process. They are trained by simultaneously training two models: a generator G and a discriminator D. G will try to generate samples as close as possible to the distribution of the data, while D will try to distinguish between real data and data generated by G. In practice, the generator will be a function that maps a random variable from a simple distribution, e.g., the uniform or gaussian distribution, to the data's distribution. Contrary to VAEs, GANs generally do not make assumptions about the distribution they model. While this allows for possibly better performance, GANs are notably harder to train.

In [78, 45, 132], GANs are employed to balance datasets, whereas in [107, 171], Wasserstein GANs (WGANs) are used. In many cases, GAN or WGAN approaches are compared to previous methods such as SMOTE and ADASYN and show improvements in performance. In [78], it is suggested that GANs are better than previous methods to generate samples because they are better at generating samples of classes that are quite close. Finally, in [179], VAEs are used with WGANs to generate samples. In [48], authors suggest that GANs are therefore more effective in generating samples than techniques like SMOTE or ADASYN because they also learn the latent representation of classes, which allows them to create more consistent samples.

2.4.2 Understanding the decision process

In order to understand how an IDS built using ML models comes to a decision, there are two possibilities: the model in itself is understandable, or external tools such as Explainable AI (XAI) are used.

Transparent models

The simplest solution to understand an IDS is to use ML techniques that are transparent to a human, i.e., their working process is easily understandable. Among the possible models, we can find:

— **Linear Regressions.** They are basically using weights to learn a function of the form $W^T \cdot X + b$.

— **Decision Trees.** They are learning rules that allow to differentiate between samples, e.g., $source\ port = 22$.

— **k-Nearest Neighbors.** They are simply classifying using the closest known samples.

— **Bayesian models.** They are based on Bayesian theory and how they work can be understood by humans.

More models based on rules or simple mathematical computations can also be included, e.g., fuzzy-rule systems, generalized additive models. However, it is important to note that any model that is transparent can also become difficult to understand as it is complexified. For example, a linear regression with thousands of weights, or a decision tree with thousands of nodes are difficult to comprehend.

In [160, 25], tree methods are used in order to build an IDS that can explain its decisions by using the trees' splitting rules. While this is doable because tree methods are among the better performing transparent models, their complexity increases with the size of training data and can make the decision process too complex to be understandable. Limiting their size is possible but will lead to decreased performance, which is also unwelcome.

Explainability

XAI is a research topic that gained traction relatively recently, mainly because of the usage of ML models in various high stake real world applications, such as autonomous driving, as well as the legislative pressure on the usage of ML.

XAI methods can be divided following three characteristics [105, 3, 11, 146]:

— **Local vs. global.** A local method will try to explain a prediction of a single sample, how a model reached its decision, while a global method will try to explain the behavior of the model for all possible data.

— **Model-agnostic vs. -specific.** This refers to the ability of the method to work with any kind of ML model.

— **Intrinsic vs. post-hoc.** Intrinsic methods integrates the explanation process into the ML model, while post-hoc methods are used after a model is built and trained.

Depending on the goal, there are also different ways to present the information provided through an explanation [11]: textual explanations, explanations by example or counterfactual, explanations by simplification and feature importance. Textual explanations, as the name suggests, provide textual responses as to explain a decision. Explanations by example are leveraging examples or counterfactuals to explain why a decision has been made as it has. They show similar examples that were classified in the same way to support the decision, while counterfactuals show what should be changed for the decision to be different. Explanations by simplifications try to build an understandable ML model,

e.g., a DT, a Linear Regression (LR), that will mimic the global or local behavior of the model to explain. Finally, feature relevance methods are possibly the most researched and used XAI methods, for their ease of use with different kinds of data.

As has been the case with DL, the privileged application domains for XAI have been images and text. As such, many of the methods are not necessarily applicable in the context of intrusion detection. For example, a few popular feature relevance methods [152, 153, 147] are built for and only really usable with images. Counterfactuals and explanations by example are usable for any kind of data, but are much easier to understand when applied to images.

For intrusion detection, much of the work has focused on two of the most popular model-agnostic methods: LIME [138] and SHAP [112]. Both of these methods provide interpretable local approximations of the model to explain.

LIME stands for Local Interpretable Model-agnostic Explanations and provides explanations by creating an interpretable model, chosen in [138] to be a sparse linear regression. This interpretable model learns with locally created interpretable representations to become locally faithful, i.e., it will locally make the same predictions as the classifier to explain. These interpretable representations are representations of features that make sense to a human. In more formal terms, LIME defines an explanation as a model $g \in G$, with G being the class of interpretable models, and g representing the presence or absence of individual interpretable components. Additionally, a measure of the explanation complexity $\Omega(g)$ is used to ensure that the interpretable model g does not become too complex. With the explained classifier being denoted as f, $\pi_x(z)$ being a proximity measure between x and z, we have $\mathcal{L}(f, g, \pi_x)$ as a measure of how unfaithful the created interpretable model g is with regard to the explained classifier f. The goal is then to find the most faithful g with low enough complexity, as described in Equation (2.10).

$$\underset{g \in G}{\operatorname{argmin}}\, \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.10}$$

SHAP stands for SHapley Additive exPlanations and is inspired from Shapley values in cooperative game theory. In [112], the authors argue that multiple XAI methods relying on local approximation of the classifier to explain can be unified and described with additive feature attribution methods. Given a classifier f, a mapping function h that maps from original features x to interpretable features x' and vice-versa, $x = h_x(x')$, an explanation

g and a number M of interpretable features, we have z_i being the absence or presence of interpretable feature i and have to find Shapley values $\phi_i, i \in \{0, ..., M\}$ such that:

$$g(z) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i \tag{2.11}$$

To follow properties of local accuracy [14], missingness [15] and consistency [16], there is only a single explanation model g with ϕ_i being computed as in Equation (2.12).

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)] \tag{2.12}$$

where $x' = h_x(x)$, $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

For intrusion detection tasks, LIME has been used in [175, 116, 77] and SHAP in [173, 116, 124, 17, 182]. While this shows which kind of explanation can be expected from these methods, few has been done to actually evaluate the quality of explanations or if they would be useful for the end-users in an operational context.

2.5 Complement signature-based approaches with detection of zero-day attacks

One of the major limiting factors for the widespread adoption of ML methods for IDSs is that existing signature-based methods work relatively well and significant work is still put into improving these methods. While ML methods have some advantages, e.g., more flexibility, possibly better performance, the ability to detect ZDAs, they also have disadvantages, e.g., more false alarms, often being impossible to understand. Therefore, it might be more productive to conceive hybrid systems, with IDSs based on both signature

14. The explanation model and original model have the same output.
15. Features missing in the original input should have no importance in the explanation.
16. If the model changes such that a feature becomes more important for a prediction, it's importance in the explanation should also increase.

and ML approaches. As such, focusing research in ML methods where signature-based methods are lacking is obviously more fruitful.

2.5.1 Zero-day attack detection

A major benefit of IDSs based on ML models is their potential ability to leverage data to detect ZDAs. There are three very different methods to do so:

— By performing anomaly detection. In this case, there are two possibilities: binary classification or Out-of-Distribution (OOD) detection. Given a labeled dataset composed of different attacks and normal traffic, performance may vary. In the former case, detection performance on known attacks will be higher, at the cost of a lower detection performance on ZDAs. In the latter case, detection performance will be similar for both known and ZDAs.

— By using multi-class unsupervised methods, such as clustering. While anomaly detection simply separates between normal and anomalies, multi-class unsupervised methods allow to separate into more than two groups.

— By using Open-Set [144] or Open-World [18] learning. Both approaches fundamentally rely on supervised methods to learn to differentiate normal traffic and known attacks, and are thus able to perform multi-class classification. In the former case, the approach can "reject" unknown samples and consider them as anomalies, because they are unable to attribute them to known classes with sufficient confidence. In the latter case, the approach goes further by trying to differentiate the detected anomalies to classify them into new classes.

Regardless of the choice, methods employed for the detection of ZDAs are different from those employed for IDSs in general. For the purpose of this thesis, since we strongly believe that IDSs should operate in a multi-class setting, we will only consider Open-Set Learning (OSL) and Open-World Learning (OWL).

In intrusion detection, much of the work about detection of new classes has focused on OSL. In [143, 92, 39, 79], it is shown that using OSL methods can lead to detection of unknown classes with a rate ranging between 20% and sometimes up to 90%. However, these approaches only consider a single anomaly class, and sometimes are tested on relatively small datasets. Furthermore, they are always tested by leaving out one attack of the dataset, which restricts the distribution of unknown attacks to that of only a single class. As such, this is unclear if and how these methods would scale when considering multiple different new classes, even if considering them as a single anomaly class.

Contrastive Learning for zero-day attacks

Contrastive Learning (CL) has gained popularity in recent years by decreasing the difference in performance between supervised and unsupervised DL on image applications [82]. CL is based on NNs that will learn a new representation of the data, to be used in downstream tasks. It typically relies on self-supervised learning, which is based on augmentations. Augmentations are transformations of the data that retain the semantic information, e.g., color shifts in images still represent the same object. These augmentations are used to create samples that share the same semantic information and are called positives. Other samples or augmentations of other samples are considered as negatives samples. The goal is then to learn a representation space where anchors are close to their positives and far away from their negatives. However, augmentations are much more difficult to define in the case of IDS datasets where modifications of some information, e.g., ports and protocols, can be too complex or even counterproductive because they do not retain the semantic information.

An important component of CL that requires careful consideration is how anchors, positives and negatives are chosen for training. Anchors are data samples that will serve as reference to select both positives and negatives. Positives are data points, possibly created from the anchors through some transformation, that share semantic information with the anchors. Negatives are other data samples that generally do not share semantic information with the anchor. The contrastive objective is then to update NNs' latent space such that positives are brought closer to anchor, and negatives are pushed further away from anchors in the latent space.

In a typical contrastive training, the objective is to create a latent space in which anchors are brought closer to positive samples that should contain the same semantic information as anchors, and further from negative samples. Given a batch I of samples and z_i being the output of x_i, the training loss can be expressed as:

$$\mathcal{L}_{contrastive} = -\sum_{i \in I} log \frac{exp(z_i \cdot z_{positive}/\tau)}{\sum_{a \in Negatives} exp(z_i \cdot z_a/\tau)} \tag{2.13}$$

In Equation (2.13), i is thus the chosen anchor. For both positives and negatives, $z_i \cdot z_j$ represents the distance between the two chosen samples. Therefore, both the distance and τ, a temperature parameter that influences the amplitude of the loss, are hyper-parameters. While the chosen distance allows to define what is spatially closer or

further away, the temperature parameter allows to define the force with which positives are brought closer and negatives pushed further apart.

In a self-supervised training, a positive is generally obtained by "augmenting" the anchor. Taking images for example, this could be done by cropping or rotating the anchor. Negatives are generally composed of all samples except the anchor. However, in intrusion detection, the creation of positives represents a substantial obstacle: how to change a flow or a packet while conserving its semantic meaning?

A solution to this problem is provided in [90] where a loss function using supervised CL is developed. Instead of "augmenting" anchors, positives and negatives can be selected using labels. An additional benefit of using this method is that positives encompass multiple instances of the same class and are not limited to augmentations of the anchors.

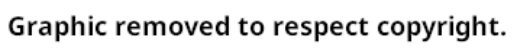


Figure 2.4 – Self-Supervised and Supervised Contrastive Learning. Credit to [90].

Still, CL appears to be a promising solution to detect ZDAs, although it is a relatively new ML topic for cybersecurity, and research using CL in intrusion detection is relatively scarce. In [181], it is shown that using a contrastive loss alongside a more common cross-entropy loss can achieve state-of-the-art results in intrusion detection. Unfortunately, the detection of new classes is not addressed. In [111], CL is used to perform intrusion detection and new class detection, where it shows similar performance ranges to [143, 92, 39, 79].

2.6 Conclusion

In a context in which IDSs are unable to act autonomously for both detection and defense, it will inevitably be used to help decision-making of human operators. As such, a human operator should be able to trust that the IDS is not mistaken in order to confidently investigate the alerts in the direction the IDS is pointing to. Therefore, in the current research landscape, there are a few important limiting factors [114, 9]:

- The inherent difficulty of the intrusion detection problem.
- The quality of datasets.
- The quality of metrics.
- Limits related to ML algorithms:
  - The difficulty to understand them.
  - The high requirements in labeled data quantity.
  - The necessity to frequently update the model as the system changes, or new data or classes need to be properly recognized.
  - The resistance to adversarial attacks.

In this thesis, we developed different methods to address some of these limitations, which will be introduced in the following sections.

2.6.1 Difficulty of the intrusion detection problem

The intrusion detection problem is inherently difficult for various reasons. First, normal traffic is generally much more diverse than other classes, i.e., different cyberattacks, and more prevalent. Therefore, when an attack signal is relatively weak, IDSs tend to lean towards identifying it as normal because of its prevalence. Secondly, many attacks try to be as discreet as possible, which further exacerbates this problem. Figure 2.5 shows the t-SNE [113] representation (a non-linear dimension reduction technique) of 30% of the CIC-IDS2017 dataset. It illustrates the fact the normal traffic is much more diverse as other classes, as well as the fact the different cyberattacks often overlap with normal traffic.

2.6.2 Limits and problems of intrusion detection datasets

An IDS should be trained on data as close as possible as what will be encountered in real-world situations. Otherwise, a notable decrease in performance is to be expected.

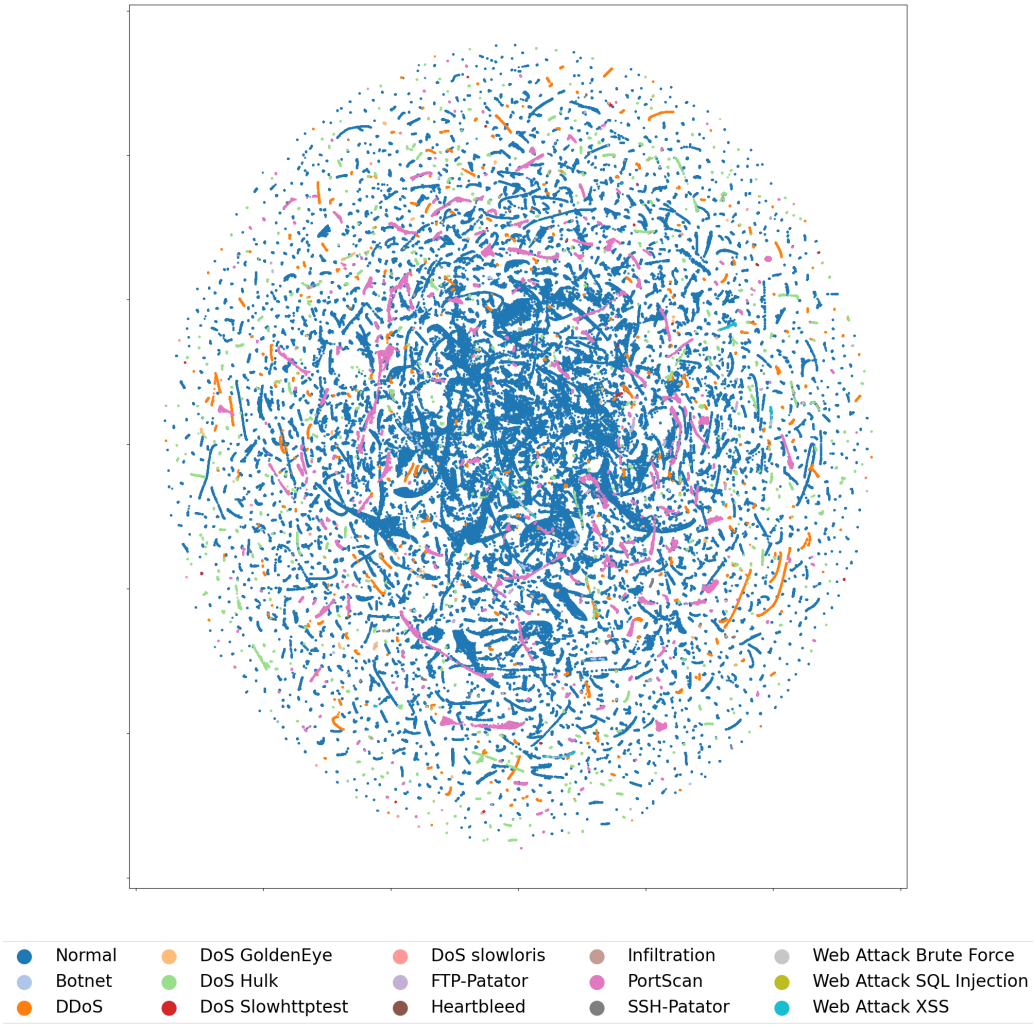


Figure 2.5 – t-SNE representation of 30% of the CIC-IDS2017 dataset

Unfortunately, many publicly available datasets are now old enough that the traffic they contain does not represent current traffic and attack methodologies. As stated in Section 2.2.2, two of the most used datasets, KDD'99 and NSL-KDD, are much too old. In [89], it is argued that even more recent datasets can quickly become obsolete in this quickly evolving threat landscape, unless routinely updated. While substantial work has been made to provide documented datasets following a more thorough methodology, such as for CIC-IDS2017, errors still remain [106, 97].

Our own experiments on the UNSW-NB15 data revealed similar findings. For example, most of the samples 424772 to 424824 of the `UNSW-NB15_4.csv` file have equal values for all features, while having different labels: `Exploits`, `Reconnaissance`, `DoS`, `Backdoor` and

`Fuzzers`. Consequently, any IDS would be unable to differentiate between these samples.

Unfortunately, there is no perfect public dataset yet, so validating an approach using the best publicly available datasets remains the only solution so far. An alternative approach would be to create and work on our own datasets, but this requires significant work and could instead impede comparison with other state-of-the-art approaches.

2.6.3 Limits of machine learning metrics

In order to properly compare different IDSs, the metrics used should correctly reflect the cost of either missing attacks or raising false alarms. While imbalance is a common problem (and still difficult to solve), intrusion detection also brings its own set of problems: cyberattacks and systems to protect are not equal, and it can be very beneficial in many cases to take this into account.

For example, a model that is very efficient in detecting DoS attacks will be much less useful on a system with sufficient capabilities to handle the additional load. An attack giving root access on a machine will also be more important to detect than port scans regardless of the system and giving additional importance to detecting elevation of privileges might make intrusion detection systems more useful.

Consequently, Chapter 3 in this thesis focuses on the creation of new ML metrics more suited to the evaluation of IDSs by integrating Common Vulnerability Scoring System (CVSS) scores into existing metrics.

2.6.4 Challenges related to ML methods

ML methods, regardless of the application domain, are often subject to the same drawbacks and limitations.

Requirements in labeled data

Multi-class classification approaches definitely require labeled data to learn. As has been proven with Large Language Models (LLMs), the ability to generalize can simply come from a larger amount of data, reducing the potential appearance of edge cases not encountered during training. However, in the context of intrusion detection, new attacks are constantly created, and it is thus necessary to be able to learn to recognize attacks with only a few instances. Few-shot and Zero-shot learning techniques can contribute to this problem by training models to generalize more easily to properly recognize relatively

rare classes. The approach developed in Chapter 5 is able to be trained with a low amount of labeled data and is thus able to answer to some of these concerns.

Interpretability of ML methods

Most current ML methods are based on DL techniques and their decision are thus not understandable to Artificial Intelligence (AI) practitioners, much less to potential end-users. One of the possibilities is to leverage XAI to explain decisions. However, as it will be shown later in Chapter 4, XAI is also subject to similar limitations as ML methods, such as the difficulty to evaluate the quality of explanations. Therefore, different metrics and properties were defined to provide a more objective way to evaluate explanations, as proposed in [126]. Following this evaluation of two popular XAI methods, LIME and SHAP, we developed a method (in Chapter 4) to leverage the evaluation of explanations to verify the accuracy or correct predictions of an IDS.

Concept drift and frequent model updates As has been mentioned previously, the cybersecurity landscape is constantly evolving, be it with new communication paradigms, new protocols and new cyberattacks. Normal traffic can change and new cyberattacks can appear. IDSs based on ML methods have to constantly be retrained to integrate these changes, which can be very time consuming, and can even create a race where ML methods would constantly need to be updated to follow recent changes. The approach developed in Chapter 5 is able to detect new cyberattacks faster, thus allowing to accelerate the update process of an IDS. Additionally, ML paradigms such as active learning or incremental learning can be beneficial to more quickly have up-to-date IDSs.

Chapter 3

Integration of cybersecurity knowledge through CVSS scores

Coming back to the typical use-case scenario presented in Figure 1.1 (Page 1), we can identify two different actors that have different knowledge and different needs: Artificial Intelligence (AI) practitioners and cybersecurity analysts. The handling of data has to be a shared endeavor, because both parties have specific needs. AI practitioners will need data in a specific format to both train and evaluate an Intrusion Detection System (IDS), while cybersecurity analysts need more raw data, both for logging and investigation purposes. While the IDS and a possible explainer are built by AI practitioners, the end users will be cybersecurity analysts. Therefore, the perception about performance of an IDS might not be the same for both actors. While an AI practitioner might be satisfied with a near perfect Recall, a cybersecurity analyst will be more interested in how much damage a cyberattack will actually cause.

Therefore, to be able to compare different Machine Learning (ML) approaches, it is important to have adequate metrics that properly represent the advantages of some approaches over others, for both AI practitioners and end-users. While most metrics, e.g., Accuracy, Precision, Recall, are usable with any classification task, some metrics are developed with specific goals in mind. For example, BLEU [131] and ROUGE [101] are metrics developed for Natural Language Processing (NLP), particularly suitable for translation tasks. Both metrics compare n-grams (groups of n words, e.g., bi-grams are groups of two words) between machine translated texts and a set of human translated references. BLEU is similar to the precision metric, but adapted to NLP, where the number of n-grams matching between machine translation and human translation are divided by the number of n-grams in the machine translation. ROUGE is similar to recall and divides the matching number of n-grams by the total number in the reference human translations. In the same way, Intersection over Union (IoU), measuring the overlap area of a ground truth box and a prediction box over the union area ($IoU = \frac{A \cap B}{A \cup B}$) is generally preferred to

other metrics for image segmentation tasks.

In the context of intrusion detection, while research has tried to mitigate the impact of imbalance, as shown in Chapter 2, there has been no tentative to create new metrics or adapt existing metrics to compare IDSs from a cybersecurity perspective. The two main requirements that need to be reflected in the given metrics are:

— Cyberattacks are not equal, and detection of attacks having a bigger impact should be prioritized.

— Systems to protect have their own requirements and a same attack will not have the same impact on different systems.

An obvious choice to integrate cybersecurity knowledge into IDSs is to leverage the MITRE ATT&CK framework [1]. MITRE ATT&CK is a knowledge base of tactics and techniques used to identify and categorize cyberattacks. However, while this is a great source of information and can definitely be useful to create a taxonomy of threats, it is difficult to apply to ML processes, particularly for evaluation purposes. Conversely, adding a numerical value representing the severity of an exploited vulnerability is a much easier task. This score is generally computed using the Common Vulnerability Scoring System (CVSS) [2], a scoring system used to grade the severity of a vulnerability according to its characteristics. CVSS scores are based on the Confidentiality, Integrity and Availability (CIA) triad and take into account parameters relative to the difficulty of performing an attack, as well as its impact, to compute a score representative of an attack's severity. Therefore, integrating CVSS scores into metrics allows solve the first requirement, i.e., more severe attacks have more importance in the results.

Research on the usage of CVSS scores in the context of cybersecurity has mainly focused on evaluating the security of systems and few has been done to evaluate IDSs. While the idea of leveraging CVSS in Intrusion Detection is not recent, as in [13] where it has been used to evaluate severity of alerts raised by probes, it has rarely been used since then. In [59, 58], CVSS scores have been used in coordination with attack graphs and Bayesian networks to evaluate or estimate the security of networks, thus extending the use of CVSS scores to also consider attack paths instead of a single vulnerability. In [26], CVSS scores are used with dynamic attack graphs to evaluate the overall security of a system. Although CVSS scores are originally defined for regular Information Technology (IT) networks, [136] have focused on extending the framework to also encompass Industrial

1. https://attack.mitre.org/
2. https://www.first.org/cvss/v3.1/specification-document

Control Systems (ICSs), showing the interest in such framework to evaluate security of a system. Finally, recent work [23] suggests that CVSS can be used to prioritize what is more severe. While research to extend CVSS scores to more use cases exists, it focuses on evaluating the security of a system and can be researched further to improve IDSs.

CVSS scores can be integrated into all phases pertaining to building an IDS, as shown in Figure 3.1.

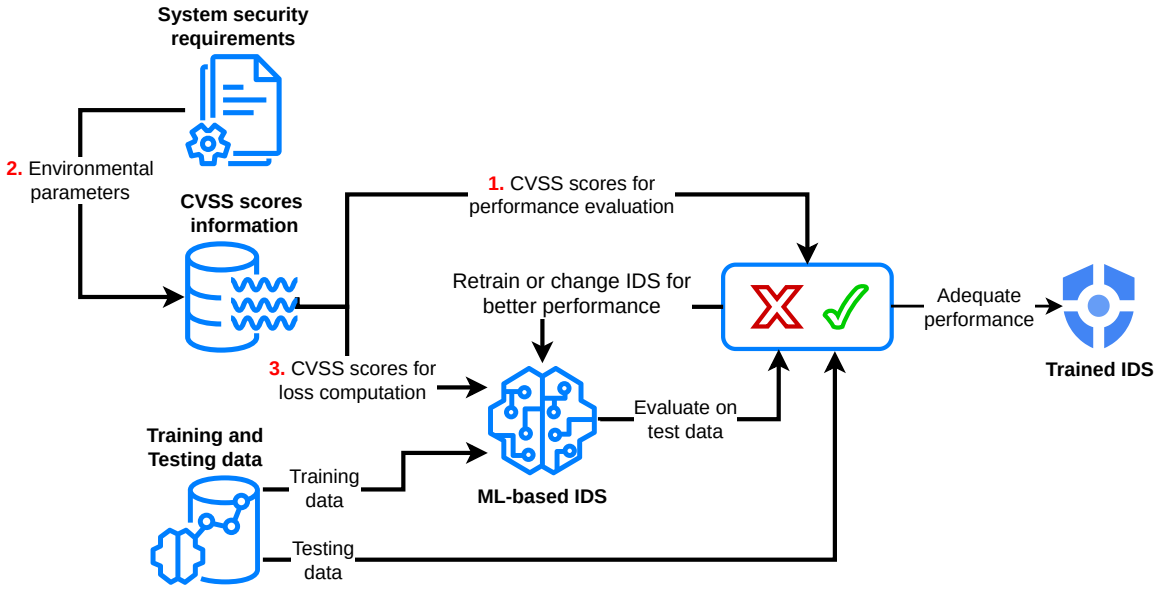


Figure 3.1 – CVSS integration into IDSs. Adapted from [51].

First, CVSS scores can be integrated into metrics for performance to reflect the actual ability of an IDS to prevent cyberattacks from causing damage, by taking into account the severity of attacks. CVSS scores can be obtained through the CVSS calculator [3]. In Figure 3.2, an example of how to use it to compute the CVSS score of a Denial of Service (DoS) attack is shown. Most parameters, i.e., Attack Vector, Attack Complexity, Privileges Required and User Interaction, used to compute the CVSS score, as well as their possible values are pretty straightforward to understand. Scope refers to the ability of an attack to impact components other that the vulnerable component, i.e., managed by different security entities.

Additionally, CVSS scores can be adjusted with environmental and temporal parameters. While temporal parameters characterize the reproducibility of an attack, environmental parameters allow to define requirements of a given system, to adapt returned

3. https://www.first.org/cvss/calculator/3.1

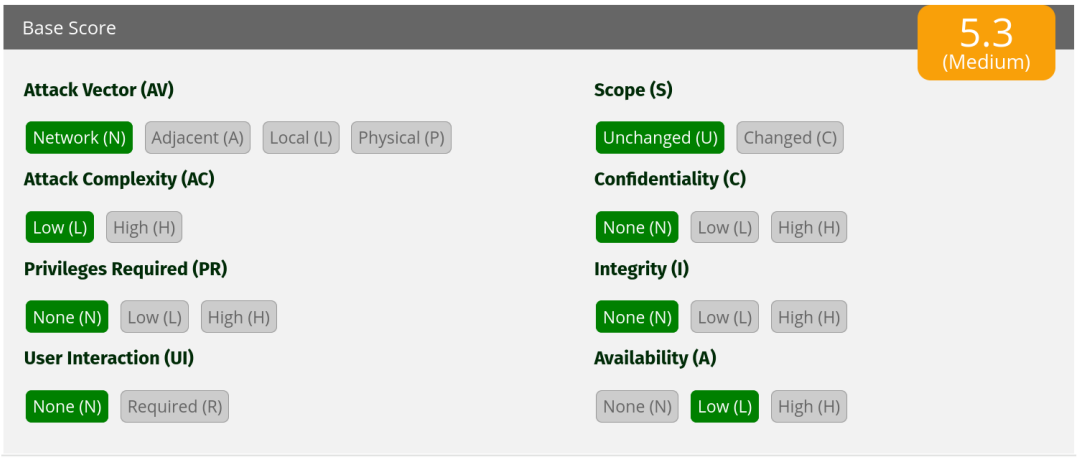


Figure 3.2 – CVSS score for a DoS attack using the CVSS calculator.

CVSS scores depending on their need for Confidentiality, Integrity and Availability. For example, DoS attacks that impact availability will have a lower CVSS score for systems with a low requirement for availability than systems with a high requirement. For a system with high requirements in Availability, CVSS scores change from 5.3 to 6.1, as shown in Figure 3.3.

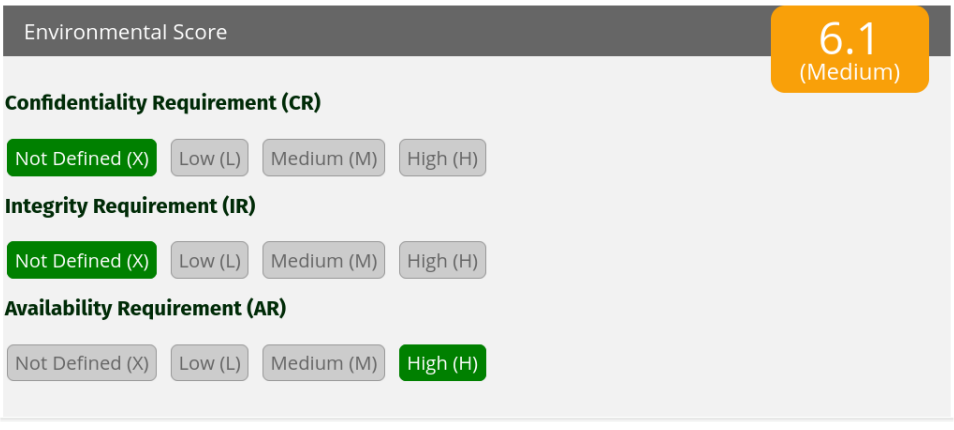


Figure 3.3 – CVSS score of a DoS attack, modified with environmental parameters to specify a system with high Availability requirements.

Finally, while integrating CVSS score into metrics allows to select an IDS that might be more adapted to a specific use-case, this does not help in building better IDSs. Therefore, it can be beneficial to integrate CVSS scores into the training phase of IDSs. While this might require substantial work for most ML-based IDSs, CVSS scores can straightforwardly be integrated in the loss formulation of Neural Networks (NNs)-based IDSs. Consequently,

the work presented in this chapter thus focuses on leveraging CVSS scores to better integrate cybersecurity knowledge into the different steps pertaining to building an IDS using ML.

3.1 Integrating CVSS scores into IDSs metrics

Besides being an evaluation of performance, a given metric (or set of metrics) in cybersecurity should be able to provide objective information about the actual cost of being mistaken, particularly in critical situations. Therefore, three new metrics using CVSS scores and accounting for missed attacks and false alarms were created in this thesis.

3.1.1 False Alarm Cost and Miss Cost

When evaluating an IDS, there are two ways it can be mistaken: it can raise false alarms, i.e., normal traffic misclassified as an attack or it can miss attacks, i.e., mistakenly considering attacks as normal traffic. Therefore, both the following metrics, False Alarm Cost (FAC) and Miss Cost (MC), are accounting for these two types of errors, while at the same time leveraging CVSS scores to include information about the cost of the misclassification.

Let c be a class, with $c \in \{0, 1, ..., C\}$ and 0 be the normal class. For every instance i, let G_i be the ground truth value and D_i be the decision for this instance. CVSS_i is the CVSS score corresponding to instance i.

$\mathbf{1}$ stands for the indicator function and $\overline{\bullet}$ is the averaging operator. $\overline{\text{CVSS}_c}$ thus corresponds to the mean of CVSS scores for instances belonging to class c, which is necessary when instances of the same class do not have the same CVSS score (as is the case in the UNSW-NB15 dataset).

For each attack class c ($c \neq 0$), and with N the total number of instances, we define the False Alarm Cost (Equation (3.1)) and the Miss Cost (Equation (3.2)) as follows:

$$\text{FAC}_c \overset{def}{=} \frac{\sum_{i=1}^{N} \mathbf{1}_{D_i=c} \cdot \mathbf{1}_{G_i \neq D_i}}{10 \sum_{i=1}^{N} \mathbf{1}_{D_i=c}} \cdot \overline{\text{CVSS}_c} \tag{3.1}$$

$$\mathrm{MC}_c \overset{def}{=} \frac{\sum_{i=1}^{N} \mathbf{1}_{D_i \neq c} \cdot \mathbf{1}_{G_i = c} \cdot \mathrm{CVSS}_i}{10 \sum_{i=1}^{N} \mathbf{1}_{G_i = c}} \tag{3.2}$$

In both formulae, the number 10 in the denominator represents the maximum possible value of a CVSS score, thus acting as a normalizing constant (bounding results between 0 and 1) while also highlighting the importance of attacks having a higher score.

As such, both formulae are generalizations of ML metrics. FAC is the generalization of the False Discovery Rate, the proportion of mistakes when predicting a specific class. In intrusion detection, it represents the frequency of false alarms, weighted by the CVSS score of these alarms. MC is the generalization of the False Negative Rate, the proportion of class instances that are incorrectly classified. In intrusion detection, it represents the frequency of missed attacks, weighted by their individual CVSS scores. These newly defined metrics are equal to their ML metrics counterparts when all CVSS scores are equal to 10 for classes different from normal traffic.

3.1.2 Cyber Informedness

Both metrics mentioned above can be combined into a single metric taking into account both False Positives and False Negatives that is defined analogously to Bookmaker Informedness (BI). Therefore, it is assumed it would similarly be relatively resistant to class imbalance, as shown in [189].

For each class c ($c \neq 0$), the Cyber-Informedness (CI) metric that contains both FAC and MC is given by (3.3).

$$CI_c \overset{def}{=} 1 - \mathrm{FAC}_c - \mathrm{MC}_c \tag{3.3}$$

This metric aims to give a cybersecurity-informed idea about the performance of an IDS, aggregating both FAC and MC, with values ranging between 1 and -1. It also represents the success of an IDS to correctly identify a specific attack, with less penalties for failing to recognize less critical attacks.

3.2 Experimental validation

In order to evaluate the usefulness of the new metrics, it is important to compare them to existing metrics with both quantitative and qualitative comparisons. Therefore, multi-

ple ML algorithms will be trained, tested and compared on three datasets, UNSW-NB15, CIC-IDS2017 and DAPT2020. This would allow to compare the overall results obtained with the newly defined metrics compared to extensively used ML metrics. Multiple metrics were retained for this comparison:

— Common ML metrics: Accuracy, F1-score, TPR (Recall), PPV (Precision). They are the most commonly used metrics for intrusion detection and also allow to compare IDS performance with other state-of-the-art approaches.

— Metrics potentially resistant to imbalance: Matthews Correlation Coefficient (MCC) and BI. Both range between -1 and 1. They allow to highlight the possible resistance to imbalance, given that the problem is formulated as a multi-class problem.

— Cyber-informed metrics: MC, FAC and CI. The former two range between 0 and 1 while the latter ranges between -1 and 1.

All metrics, except Accuracy and MCC, were computed on a per-class basis. The averaging method retained is macro-averaging, which averages irrespective of the class imbalance to reduce its influence. While most, if not all, results presented in the literature use micro-averaged results, we argue that difference is often not significant, because of the imbalance in IDS problems. Therefore, although less often used, macro-averaging properly reflect results on all classes without bias, contrarily to micro-averaging.

Dataset preprocessing

Three datasets were used for experiments in this chapter: UNSW-NB15, CIC-IDS2017 and DAPT2020. All datasets were split using a stratified scheme into 70% train (60% and 10% validation for DNNs) and 30% test sets. For all datasets, source and destination IP addresses were removed. While IP addresses can be useful descriptors, they allow to identify attackers more than they allow to learn how to recognize attacks. Therefore, it is best to remove them when training IDSs to recognize attack patterns.

For the UNSW-NB15 dataset, timestamps were removed because they do not inform about characteristics of an attack and can induce a bias. The feature *attack_cat* that reports the specific attack class (but is separated from the label feature) was also removed. Finally, categorical features or features having a small number of unique values were one-hot encoded. The resulting dataset has 229 features.

For the CIC-IDS2017 dataset, two features (`Init_Win_bytes_forward` and `Init_Win_bytes_backward`) were removed because of problematic values: most values are negative while a byte number should be positive. A further 5792 instances were re-

moved because of problematic values, e.g., NaN (Not a Number). A further eight features were removed because they were constant. The resulting dataset has 70 features.

For the DAPT2020 dataset, Flow ID and timestamps were removed because they do not inform about an attack and can induce a bias in the IDS. For labels, a combination of the Stage feature and the Activity feature was used (shown in Table 2.1). Stage corresponds to the current phase of an Advanced Persistent Threat (APT), i.e., what is the current goal pursued by the APT. For this dataset, an APT's behavior is separated into four stages:

— *Reconnaissance*. It entails gleaning as much information as possible about the targeted system.
— *Establish Foothold*. It is about getting a persistent access to the attacked machines, for example by obtaining user credentials.
— *Lateral Movement*. It is about moving in the targeted system to find more vulnerable machines, or obtain better credentials, such as root access.
— *Data Exfiltration*. It is simply about the exfiltration of collected data.

This combination gives a total of 21 classes, e.g., Network Scan Reconnaissance (Reconnaissance is the APT stage, while network scan is the activity) or Malware Download Establish Foothold.

Finally, an additional required pre-processing step in this case is to obtain CVSS scores of the different attacks realized to constitute the datasets.

Obtain datasets' CVSS scores

Ideally, datasets would be constituted with CVSS scores or Common Vulnerabilities and Exposures (CVE) identifiers, an identifier that represents the exploited vulnerability and allows to ask vulnerability databases for information about the vulnerability, including the CVSS score related to it [4]. For the UNSW-NB15 dataset, for example, CVE identifiers are often reported for a given cyberattack. Fortunately, tools commonly used to generate attacks, e.g., Metasploit [5], generally give the CVE identifiers of the exploited vulnerabilities, so integrating CVSS scores or CVE identifiers in the datasets is often relatively easy.

For the CIC-IDS2017 dataset, DAPT2020, and many other publicly available datasets, there is no such information. In these cases, although the information is missing, it is often

4. https://www.cve.org/
5. https://www.metasploit.com/

possible to manually score attack classes, given they are sufficiently detailed and classes are homogeneous enough, i.e., a class represents very similar attacks. For CIC-IDS2017 and DAPT2020, the attacks were described in the original papers [148, 125] and are sufficiently detailed to score attack classes with the CVSS calculator, as shown in Figure 3.2. For the DAPT2020 dataset, activities, e.g., network scan, malware download, etc., were used to attribute CVSS scores. The vectors used for computation are visible in Table 3.1. In both datasets, attacks except Infiltration are executed over the network and are repeatable, so the complexity is low. They also do not require any user intervention and are not able to impact components secured by other entities, so they have the same values for the Attack Vector (AV), the Attack Complexity (AC), the User Interaction (UI) and the Scope (S). For the Privileges Required (PR), it has been chosen to keep everything as *None* since the information is mostly lacking from dataset descriptions. Impact regarding Confidentiality (C), Integrity (I) and Availability (A) has been graded according to the expected capabilities of the given attacks, as well as their descriptions in the research papers introducing the datasets. The Infiltration attack from CIC-IDS2017 is graded as such because it requires interaction with a user that needs to download an infected file through dropbox or use an infected USB key. Finally, while DoS and Distributed Denial of Service (DDoS) attacks can have a relatively similar impact, DoS attacks tend to reduce performance of the target until an eventual shutdown, whereas DDoS can quickly make the targeted resource unavailable, thus the difference in impact. Attack classes have been grouped according to their name for better clarity and are visible in Table 2.1.

ML algorithms

In order to evaluate the proposed set of metrics and understand the differences brought by the introduction of cybersecurity-based metrics, experiments were run with a wide range of algorithms, trying various hyper-parameter combinations to find the best performing IDS on the two datasets considered, UNSW-NB15 and CIC-IDS2017. The retained algorithms are:

— A dummy classifier, classifying every instance as of the most frequent class (normal traffic in both datasets) to serve as a baseline.
— Relatively simple algorithms that should give an idea about the complexity of the classification task: Gaussian Naïve Bayes (GNB), Linear Support Vector Classification (LSVC), Decision Trees (DTs).
— More complex algorithms that should reflect the expected performance of IDSs re-

Table 3.1: CVSS scores for CIC-IDS2017 and DAPT2020

Attacks	AV	AC	PR	UI	S	C	I	A	CVSS Score
DoS attacks	Network	Low	None	None	Un-changed	None	None	Low	5.3
Scan, Patator and Brute Force attacks	Network	Low	None	None	Un-changed	Low	None	None	5.3
Web Attack XSS and CSRF	Network	Low	None	None	Un-changed	Low	Low	None	6.5
Infiltration	Local	High	None	Required	Changed	High	None	None	5.5
SQL Injection attacks	Network	Low	None	None	Un-changed	Low	Low	Low	7.3
Malware Download, Backdoor	Network	Low	Low	None	Un-changed	None	Low	None	6.5
DDoS	Network	Low	None	None	Un-changed	None	None	High	7.5
Heartbleed, Data Exfiltration	Network	Low	None	None	Un-changed	High	None	None	7.5
Botnet, Privilege Escalation	Network	Low	None	None	Un-changed	High	High	High	9.8

AV: Attack Vector, AC: Attack Complexity, PR: Privileges Required, UI: User Interaction, S: Scope, C: Confidentiality, I: Integrity, A: Availability.
Details about possible values for each category, as well as their signification, can be found at https://www.first.org/cvss/v3.1/specification-document.

lying on ML: Random Forests (RFs), Multi-Layer Perceptron (MLP), Deep Neural Networks (DNNs).

All algorithms are from the *scikit-learn*[6] library except DNNs that were programmed using the *PyTorch*[7] and *PyTorch Lightning*[8] libraries.

3.2.1 Results

In order to evaluate the usefulness of the newly defined metrics, IDSs based on the algorithms just mentioned were trained and tested on both the UNSW-NB15 and CIC-IDS2017 datasets.

Quantitative comparison

Results for both datasets are presented in Table 3.2. For each category of ML algorithm, a coarse grid-search scheme was used to pick hyper-parameter values and the IDS obtaining the best results was kept. For those IDSs, results are shown for the retained metrics.

Table 3.2: Performances on the UNSW-NB15 and CIC-IDS2017 datasets

Dataset	Algorithm	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
UNSW-NB15	**Dummy**	0.873	0.093	0.1	0.087	0	−0.102	0.654	0	0.346
	GNB	0.490	0.130	0.296	0.264	−0.039	−0.253	0.329	0.494	0.175
	LSVC	0.972	0.445	0.436	0.480	0.879	0.394	0.264	0.344	0.391
	DT	0.979	**0.586**	**0.565**	0.666	0.910	**0.535**	**0.186**	0.228	0.584
	RF	**0.981**	0.571	0.549	0.738	**0.919**	0.521	0.205	0.180	0.614
	MLP	0.980	0.520	0.518	**0.772**	0.912	0.488	0.220	**0.153**	**0.625**
	DNN	0.978	0.505	0.511	0.627	0.908	0.480	0.206	0.257	0.535

Continued on next page

6. https://scikit-learn.org/stable/index.html
7. https://pytorch.org/
8. https://www.pytorchlightning.ai/

Table 3.2: Performances on the UNSW-NB15 and CIC-IDS2017 datasets (Continued)

Dataset	Algorithm	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
CIC-IDS2017	**Dummy**	0.803	0.059	0.066	0.053	0	−0.172	0.603	0	0.397
	GNB	0.723	0.499	**0.848**	0.469	0.572	0.579	**0.069**	0.321	0.609
	LSVC	0.986	0.546	0.589	0.602	0.960	0.574	0.256	0.253	0.490
	DT	**0.998**	0.839	0.843	0.836	**0.995**	**0.842**	0.101	0.104	0.794
	RF	**0.998**	**0.850**	0.836	0.870	**0.995**	0.834	0.106	0.085	**0.808**
	MLP	0.996	0.725	0.721	0.835	0.989	0.718	0.177	0.103	0.719
	DNN	0.997	0.757	0.739	**0.896**	0.991	0.736	0.168	**0.067**	0.764

Values were truncated to the third decimal. Best results for a given metric and dataset are in **bold**.

Both Accuracy and MCC were used with a multi-class formulation instead of performing macro-averaging. They are therefore much more impacted by imbalance than other metrics. Table 3.3, that shows the average difference in performance of the four highest performing algorithms (DT, RF, MLP and DNN), highlights the fact that differences in results for accuracy and MCC are generally an order of magnitude smaller than for other macro-averaged metrics.

Table 3.3: Average differences in results of the four best algorithms on the UNSW-NB15 and CIC-IDS2017 datasets

Dataset	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
UNSW-NB15	0.0017	0.049	0.0322	0.0845	0.0058	0.0330	0.0172	0.0600	0.0500
CIC-IDS2017	0.0012	0.0762	0.0772	0.0362	0.0037	0.0783	0.0483	0.0215	0.0495

Values were truncated to the fourth decimal.

Disregarding CVSS scores, TPR (Recall) and MC are actually showing the same information, albeit in an opposite way. It is thus not surprising to see algorithms having a higher TPR also having a lower MC. However, it is possible to find models that perform

better on more severe attacks by comparing relative differences between both metrics. Results of the DT and RF for the UNSW-NB15 dataset can illustrate this. As can be seen in Table 3.4, on the UNSW-NB15 dataset, RF and DNN have a TPR respectively of 0.549 and 0.511, which represents a relative difference of 7%. Comparatively, the relative difference for MC is only of 0.5%. This means that although possibly less efficient overall, DNN might be better than RF at detecting more severe attacks. However, an important difference between both metrics is that contrary to MC, TPR results are also influenced by how well the IDS recognizes normal traffic. From FAC, it can be seen that DNN raise much more false alarms that RF, which means that their ability to detect attacks is actually similar. RF thus has a higher TPR (and much lower FAC) because it is better at recognizing normal traffic. The assumption that comparing relative differences allows to find models better at detecting more severe attacks is thus valid when FAC does not show contrasting information.

Table 3.4: Performances of DT and RF algorithms for UNSW-NB15

Algorithm	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
RF	0.981	0.571	0.549	0.738	0.919	0.521	0.205	0.180	0.614
DNN	0.978	0.505	0.511	0.627	0.908	0.480	0.206	0.257	0.535

On CIC-IDS2017, as reported in Table 3.5, GNB has a marginally higher TPR than DT, but a much better MC while having a worse FAC. While this shows that GNB will raise more false alarms, it has a near perfect detection of most attacks, including Botnet and Heartbleed where the DT model is struggling.

Table 3.5: Performances of GNB and DT algorithms for CIC-IDS2017

Algorithm	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
GNB	0.723	0.499	0.848	0.469	0.572	0.579	0.069	0.321	0.609
DT	0.998	0.839	0.843	0.836	0.995	0.842	0.101	0.104	0.794

Qualitative comparison

The following example (Table 3.6) provides a more detailed comparison for the LSVC (Linear Support Vector Classification) and MLP (Multi-Layer Perceptron) on the UNSW-NB15 dataset.

Table 3.6: Performances of LSVC and MLP algorithms for UNSW-NB15

Algorithm	Acc.	F1	TPR	PPV	MCC	BI	MC	FAC	CI
LSVC	0.972	0.445	0.436	0.480	0.879	0.394	0.264	0.344	0.391
MLP	0.980	0.520	0.518	0.772	0.912	0.488	0.220	0.153	0.625

When looking at the results, the Accuracy of both IDSs is very close, whereas results are very different according to Precision, False Alarm Cost and Cyber Informedness. For the Accuracy, this is understandable because results on most classes are very close. Both IDSs have relatively similar performance (under a 5% difference) on all classes, except `Exploits` and `dos`. LSVC outperforms MLP detecting `DoS` instances (69% versus 37%). On the other hand, MLP significantly outperforms LSVC for detecting `Exploits` (74% versus 46%).

In the UNSW-NB15 dataset, for attacks that do have CVE IDs and thus an assigned CVSS score, `Exploits` is the class with the highest average CVSS score because most instances have a high CVSS score (9.3 or 10). `DoS` attacks, on the contrary, generally have CVSS scores between 5 and 8. `Exploits` attacks are generally more dangerous, i.e., have a higher CVSS score, which is directly translated into those two metrics. Indeed, the MLP that performs better on `Exploits` has results that are more than two times better for the False Alarm Cost and close to 60% better on the Cyber Informedness metric. Furthermore, the relative difference in results of both models is higher for metrics using CVSS than their counterparts (Cyber Informedness vs BI, False Alarm Cost vs Precision, Miss Cost vs Recall). Since this relative difference is higher on metrics leveraging CVSS scores, it means the MLP model is more efficiently detecting attacks with a high CVSS scores.

Operationally, it means the MLP-based IDS will more often detect attacks that are critical and might endanger the system. When using such an IDS, automated mitigation strategies can be used with more confidence, and human operators will be able to divert their energy in investigating other more relevant alarms that might represent previously undetected attacks.

3.3 Adaptation to specific systems through environmental scores

The CVSS Environmental Score allows to define requirements in Confidentiality, Integrity and Availability, and these requirements are used to modify the returned CVSS scores. In doing so, scores returned by the CVSS scoring system differ from what would be returned without any requirements. Such requirements can easily be identified by cybersecurity analysts before building IDSs, to take into account specificities of the system to protect. Furthermore, systems could potentially benefit from a near automated adaptation of the IDS's evaluation according to their requirements. The only obstacle is the ability to automate the process of obtaining CVSS scores modified by environmental parameters, which could easily be done when sufficient information about the attacks is available.

3.3.1 Different system requirements

To evaluate changes in performance with CVSS-related metrics with CVSS environmental scores, three different environments, i.e., systems with specific security requirements, were considered:

— The basic environment, without any modification.
— A high Confidentiality, medium Integrity and low Availability (Environment 2), e.g., a marketing company's client database with other backups for data redundancy.
— A Low Confidentiality, medium Integrity and high Availability (Environment 3), e.g., a video streaming service.

Environment 2 (Env. 2) and Environment 3 (Env. 3) in Table 3.7 are representative of two different systems with different Confidentiality, Integrity and Availability requirements. Therefore, CVSS scores changes are reported. These changes in CVSS scores will in turn impact the relative performance reflected by CVSS-related metrics to help in selecting the IDSs that are most adapted to a particular environment.

3.3.2 Results

The main advantage of using the CVSS' environmental parameters is being able to find and differentiate IDSs that might be more adapted to protect a system given its security requirements, when their performance according to common metrics are relatively

Table 3.7: CVSS scores for CIC-IDS2017[a] and DAPT2020[b]

Attacks	**CVSS Scores**		
	Basic	**Env. 2**	**Env. 3**
DoS attacks[a]	5.3	4.6	6.1
Scan[a,b]**, Patator**[a] **and Brute Force attacks**[a,b]	5.3	6.1	4.6
Web Attack XSS[a] **and CSRF**[b]	6.5	6.5	6.5
Infiltration[a]	**5.5**	**6.5**	**2.9**
SQL Injection attacks[a,b]	**7.3**	**7.4**	**7.4**
Malware Download[b]**, Backdoor**[b]	6.5	6.5	6.5
DDoS[a]	7.5	5.7	9.3
Heartbleed[a]**, Data Exfiltration**[b]	7.5	9.3	5.7
Botnet[a]**, Privilege Escalation**[b]	9.8	9.8	9.8

[a,b] are used to show the datasets attack classes are belonging to.

similar. In Table 3.8, two very similar Decision Trees (DT) were compared using the three environments defined in Section 3.3.1. Both are `DecisionTreeClassifier` from the *scikit-learn* library, and the only difference between both is the criterion used, i.e., how is determined the quality of a node split. The first, DT 1, was trained with a Gini criterion while the second, DT 2, was trained with an entropy criterion with otherwise equal parameters. Both IDSs being very similar, performance is also expected to be very similar.

Both models results are almost equal on all classes, except for Infiltration (High impact on Confidentiality, DT 1 missed 40% while DT 2 missed 80%) and Web Attack SQL Injection (Low impact on Confidentiality, Integrity and Availability, DT 1 missed 67% while DT 2 missed 33%). Using common ML metrics, it is uncertain which IDS is the best, although cybersecurity analysts could decide which attack is considered more important to detect. However, CVSS's environmental parameters allows to automate the process of deciding which model is the best, depending on a system's security requirements.

The difference in CVSS scores for both Infiltration (Basic Env.: 5.5, Env. 2: 6.5, Env. 3: 2.9) and Web Attack SQL Injection (Basic Env.: 7.3, Env. 2: 7.4, Env. 3: 7.4) can be read in Table 3.7. When choosing which IDS to use on a specific system, it is important that its

Table 3.8: Model performances of two DTs on the CIC-IDS2017 dataset for different environments

Environment	Model	TPR	PPV	BI	MC	FAC	CI
Basic Environment	**DT 1**	0.830	0.880	0.828	0.117	0.080	0.802
	DT 2	0.824	0.882	0.822	0.114	0.069	0.815
Env. 2 (High C, Medium I, Low A)	**DT 1**	0.830*	0.880*	0.828*	0.130	0.085	0.783
	DT 2	0.824*	0.882*	0.822*	0.132	0.080	0.786
Env. 3 (Low C, Medium I, High A)	**DT 1**	0.830*	0.880*	0.828*	0.104	0.076	0.818
	DT 2	0.824*	0.882*	0.822*	0.096	0.059	0.843

Values were truncated to the third decimal. C: Confidentiality, I: Integrity, A: Availability.
* TPR, PPV and BI values are equal for all environments.

performance in detecting specific attacks aligns with the requirements of the system. For example, when choosing an IDS for a system with similar requirements as Env. 3 (high availability requirements), DT 2 seems more suitable since it is better at detecting Web Attack SQL Injection that impacts availability.

3.4 Enhance training of intrusion detection systems with CVSS scores

While Miss Cost, False Alarm Cost and Cyber Informedness allow to choose IDSs that are more adapted to specific systems and take into account the severity of attacks, this does not make IDSs better with regard to this.

Integrating CVSS scores into the loss formulation of a NN allows to also build IDSs that can detect more severe attacks more often, as well as be better adapted to specific systems and their security requirements. In this way, it is possible for AI practitioners to leverage cybersecurity knowledge in order to build more efficient IDSs.

3.4.1 CVSS in the loss computation

In much the same way as in the previously defined metrics, CVSS scores can be integrated into a loss used by a NN to train. Since in most multi-class classification

problems, the loss used is a Cross-Entropy loss (CE), CVSS scores have been integrated into a custom loss based on the CE.

Let c be a class, with $c \in \{0, 1, ..., C\}$ and 0 being the normal class. For every instance i, let x_i represents the output logits of the Neural Network, G_i be the ground truth value and D_i be the decision for this instance. CVSS_i is the CVSS score corresponding to instance i. Finally, let V be the set of indices for which CVSS_i exists.

As a reminder, the basic CE is defined in Equation (3.4).

$$\text{CE} \stackrel{def}{=} \sum_{i=1}^{N} -log \frac{exp(x_{i,G_i})}{\sum_{c=1}^{C} exp(x_{i,c})} \tag{3.4}$$

When building an IDS, it has to learn by correcting two different types of errors: missing attacks, and raising false alarms. Therefore, the envisioned loss leveraging CVSS scores needs to at least account for these two types of errors. As was the case for the metrics defined previously, CVSS scores cannot be added in the same way for the two error types, which need to be separated into different parts:

— Miss Cross-Entropy loss (MCE), a part accounting for missed attacks, defined analogously to the Miss Cost. It is defined in Equation (3.5).

$$\text{MCE} = \sum_{i=1}^{N} -log \frac{exp(x_{i,G_i})}{\sum_{c=1}^{C} exp(x_{i,c})} \cdot \mathbf{1}_{G_i \neq 0, G_i \neq D_i, i \in \text{V}} \cdot \text{CVSS}_i \tag{3.5}$$

— False Alarm Cross-Entropy loss (FACE), a part accounting for false alarms, defined analogously to the False Alarm Cost. It is defined in Equation (3.6).

$$\text{FACE} = \sum_{i=1}^{N} -log \frac{exp(x_{i,G_i})}{\sum_{c=1}^{C} exp(x_{i,c})} \cdot \mathbf{1}_{G_i = 0, D_i \neq 0} \cdot \overline{\text{CVSS}_{D_i}} \tag{3.6}$$

— Remaining Cross-Entropy loss (RCE). This part is not always required, but has been added to account for cases when missed attacks might not have a corresponding CVSS scores, as is sometimes the case for the UNSW-NB15 dataset. It

is defined in Equation (3.7).

$$\text{RCE} = \sum_{i=1}^{N} -log \frac{exp(x_{i,G_i})}{\sum_{c=1}^{C} exp(x_{i,c})} \cdot \mathbf{1}_{G_i \neq 0, G_i \neq D_i, i \notin \text{V}} \tag{3.7}$$

Contrarily to the defined metrics, MCE and FACE do not require numerical constants to normalize the values. While it could be added, it only influences the amplitude of error gradients and is easily counterbalanced by the training optimizer's learning rate, thus proving to be unnecessary. Finally, the complete loss, CVSSCE, is defined in Equation (3.8), simply being a sum of its three parts.

$$\text{CVSSCE} \overset{def}{=} \text{MCE} + \text{FACE} + \text{RCE} \tag{3.8}$$

Because it is composed of three different losses that are simply summed, it is relatively trivial to give more importance to one loss, e.g., give more importance to MCE by adding a weight to it if missing attacks is more critical than raising false alarms.

3.4.2 UNSW-NB15 data subset for experimental validation

To more effectively evaluate the impact and effectiveness of integrating CVSS scores into the loss formulation, only a subset of the UNSW-NB15 dataset has been retained. This particular choice has been made to focus on the influence of the loss integrating CVSS scores while reducing the impact of other variables. As shown in Table 3.9, only browser exploits and normal traffic were retained. Browser exploits were then separated into different classes according to their CVSS score, e.g., browser exploits with a CVSS score of 10 belong to class `exploits-Browser-10`, because they are exploits targeting different vulnerabilities and with potentially different attack mechanisms. By selecting a single attack class separated in multiple sub-classes, traffic should overall be much more homogeneous between attack classes, and as stated previously, should reduce the influence of variables other than the loss using CVSS scores.

This task is highly difficult for two reasons: classes are even more imbalanced than in the original UNSW-NB15 dataset as normal traffic represents more than 99% of the data, and, as stated previously, the different attack classes are very similar, making it much harder to differentiate them. Moreover, one attack class is also much more present than the eight other classes, adding to the imbalance in the data, and potentially biasing the

Table 3.9: Datasets details

Dataset	Number of instances per class	Total
UNSW-NB15	Normal: 2218761, exploits-Browser-10.0: 537, exploits-Browser-9.3: 13988, exploits-Browser-8.5: 232, exploits-Browser-7.6: 233, exploits-Browser-7.5: 1149, exploits-Browser-7.1: 94, exploits-Browser-6.8: 511, exploits-Browser-5.1: 1589, exploits-Browser-5.0: 274, exploits-Browser-4.3: 443	2237811

IDS to mistake other classes for this more prevalent attack class.

3.4.3 Results

In order to test the advantages of using a loss integrating CVSS scores, two NNs were trained and tested on the dataset reported in Table 3.9, one with a classic CE as in Equation (3.4), and the other with the loss using CVSS scores as formulated in Equation (3.8). The NNs used the exact same architecture as those reported in Table 3.2, because more complex architectures did not seem to improve performance and instead made the training more unstable.

Results of IDSs trained with either loss are reported in Table 3.10. Interestingly, only taking classic ML metrics into account, the IDS trained using the loss with CVSS already exhibits a close to 50% higher PPV (Precision), which means the IDS is much more often right when predicting different attacks.

Table 3.10: Performance of NNs trained with a basic CE or with CVSSCE

Training loss	TPR	PPV	BI	MC	FAC	CI
Basic CE	0.258	0.246	0.233	0.579	0.589	−0.169
CVSSCE	**0.288**	**0.360**	**0.263**	**0.547**	**0.488**	**−0.035**

Values were truncated to the third decimal.

Taking into account CVSS-related metrics, the IDS trained using the loss with CVSS is also clearly more effective in detecting attacks, and possibly more severe attacks.

In order to properly investigate whether training with a loss using CVSS scores do help in detecting more severe attacks, it is required to go over performance of the IDSs on different classes in more details. For convenience, the IDS trained without CVSS scores will be called CE-IDS, and the one trained with will be called CVSSCE-IDS.

Because `exploits-Browser-9.3` and `exploits-Browser-5.1` are more frequent than other attack classes, their Accuracy is expectedly higher for both IDSs, as shown in Figure 3.4. However, CVSSCE-IDS has an Accuracy around 10% higher for the four other most severe attacks, which is quite significant considering these attacks are completely undetected by CE-IDS.

Furthermore, as shown in Figure 3.5 almost all attacks were completely missed 10 to 20% less by CVSSCE-IDS, and were often misclassified as `exploits-Browser-9.3` if not correctly classified. A possible explanation is that using CVSS scores in the loss

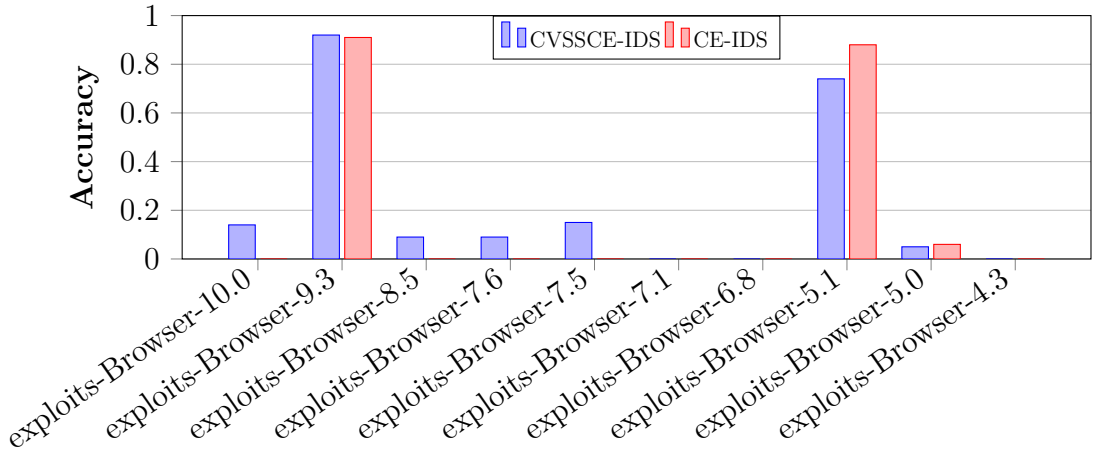


Figure 3.4 – Accuracy of two IDSs trained with a CVSSCE (CVSSCE-IDS) and with a basic CE (CE-IDS), for each class ([51]).

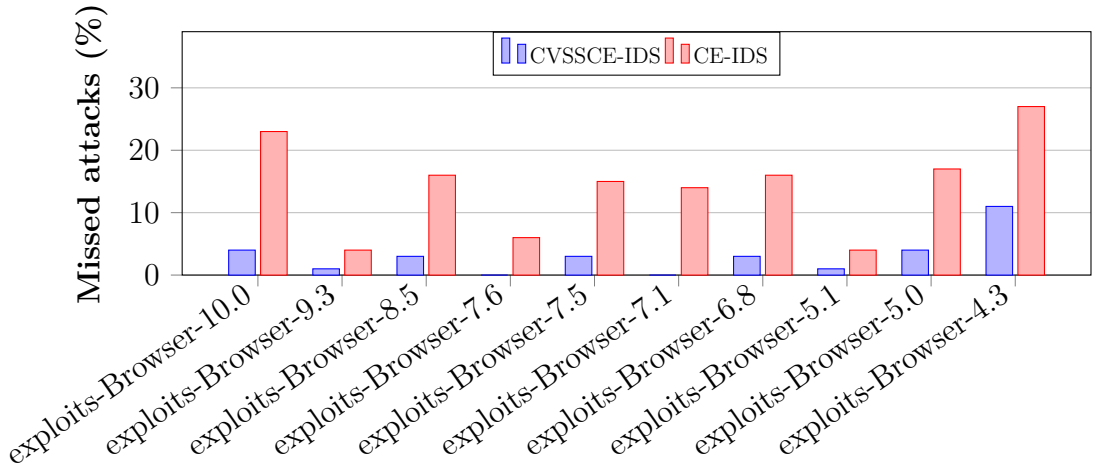


Figure 3.5 – Percentage of missed attacks for the two IDSs trained with a CVSSCE (CVSSCE-IDS) and with a basic CE (CE-IDS), for each class ([51]).

formulation ends up penalizing missed attacks (particularly severe attacks) much more, thus forcing the IDS to refine its representation of attack classes more than without using CVSS scores.

Finally, results using the Cyber Informedness metric for each class (Figure 3.6) also validate the higher performance of the IDS trained using CVSS scores. While performance according to this metric is more or less equal for the two most prevalent attack classes (`exploits-Browser-9.3` and `exploits-Browser-5.1`) and attack classes with a CVSS score of 7.1 or below, the IDS trained using CVSS scores is clearly more efficient to detect attacks with a CVSS score of 7.5 or higher.

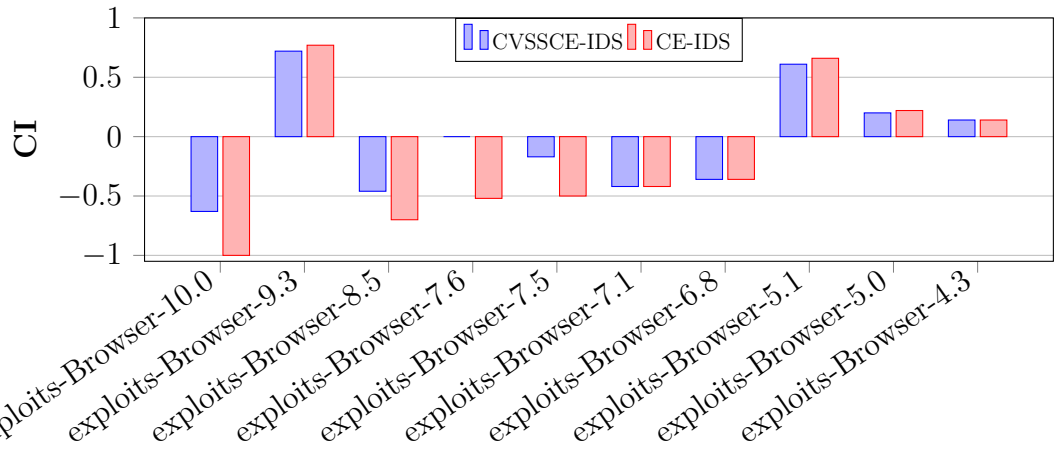


Figure 3.6 – Cyber Informedness results for the two IDSs trained with a CVSSCE (CVSSCE-IDS) and with a basic CE (CE-IDS), for each class ([51]).

3.5 Validation of the approach on DAPT2020

To ascertain that the proposed whole approach is applicable to more recent attack methodologies, experiments performed previously have been reproduced on the DAPT2020 dataset. Once again, NNs having the same architecture were trained using both CVSSCE and CE losses.

3.5.1 CVSS scores in the loss

As shown in Table 3.11, results obtained validate that NNs trained using CVSS information tend to completely miss attacks much less often, and thus also exhibit a much

Table 3.11: Performance of NNs trained with a basic CE or with CVSSCE on DAPT2020

Training loss	TPR	PPV	BI	MC	FAC	CI
Basic CE	0.391	**0.483**	0.362	0.338	**0.348**	0.314
CVSSCE	**0.491**	0.317	**0.419**	**0.097**	0.453	**0.449**

Values were truncated to the third decimal.

higher performance with CI. The approach of using CVSS scores inside the loss formulation seems to be even more effective on this dataset that better represents current attack methodologies. This shows promising prospects with regard to application to real-world scenarios.

Computation time While using a loss based on CVSS might slow down convergence or require a bit more computations, experiments performed on DAPT2020 tend to show a training time overhead of around 10%. On bigger datasets, a similar overhead can be expected. Additionally, since the loss is only used at training time, there is no overhead for inference.

3.5.2 CVSS score in the loss with different environments

While we have shown in previous sections that CVSS scores can be leveraged to better evaluate IDSs, potentially targeting a specific system with particular protection needs, as well as help training NN-based IDSs, we tested the approach on separate goals. It is therefore interesting too see if combining every part, i.e., training a NN-based IDS targeted at a specific system using environmental scores for both training and evaluation, helps in building IDSs more adapted to a given system.

To do so, we used the three environments previously defined, with their respective CVSS scores, as shown in Table 3.7. Four NNs using the exact same architecture were trained without any CVSS score, or on a specific environment, using its CVSS scores in the loss computation, and all four were tested on all three environments. In doing so, it is possible to see if training to adapt to a particular environment helps having a higher performance, using Miss Cost, False Alarm Cost and Cyber Informedness.

The first observation in Table 3.12 is that, regardless of the CVSS scores used during training, NNs trained using CVSS scores end up with an overall higher performance for all

Table 3.12: Performance of NNs trained with different environments' CVSS scores on DAPT2020

NN Name (Training environment)	Basic Env			Env 2			Env 3		
	MC	FAC	CI	MC	FAC	CI	MC	FAC	CI
NN-0 (None)	0.338	0.348	0.314	0.327	0.372	0.301	0.295	0.401	0.304
NN-1 (Basic env)	**0.108**	0.356	**0.537**	0.128	0.420	0.452	0.129	0.437	0.434
NN-2 (Env 2)	0.158	0.447	0.395	**0.157**	0.435	**0.408**	0.136	0.368	0.495
NN-3 (Env 3)	0.098	0.453	0.449	0.094	0.471	0.434	**0.081**	**0.382**	**0.536**

Values were truncated to the third decimal.

environments compared to the NN trained without CVSS scores. The second observation is that training without CVSS scores leads to NN-0 often raising marginally less false alarms than other NNs, but it also leads to missing much more attacks.

For both the basic environment and environment 3, the NNs trained on these environments, NN-1 and NN-3, end up achieving the highest performance. On the second environment, however, although NN-2 is better at detecting attacks than NN-0, and has an overall higher performance, its performance is below both NN-1 and NN-3. A possible reason is that CVSS scores for the second environment are much more homogeneous: they are mostly between 6.1 and 7.4, whereas the lowest CVSS scores reach 5.3 or 4.6 for the other environments. Therefore, it might lead to the IDS focusing more on more prevalent attacks, reducing its performance on more severe attacks and thus impacting its overall performance.

However, it is important to note that results were obtained using NNs sharing the same architectures and training parameters. Therefore, properly optimizing hyperparameters might still lead to IDSs trained in a specific environment being more effective in their environment than IDSs trained in other environments. Nevertheless, training using CVSS scores, regardless of the training environment, helps in targeting more severe attacks compared to training without it.

3.6 Conclusion

Using CVSS scores with IDSs seems to be able to fulfill multiple purposes, be it for training to help IDSs detect more severe attacks, or to be integrated in evaluation metrics to find IDSs that are better at detecting severe attacks or more adapted to a particular system. When models show very similar performance on the newly defined metrics, it generally means that their performance is similar on all attacks or that they differ for attacks having similar CVSS scores. The high performance on critical attacks is thus adequately highlighted. Interestingly, some IDSs, although exhibiting relatively poor performance in general, can have an unexpectedly good performance in some aspects, e.g., the GNB-based IDS for CIC-IDS2017 (in Table 3.2) which is the best attack detector at the cost of more false alarms. Thus, using those IDSs could be interesting when implementing ensemble methods for intrusion detection.

Using CVSS scores with environmental parameters also seems to enable building IDSs being specialized for protecting specific systems with different requirements. However, CVSS base vectors of the encountered attacks need to be available to be able to compute modified CVSS scores, thus requiring more information on the attacks. If available, this could potentially lead to a fully automated evaluation targeted at specific systems.

Integrating CVSS scores into a loss formulation to train NN-based IDSs seems highly effective and does not seem to really suffer from evident drawbacks. Although integrating CVSS scores into the training of an IDS based on NNs is doable, more work still needs to be done for IDSs based on other ML models.

CVSS scores are the most often used to describe the impact of exploiting vulnerabilites with regard to the CIA triad. Furthermore, this is a metric used by the cybersecurity community to score every new CVE, which makes it one of the best options in leveraging cybersecurity knowledge to create better IDSs. The only times it might prove less suitable is when there is no immediate impact on the system, e.g., privilege escalation. In these cases, attacks could be attributed a CVSS score depending on what it can enable. For example, privilege escalation can give full access and could therefore be given a high impact on each component of the CIA triad. This reflects the fact that privilege escalation should really be detected, and thus its CVSS score should be high. Although these new metrics are based on a *de facto* cybersecurity standard score, using such standard requires a more consistent effort in data collection to take advantage of CVSS scores to train and find better and more adapted IDSs.

Finally, in a real world scenario, alerts raised by signature-based methods can often be matched to CVE IDs either automatically or by a human expert, which could help create a local training dataset containing CVSS information. In the few cases where this is not possible, e.g., attacks with various capabilities, such as computer viruses, they could nevertheless be split into parts with a single capability as has been done for DAPT2020 or be represented in attack graphs to be properly scored.

Chapter 4

Explainability for Intrusion Detection Systems: What, why and how?

As has been mentioned in previous chapters, Intrusion Detection System (IDS) leveraging Machine Learning (ML) techniques involve both Artificial Intelligence (AI) practitioners that build and train the IDSs, as well as cybersecurity experts that will be their users. While an evaluation using cybersecurity knowledge, as was developed in Chapter 2, is necessary for cybersecurity experts to have more trust in the performance of the IDS, it might also be necessary to understand its decision process for various reasons, e.g., for a faster investigation, for trust, for legislative reasons, such as the European's AI Act [1].

While there is a necessity to be able to understand the decision process of an ML algorithm, there is also a need to take into account the recipient of the explanations. As noted in [121], Explainable AI (XAI) can benefit from research in social sciences to improve the quality of an explanation:

— Explanations are contrastive: users usually do not ask why an event E happened, but why another event X happened instead of event E. For intrusion detection, this might translate to asking why this event is categorized as a Denial of Service (DoS) instead of normal traffic caused by high quality video streaming.

— Explanations are selected in a biased manner: even if all causes of an event are provided, users will select a few that are most important to them, i.e., information that is relevant according to their own experience. For intrusion detection, a cybersecurity expert might only look for a very high volume, as well as where does the traffic come from, as causes for a DoS.

— Probabilities probably do not matter: users will rather rely on causes of an event than probabilities or statistical relationships. For intrusion detection, this trans-

1. https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

lates to users preferring to see that "This is a DoS because of the very high volume of traffic" rather than "High volume of traffic is linked to Denial of Service 70% of the time".

— Explanations are social: explanations are a transfer of knowledge and should be presented considering the users' knowledge. For intrusion detection, this means that explanation methods should use cybersecurity experts' knowledge to present information.

The latter three points do apply in intrusion detection, and are particularly important in a time-constrained context. A cybersecurity expert will put more emphasis on being sure that a particular event happened, as quickly as possible, because it is much more important, in case of an attack, to be able to enable remediation as fast as possible. The first point, however, is less important and would rather benefit post mortem investigations, when there is time to fully understand what happened. In [127], authors expand on the characteristics of an explainable IDS and also stress the need for explanations to be adapted to, and use, the knowledge of the ones receiving them.

4.1 Benefits and drawbacks of XAI for IDSs

Among possible XAI methods, transparent ML algorithms can be used to build explainable IDSs. However, it is relatively difficult to properly balance performance and explainability. Most interpretable algorithms, e.g., linear regressions, bayesian models, etc., have a lower performance than other methods. Decision Trees (DTs) are generally among high performance models, as shown in Chapter 2, but the ability to understand their decision process is also influenced by the depth of the tree, as well as the number of nodes. Our own experiments, on both CIC-IDS2017 and UNSW-NB15, showed that to reach similar performance to other high performing algorithms, DTs are growing too big, with tens of thousands of nodes and a depth sometimes close to 100. These DTs are therefore not really understandable from the viewpoint of a user trying to follow its decision process.

Because explanations have to be provided for users, they should be straightforward and compact (provide a low quantity of information) enough that users could quickly handle this information. A good indicator would be the user being able to mentally simulate the possible decision process of the IDS. In this context, it thus seems a better solution to focus on post-hoc methods, that are able to provide explanations for trained ML algorithms.

4.1.1 Post hoc methods

While LIME [138] and SHAP [112] have often been used to provide explanations about IDSs, these methods are feature importance methods, i.e., provide relative importance of different features with regard to the decision of the IDS, and do not necessarily conform to the four points previously mentioned at the beginning of this chapter. Another method from the authors of LIME, Anchors [139], might prove a bit more informative since explanations are presented in the form of rules, which might be more understandable than feature importances. All these methods are local methods, i.e., will explain a single decision, and model-agnostic, which allows them to explain any ML algorithm.

While research has tried using XAI methods for IDSs, with LIME used in [175, 116, 77, 109] and SHAP used in [173, 116, 124, 17, 182, 77], they only showed how these methods could be used to try explaining an IDS and did not try to evaluate the quality of these approaches with either qualitative or quantitative metrics. Therefore, it is still unclear how these explanations would be received by cybersecurity experts and how useful they could be.

In order to measure the usefulness of XAI methods, we tried to apply LIME, SHAP and Anchors on CIC-IDS2017 to obtain explanations. We then qualitatively evaluated the provided explanations to see if they could be useful to cybersecurity experts.

To test these XAI methods, two Neural Networks (NNs) were trained:

— A very simple NN, a single hidden layer with 256 neurons and ReLU non-linearities, named NN-1.

— A bigger NN with 6 hidden layers of size 256, 512, 1024, 512, 256, 128, also with ReLU non-linearities, named NN-6.

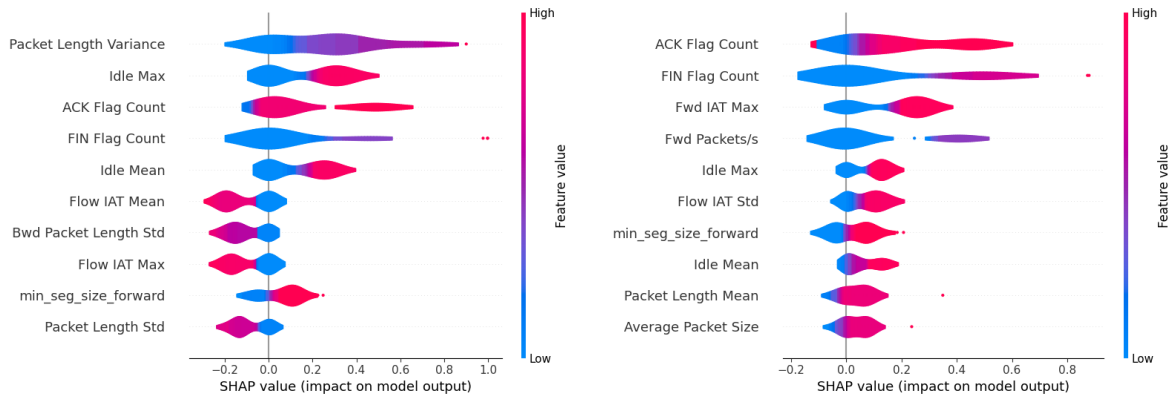
To ensure that these models were still performing relatively well to detect cyberattacks, accuracy of both NNs are reported in Table 4.1. While their global accuracy is very high, these NNs are still struggling to detect some cyberattacks, e.g., `Web Attack SQL Injection`, which might in turn impact the quality of explanations on attacks more difficult to detect.

Class explanations First, it is interesting to see if correctly learned attacks are explained in a way that conforms to the knowledge of cybersecurity experts. To this end, we chose explanations of correctly classified `DoS Hulk` (Denial of Service HTTP unbearable load king) instances. As shown in Table 4.1, this class is particularly well detected by both NNs, which should create a consistent representation of this attack. `DoS Hulk` attacks are

Table 4.1: NNs Accuracy on CIC-IDS2017

IDS	**NN-1**	**NN-6**
Accuracy	0.9963	0.9969
Recall Dos Hulk	0.9998	0.9988
Recall Web Attack SQL Injection	0.1667	0.1982

characterized by the usage of various User Agents, the usage of different times for keep-alive connections, the disabling of cache, as well as the usage of randomized unique URL requests. More globally, DoS attacks are generally characterized by a high number of packets, and possibly also a higher size of packets, either to or from the targeted machine.

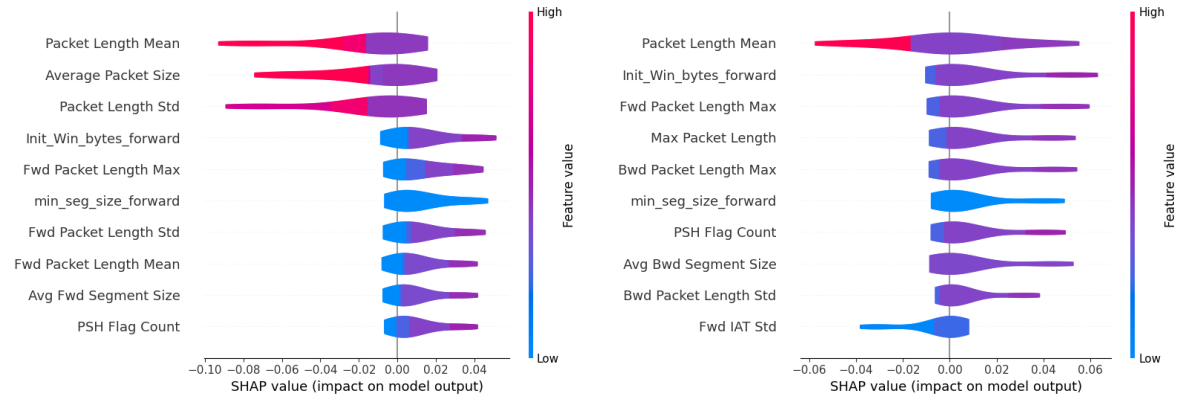


(a) Feature importance for NN-1 (b) Feature importance for NN-6

Figure 4.1 – Feature importance for the DoS Hulk attack, using SHAP

In Figure 4.1 is shown the importance of different features in identifying the class `DoS Hulk`, i.e., it shows feature importance for all instances of this attack, for both NNs. Features are ordered by importance, with the most important being at the top. The more important to a decision a specific feature value will be, the higher its SHAP value will be. At first glance, it is interesting to notice that `Packet Length Variance`, which is the most important feature for NN-1, is not among the 10 most important features for NN-6. For four features, namely `FIN Flag Count`, `Idle Mean`, `Idle Max` and `ACK Flag Count`, both NNs give a high positive importance to higher values of these features, which

is consistent with characteristics of this attack. High numbers of `FIN Flag Count` and `ACK Flag Count` show a high number of connections, while high values for `Idle Mean` and `Idle Max` show that the connection is kept alive for a long time, both representative of DoS attacks. Other important features for NN-1 make less sense to characterize DoS attacks, with four of these features having mostly a negative influence, i.e., the higher their values, the more chances that the IDS will predict something else than DoS Hulk. For NN-6, a high number of packets sent from Source to Destination per second (`Fwd Packets/s`) being important also make sense.



(a) Feature importance for NN-1

(b) Feature importance for NN-6

Figure 4.2 – Feature importance for the Web Attack SQL Injection attack, using SHAP

On the other hand, for attacks that are much harder to detect, like `Web Attack SQL Injection`, explanations can prove to be mostly useless. This is further amplified by the fact that SQL injections are mostly recognized by investigating packet content, so relying on statistical features does not help. As shown in Figure 4.2, while some feature values have a positive influence, this influence is relatively low (up to 0.08, for a maximum of 1), which shows they do not help much in detecting SQL injections.

While it is possible to ask for explanations for a single decision, conclusions remain mostly the same as what happens for class explanations: the usefulness of the explanation is heavily influenced by the performance of the IDS, as well as the fact that the data's feature set possess characteristics of different attacks. In Figure 4.3, an explanation of a single `Web Attack SQL Injection` is provided. As can be seen, the importance of the different features is relatively low, and some features even have a negative influence on the prediction. This shows that the IDS is not able to correctly detect this attack, and the explanation is also unable to provide any insight.

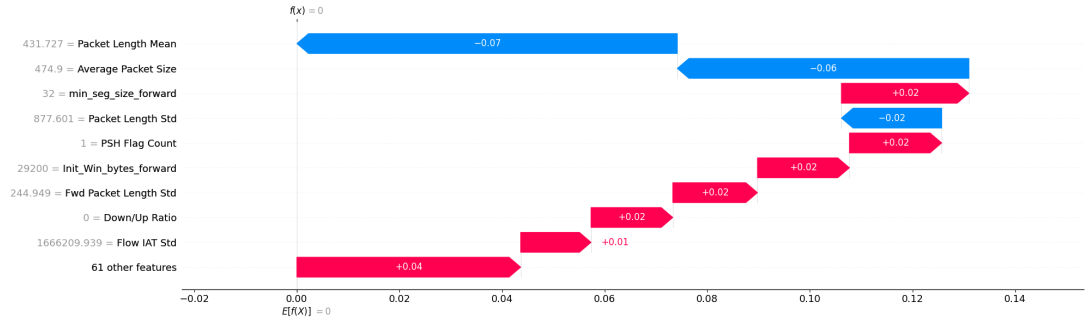


Figure 4.3 – Explanation with SHAP of a single Web Attack SQL Injection instance

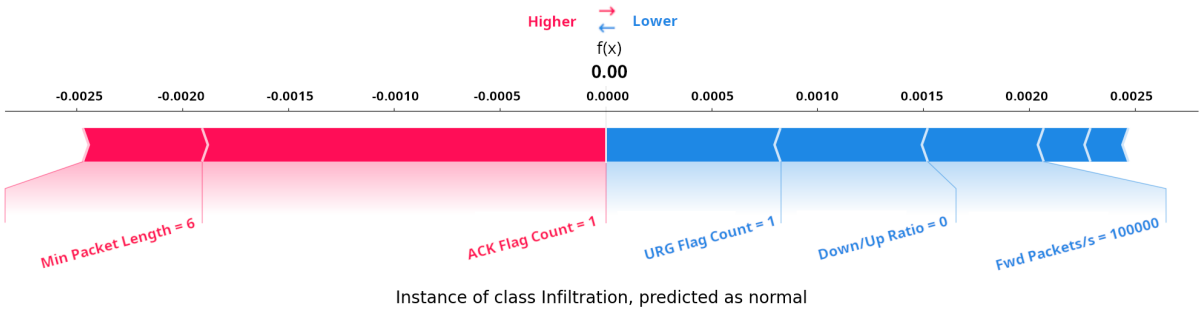
Consequently, it can be assumed that explanations can only make sense and be actually useful when, and if, the performance of the IDS is high enough. Furthermore, XAI methods showing feature importances are heavily dependent on the feature set. If characteristics of an attack are not visible in the feature set, e.g., for attacks that can be recognized with the packets payload, explanations are unable to be useful to the end users, which in this case would be a cybersecurity expert.

4.1.2 XAI to find spurious correlations or identify problems

While XAI methods are unable to properly explain how an IDS reaches a decision, especially when its detection ability is low, they can still be useful to find errors or biases in the data. Removing these errors and reducing an existing bias can improve performance of any ML algorithm, as highlighted in [4, 133]. In [109], the data used was heterogeneous, and depending on how recent it was, it did not necessarily have the same features. XAI allowed to find that the IDS was heavily biased towards detecting attacks by using the absence or presence of some features in the data, instead of the attacks characteristics. Consequently, removing such biases allows to increase trust in the IDS for cybersecurity analysts, even in the absence of proper explanations of the decision process.

As an example, the `Infiltration` attack in CIC-IDS2017 is sometimes undetected. NN-6 detected 60% of the `Infiltration` instances. In this case, explanations can help in highlighting the reasons the model is unable to find them, and could potentially be used to correct the IDS or enrich the dataset used to train it.

In Figure 4.4a, it can be seen that although having positive and negative importance, the most important features actually do not impact much the decision, hence causing the IDS to not detect the `Infiltration`. What is more interesting, however, are two of



(a) SHAP explanation for an undetected `Infiltration`

`Infiltration` instance predicted as `Normal`
Anchor: CWR Flag Count $>$ 7.6579×10^{-6}
Anchor Precision: 1.0

(b) Anchors explanation for an undetected `Infiltration`

Figure 4.4 – Explanations obtained using SHAP and Anchors. Figure 4.4a and Figure 4.4b explain a single (different) `Infiltration` instance.

the most important features: `ACK Flag Count` and `Fwd Packets/s`. For any successfully initiated connection, its respective data flow will then have an `ACK Flag Count` equal to 1. Therefore, this is definitely not a characteristic specific to `Infiltration` attacks, and the positive importance of the feature can be quite misleading. On the other hand, although having a negative importance, the feature `Fwd Packets/s` having a value of 100000 is quite incoherent with regard to what is an `Infiltration` attack. With further investigation, it is possible to find that for the CIC-IDS2017 dataset, `Infiltration` attacks (normally quite stealthy) were used as a first step to launch port scans, which can generate much more packets. Therefore, this particular error in prediction might be due to a case of mislabeling, as highlighted in [97].

While SHAP allows to show feature importance with regard to any class (normal traffic or any attack), Anchor explanations are given with regard to the predicted class. Therefore, in Figure 4.4b, the condition given represents a sufficient condition to, in this case, predict the given instance as normal traffic. Therefore, the condition, and the anchor precision being equal to 1 means that whenever the `CWR Flag Count` is equal to 1, the IDS will consider this instance, and any instance satisfying the same condition, as normal traffic. This is problematic because CWR flags are generally used alongside ECE flags to

inquire on the ability to notify about congestions. Therefore, any attacker getting hold of this information would be able to attack without being detected by the IDS by crafting packets to add CWR flags. This points out to potential limits of the IDS and enriching the training dataset is thus necessary to prevent such limits from being exploited.

4.2 Evaluation of XAI methods

By using and testing different XAI methods in this thesis, we realized that methods using similar explanation mechanisms could still produce vastly different explanations for the same situation. It thus raised a few questions:

— Do these explanations all identify true causes of an event?

— Are some explanations more "right" than others?

— Are there other important considerations?

While XAI is a research topic that is quickly gaining traction, it is still relatively new. Therefore, there are areas where XAI is less developed, which can bring about significant limits. One such limit is about how to evaluate explanations. Although this has been researched, there is no consensus yet about how to properly evaluate explanations.

In [47], evaluation of XAI is separated between human-grounded, application-grounded and functionally-grounded evaluations. The evaluation method should thus correspond to the objective researched. In [75], authors consider evaluation of explanations more as a subjective scoring of the explanation's quality compared to expectations. Finally, in [126], twelve essential properties of explanations are presented, separated into three main categories: content, presentation and user. However, XAI methods are still lacking evaluation metrics used as a standard [71], especially in the context of IDSs [127, 187]. As highlighted in [33], what represents a "good" explanation, particularly in the context of IDSs, is difficult to define and different stakeholders might have different needs.

Nevertheless, a number of quantitative metrics have been developed in an attempt to evaluate various properties of XAI methods such as Faithfulness, Robustness or Complexity [156]. Faithfulness of an explanation is a desirable property that describes the ability to capture the features used by a predictor [21], but can be quite complex to compute. However, other methods can also be used to compute metrics related to Faithfulness [103, 172]. Robustness measures the stability and consistency of a given XAI method [99, 151], while Complexity [99] ensures that explanations would be easily understandable by users. Finally, it can be interesting to measure how explanations coincide with ground truth

[173, 151].

In this thesis, we chose to focus on the properties defined in [126] to evaluate explanations, since it is quite comprehensive. While user-based properties are subjective and should be evaluated by questioning users and presentation-based properties are about controllable characteristics, content-based properties can be used to potentially define metrics for XAI evaluation. Among the possible properties, Correctness, i.e., how faithful is the explanation with regard to the model, and Completeness, i.e., how much of the decision process in explained, are the most meaningful properties with regard to the actual quality of an explanation. We thus developed an implementation of these properties to quantitatively evaluate explanations.

4.2.1 Compute Correctness and Completeness of explanations

From another viewpoint, Correctness measures the ability of an explanation to represent "nothing but the truth". Therefore, most important features in the explanation should should have a higher significance towards the predicted class. For example, a very high value of `Packets/s` might not be the most "correct" feature to explain a DoS attack, since it could be more relevant to predict a Brute Force attack. On the other hand, Completeness measures its ability to represent "all the truth". An explanation showing an abnormal value for the sensor of a particular industrial system component might be enough to explain a cyberattack targeting this component, if there are no other cyberattack targeting the same component.

Both properties combined can already give a quite comprehensive view of how the explanation portrays the decision process of an IDS. However, the quality of explanations presenting feature importances is heavily dependent on the number of features used. Ideally, a cybersecurity expert would have to handle explanations with only a few features, to close in on the real causes of an event more quickly. Therefore, it was decided to look at how Correctness and Completeness of an explanation change as the number of features used to explain increases.

Completeness The easiest way to measure that an explanation represents "all the truth" is to "remove" (replace values by median values) features not present in the explanation, and see if the prediction of the IDS changes. An algorithm describing the process is presented in Algorithm 1.

Algorithm 1: Completeness computation

Data: IDS the trained IDS,
$\mathcal{P}_{IDS}$ the probabilistic output of the IDS,
X the dataset,
x a sample of the dataset to be explained,
$features_{imp}$ the important features returned by the explanation

$pred \leftarrow argmax(\mathcal{P}_{IDS}(x))$
$incomplete\_x \leftarrow median(X)$
$incomplete\_x[features_{imp}] \leftarrow x[features_{imp}]$
if $argmax(\mathcal{P}_{IDS}(incomplete\_x))$ *is* $pred$ **then**
    `// Explanation was complete`
    return True
else
    return False
end

Correctness Correctness, which measures the fact that an explanation is "nothing but the truth", is more difficult to compute than Completeness. To do so, we decided to incrementally "remove" features and measure the impact it brought to the probabilistic output of the IDS. If the change in output was the highest for the predicted class, it means the explanation was correct. The algorithm used to compute Correctness is presented in Algorithm 2.

4.2.2 Datasets, ML and XAI algorithms for evaluation

In order to test the influence of the number of features as well as the performance of the IDS on the quality of the explanations, three datasets were used: WADI [6], CIC-IDS2017 [148] and UNSW-NB15 [123]. All three datasets were split using a stratified scheme into 70% train (60% and 10% validation) and 30% test sets.

Both CIC-IDS2017 and UNSW-NB15 were pre-processed in the same manner as in Section 3.2 (49). For the WADI dataset, features such as Row, Date, Time and four other features that are missing all values were removed. The resulting dataset has 124 features. Attacks are named `Attack_i` (i from 1 to 15, e.g., `Attack_1`) and have different targets and objectives. They can either affect physical equipment, starting pumps, opening or closing valves, or manipulate sensor readings.

For the ML algorithm used as an IDS, NN algorithm is retained as a first experiment

Algorithm 2: Correctness computation

Data: IDS the trained IDS,
$\mathcal{P}_{IDS}$ the probabilistic output of the IDS,
X the dataset,
x a sample of the dataset to explain,
$features_{imp}$ the important features returned by the explanation,
max_di is an array conserving the maximum change in probability brought by deleting a feature, for each class

$P \leftarrow \mathcal{P}_{IDS}(x)$
$pred \leftarrow argmax(P)$
$previous\_P \leftarrow P$
$incomplete\_x \leftarrow x$
// $|max\_di|$ = $|classes|$
$m = median(X)$
$max\_di \leftarrow [0, 0, ..., 0]$
for $feature$ *in* $features_{imp}$ **do**
  $incomplete\_x[feature] \leftarrow m[feature]$
  $incomplete\_P \leftarrow \mathcal{P}_{IDS}(incomplete\_x)$
  // Unit-wise division
  $deletion\_impact = \frac{P}{incomplete\_P}$
  for $i \leftarrow 0$ *to* $size(deletion\_impact)$ **do**
    $max\_di[i] = max(deletion\_impact[i], max\_di[i])$
  end
  $previous\_P \leftarrow incomplete\_P$
end
if $argmax(max\_di)$ *is* $pred$ **then**
  // Explanation was correct
  return True
else
  return False
end

for multiple reasons. First, it is often one of the best performing algorithms. Secondly, NNs are also among the less inherently interpretable ML algorithms, thus the interest in explaining their predictions. The NN architectures used are those obtaining the overall highest Accuracy on the three datasets. They are fully connected with six hidden layers of size 256, 512, 1024, 512, 256, 128, with a ReLU activation function.

To explain NNs, as well as compute Completeness and Correctness, LIME is used. This particular method has been chosen over other methods because this is the most extensively used compared to other similar feature importance methods, along with SHAP, but is more computationally efficient. Raw values of feature importances returned by LIME are used to compute XAI metrics, as shown in Algorithm 1 and Algorithm 2, where feature importances are given with the parameter $features_{imp}$.

4.2.3 Completeness and Correctness results

First, because performance of the IDS might impact the performance of XAI methods, it is important to evaluate the IDS to get an idea of its ability to detect different cyberattacks. Therefore, detection rate of the attacks mentioned will be reported to ease reading (Table 4.2, Table 4.3 and Table 4.4).

For each instance in the test set, the IDS made a prediction, then an explanation of this prediction was provided by LIME using a high number of features, which in this case we chose to be 50. Since features are ordered by importance, it is possible to remove the last features to reduce the size of the explanation. In this way, Correctness and Completeness can be computed for different numbers of features. Therefore, for each data point, it is possible to know if its corresponding prediction's explanations (containing from 3 to 49 features) satisfy these two properties. Since both properties return a `True` or `False` result, Figure 4.5 shows the proportion of instances having `True` results.

Table 4.2: NN Detection rate of specific attacks on WADI

Attack	**Attack_8**	**Attack_9**	**Attack_13**
Detection Rate (%)	97.5	88.8	85.2

Figure 4.5a shows Completeness results on the WADI dataset. First, Completeness seems to be correlated with the number of features in most cases, with a higher number of features meaning a more complete explanation, which is not really surprising. Secondly,

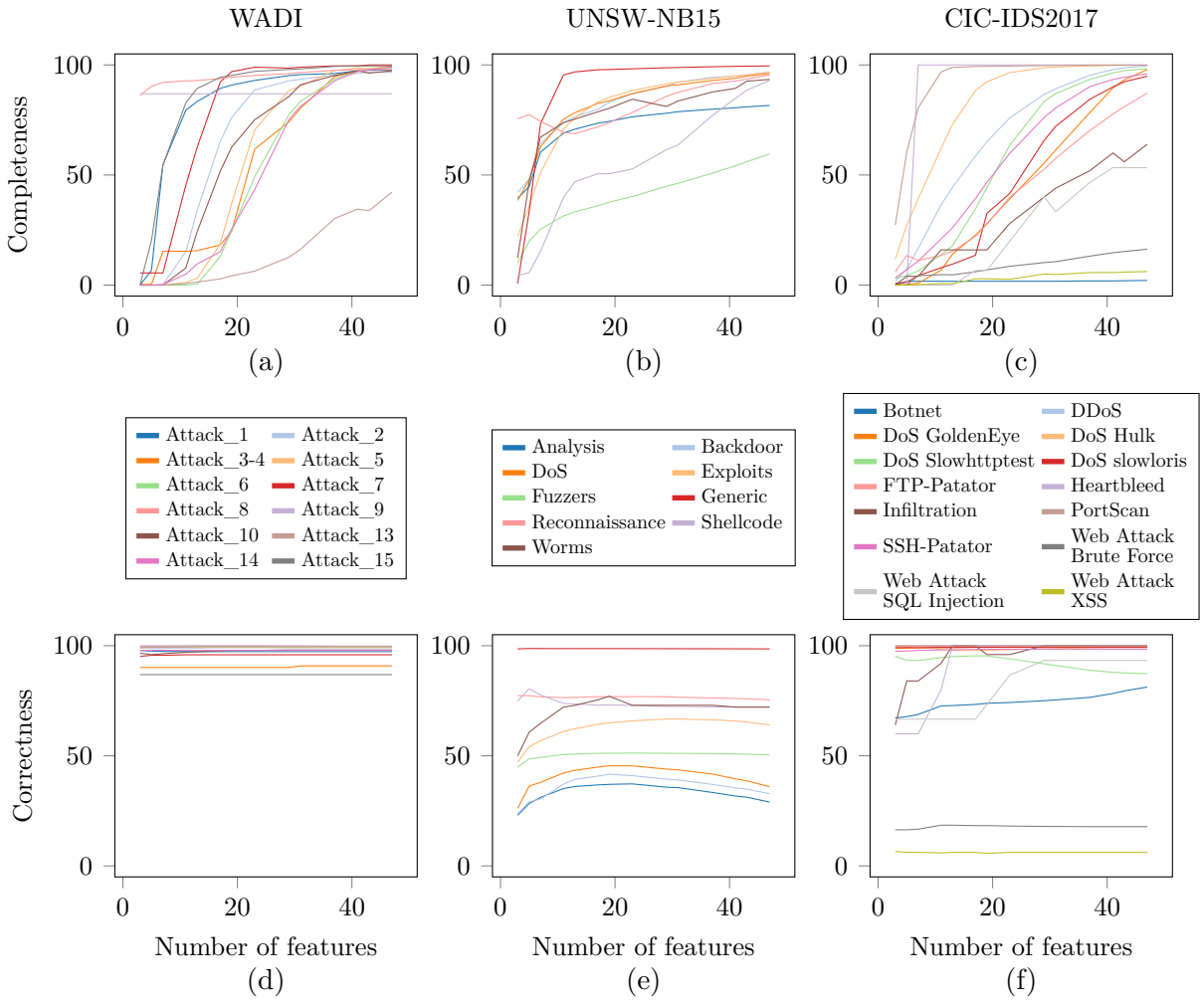


Figure 4.5 – Completeness and Correctness results for each class on WADI (a) and (d), UNSW-NB15 (b) and (e), and CIC-IDS2017 (c) and (f). Adapted from [49].

there are two exceptions to the previous observation: `Attack_8` and `Attack_9` have a very high Completeness value even with the lowest number of features returned. `Attack_9` (turns on a pump) is relatively simple to detect considering the feature `1_P_006`'s (showing the status of the pump) value that is normally 0, is 2 when that attack occurs. However, in some cases during `Attack_9`, the feature `1_P_006`'s value remains 0, causing both a prediction error, and the explanation to not be complete. In the case of `Attack_8` (opens the Motor Control Valve, MCV_007), a high enough value ($value \geq 30$) for `2_MCV_007_CO` (showing opening percentage) also seems to be the only important feature. Finally, `Attack_13` (reduces a booster set point pressure) explanations are often not complete, because this attack is weakly impacting the whole water plant, thus needing many features to be detected.

In Figure 4.5d showing Correctness on WADI, results seem to not be correlated with the number of features. Moreover, and interestingly, Correctness results for each class tend to be relatively close to the IDS's ability to detect a specific attack, i.e., the proportion of instances that have an explanation with Correctness being `True` is similar to the proportion of instances that are correctly classified.

Table 4.3: NN Detection rate of specific attacks on UNSW-NB15

Attack	**Analysis**	**Backdoor**
Detection Rate (%)	2.8	3.4

In the case of the UNSW-NB15 dataset, as shown in Figure 4.5b, explanations generally need less features to be complete than for other datasets. This probably means that some features in this dataset offer more discriminating power with regard to detecting some attacks. However, Completeness is often above the detection rate of some classes, e.g., `Backdoor` whose detection rate is very low and Completeness reaches quickly above 60%, which means that although important enough that the IDS is certain of its prediction, these features are misleading and often push the IDS towards the wrong prediction. For Correctness, increasing the number of features seems to have a negative impact for some attacks, which seems a peculiar behavior. The most likely possibility is that some less important features in these cases tend to be very important for other classes. Last but not least, similarly to Completeness, Correctness values are above the detection rate of many classes, which means the IDS is often confidently wrong, e.g., detection rate of

`Analysis` and `Backdoor` is notably low (Table 4.3) whereas Correctness is higher than 25%, which is concerning.

Table 4.4: NN Detection rate of specific attacks on CIC-IDS2017

Attack	**Web Attack SQL Injection**	**Web Attack Brute Force**	**Web Attack XSS**	**Botnet**
Detection Rate (%)	0.0	13.2	2.1	63.4

For CIC-IDS2017, increase rate in Completeness seems disparate for the different attack classes. This might mean that attack complexities differ a lot for this dataset. Completeness remains close to 0% for `Web Attack XSS`, `Web Attack Brute Force`, and `Botnet`. For `Botnet` where the detection rate is around 60%, it probably means that a combination of many features is generally required to predict correctly this class. For `Web Attack XSS` (and `Web Attack Brute Force`), it is probably caused by the low detection rate (Table 4.4). Interestingly, the IDS also seems often confidently wrong in the case of `Web Attack SQL Injection` where Completeness reaches around 50% whereas detection rate is 0%. The same behavior seems to be validated by Correctness results for the different Web Attacks.

Overall, Correctness does not seem to be impacted much by the number of features. It means that features in the explanations, especially the most important ones, have the highest impact on the class predicted by the IDS, so explanations indeed properly reflect the IDS's decision process. Moreover, Correctness also seems to be highly correlated with performance on the different classes. This is important because it possibly means that incorrect explanations might be due to incorrect predictions, thus allowing to find IDS errors. Completeness, however, is very dependent on the number of features, except in cases where one or a few features make the prediction too obvious, e.g., `Attack_9` in WADI. Performance also seems to have an impact, e.g., `Web Attack XSS`, `Web Attack Brute Force` in CIC-IDS2017, although the impact is lower than for Correctness. This metric also shows that many attacks are generally complex and require many features to be properly explained. Therefore, relying on explanations to explain and validate predictions might be more time-consuming than expected for a human operator.

Finally, a concerning observation is that according to both metrics, explanations better reflect the behavior of the IDS when they contain more than 20 features, which is a lot of information if used as explanations sent to cybersecurity experts.

4.2.4 Correlation between Completeness, Correctness and IDS results

An interesting observation is that the IDS performance seem to heavily influence the ability to obtain complete and correct explanations. Attack classes that are poorly detected generally lead to their explanations being incorrect and often also incomplete, e.g., `Web Attack XSS` and `Web Attack Brute Force` explanations are almost all incorrect and incomplete, regardless of the number of features.

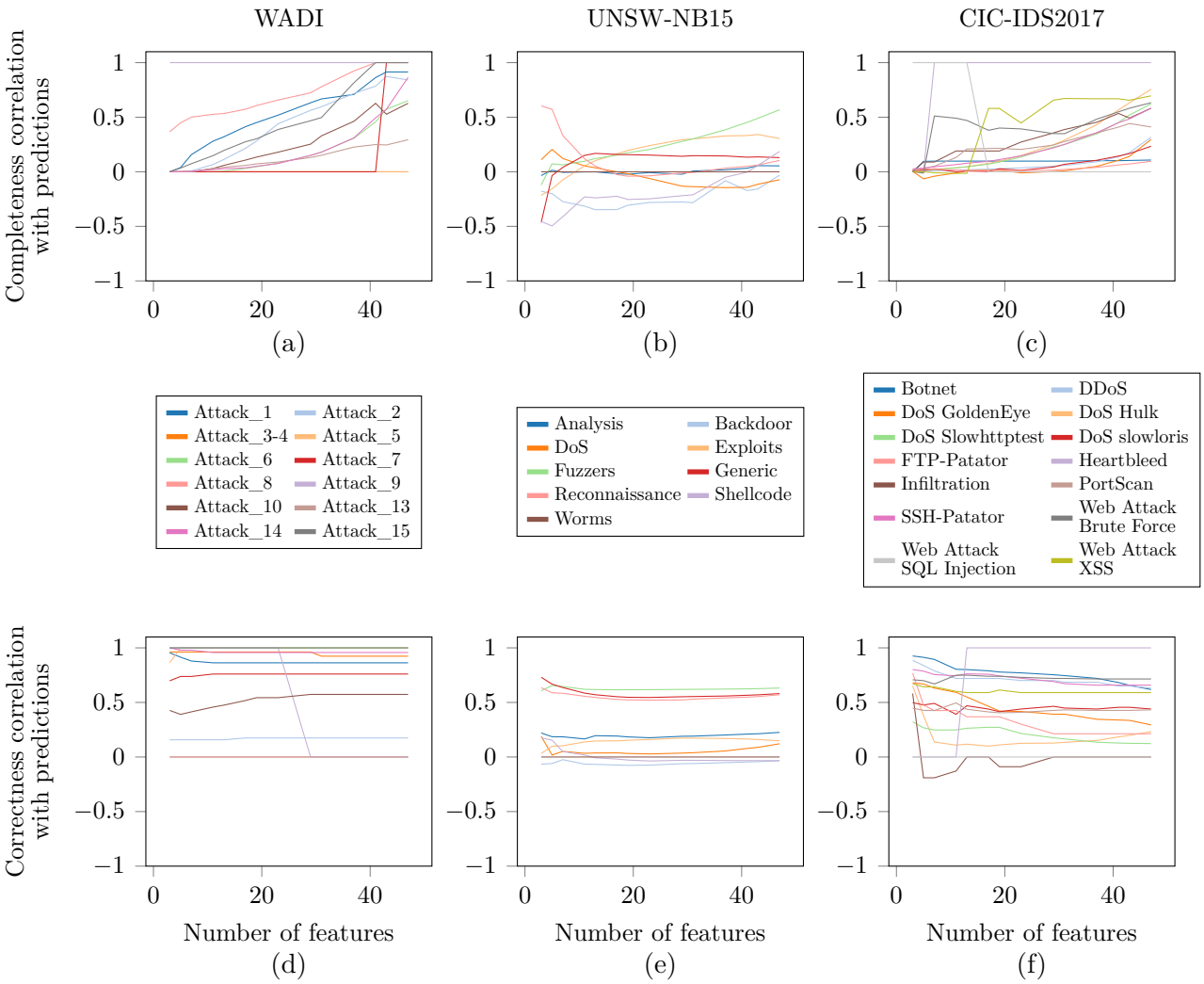


Figure 4.6 – Completeness and Correctness correlation with predictions for each class on WADI (a) and (d), UNSW-NB15 (b) and (e), and CIC-IDS2017 (c) and (f). Adapted from [49].

Correlation is positive in two different cases: Completeness (or Correctness) is `True` and the IDS's prediction is correct, or Completeness is `False` and the IDS was mistaken. Correlation is thus negative in the other cases. Figure 4.6 shows the correlation between Correctness and Completeness with predictions.

Because Correctness seemed to be often similar to the Recall of each class, it is interesting to explore correlation between both metrics and prediction results of the IDS. In case of a high correlation, both metrics might be useful in detecting errors in prediction. Correlation with Completeness might provide additional information, especially at a high number of features, or when some features are by themselves the determining factor, e.g., for `Attack_8` and `Attack_9` in WADI. Results on the train set are shown in Figure 4.6. For WADI, in two cases (`Attack_9` and `Attack_13`), correlation between Correctness and prediction results is equal to or reaches 0 because performance on the class is 100%, thus lowering artificially the correlation to 0 as long as one instance's explanation is not correct, which is the case here.

Overall, for Completeness, correlation with prediction results seems very dependent on model performance. The lower the performance, the lower the correlation. However, for `Attack_8` and `Attack_9` in WADI, correlation is (or reaches) 100% which means finding cases where Completeness is `False` could be used to find and correct all errors in prediction. For Correctness, the impact of model performance seems lower, but nonetheless still present, and correlation seems positive, or even highly positive, for all three datasets. Both metrics could thus be used to point out errors in prediction and would be effective in different cases.

4.3 XAI to validate or correct predictions

To test the potential of both metrics to find errors on the test set, Completeness and Correctness are computed for uncertain predictions (uncertain means the probability of the second most likely class is superior to a threshold, to reduce unnecessary computation).

As stated in Chapter 2, LIME gives explanations by creating samples in the surrounding of the sample to explain, then training a linear regression model using this new data. An important thing to note is that LIME can provide explanations for a single instance with regard to all possible classes, e.g., a `DoS` instance in UNSW-NB15 is predicted as `Fuzzers`, it is still possible to have important features for this instance with regard to other classes. Therefore, it is possible, for a single instance, to decide for which class its

explanations are the most correct and complete: a `DoS` predicted as `Fuzzers` might still have a better explanation for the `DoS` class.

4.3.1 Point out potential errors

The first possibility is to leverage Correctness and Completeness of explanations to find if some predictions might be erroneous. In this case, the potential errors might be reported to a human operator that should investigate and has sufficient time to find the true class of the investigated sample. Therefore, the investigation is considered as an oracle and it is possible to measure the improvement in performance induced from pointing out errors that will be corrected. While it is possible to compute Completeness and Correctness for each instance, it might induce a high computation overhead (more than 10 times the time for inference). It is therefore necessary to try to reduce the number of required computations, possibly without missing too many errors. It was thus decided to focus on IDS predictions that are more uncertain: predictions where the output probability of the second most likely class is between 0.05 and 0.49. Values between these two values will thus be used as a threshold to define uncertain predictions for which explanations and their Correctness and Completeness will be computed.

To this end, the number of features returned by explanations needs to be fixed for both Completeness and Correctness. It has been fixed at 40 for Completeness, because correlation prediction results seem higher with more features. For Correctness, it has been fixed at 30, although the number does not matter much, since correlation seems to not depend on the number of features. Detection rate at 0.5 in threshold value thus represents the original detection rate.

Finally, there are multiple ways to leverage Completeness and Correctness to look for and identify errors. Since both tend to be highly correlated with detection rate of attacks, it is possible to point out errors when either (instead of both) Completeness or Correctness points towards an error. However, while it might be more interesting, performance-wise, to find more errors by performing more investigations, this might lead to a high increase in required investigations, which could prove to be counterproductive. Therefore, following our own experiments, we decided to use the two metrics to point out potential errors if both are in agreement against the prediction of the IDS, as shown in Algorithm 3.

As the threshold values, i.e., the values compared to the IDS's probabilistic outputs to determine prediction uncertainty, decreases, Figure 4.7a, Figure 4.7b and Figure 4.7c show the increase in detection rate, while Figure 4.7d, Figure 4.7e and Figure 4.7f show

the number of potential errors that will require investigations. We can see that manual investigations are mainly required for the `Normal` class. This is expected because many attacks are missed and classified as `Normal`, thus the need to investigate this class.

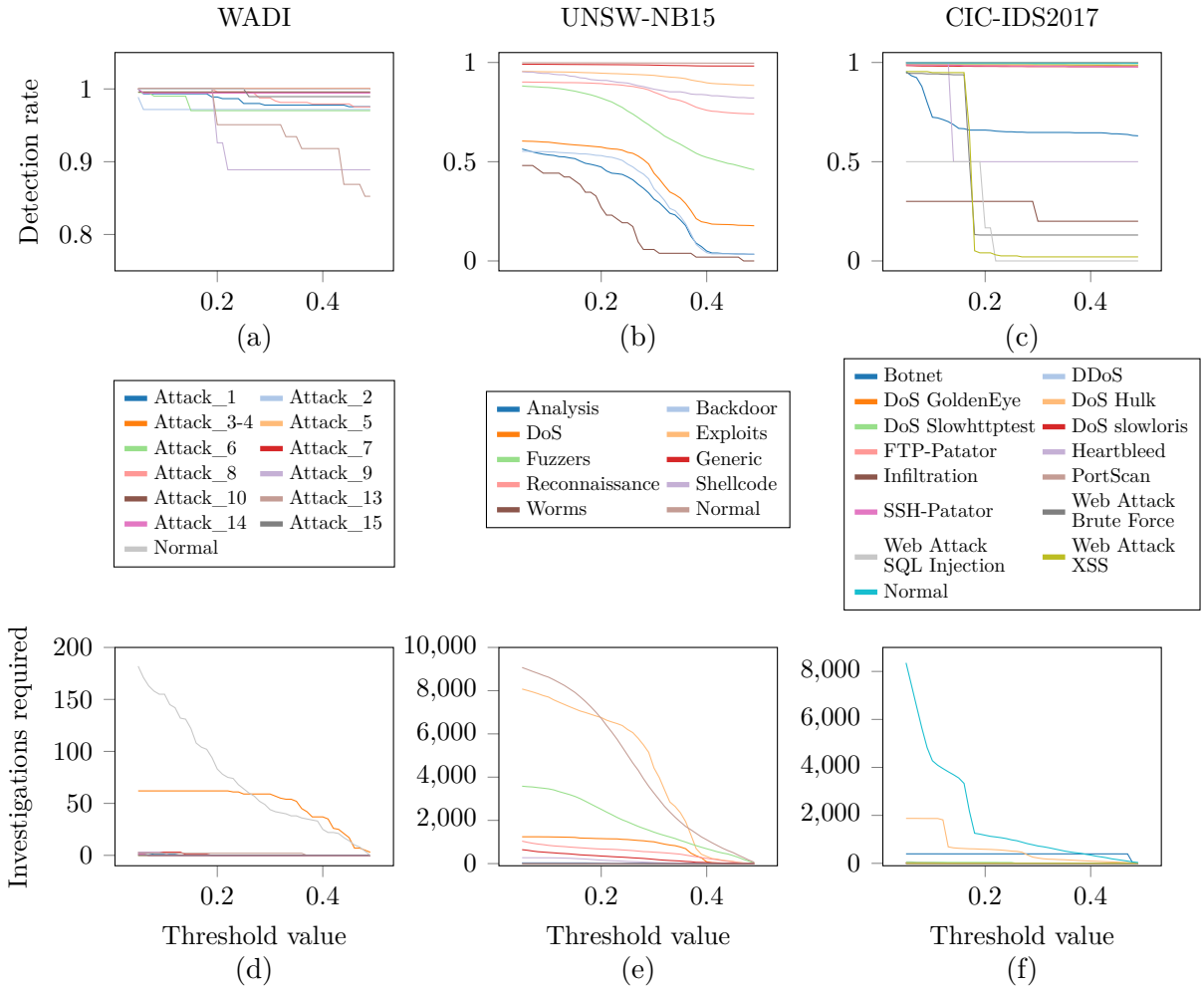


Figure 4.7 – Possible gains in Accuracy (a, b, c) and Manual investigations required (d, e, f) for each class, with XAI pointing errors.

On the WADI dataset, although performance was already very high for all classes, it is interesting to see that using Completeness and Correctness to point out errors allows to find most of the remaining errors, thus achieving near perfect detection of attacks.

On the UNSW-NB15 dataset, although the possible increase in performance is much more important, this also requires a non-negligible amount of investigations. Indeed, more than 20000 investigations would be required, mainly `Normal` and `Exploits` instances,

because incorrectly classified attacks are generally assigned to these two classes. For the `Exploits` class, this is because this attack class is more prevalent and much more diverse than other attack classes, and thus they share similarities. Unfortunately, although the gain in performance is significant, this amount of investigation is probably not manageable by cybersecurity teams. In this case, it might be more interesting to focus on possible errors in `Exploits` predictions, since performance on this dataset shows that most misclassified attacks are confused with `Exploits`.

On CIC-IDS2017, finally, the overall increase in performance is similar to UNSW-NB15, while the required amount of investigations is significantly lower. By pointing out potential errors with XAI for the CIC-IDS2017 dataset, there is a definite improvement for many classes compared to only using an IDS. For attacks where performance was originally low (below the 75% mark), the gain in Accuracy ranges between 5-10% to even 90% for `Web Attack XSS`. Interestingly, pointing out errors also induces a high number of investigations for `DoS Hulk` predictions. The class often possesses similar signal to other DoS attacks, which explains why the probability of other DoS attacks is often non-null when predicting `Dos Hulk`. However, the IDS is mostly correct, thus resulting in wasteful investigations.

As could be seen with UNSW-NB15 and CIC-IDS2017, while there is a notable increase in performance, the amount of investigations required is probably too high for being doable by a team of cybersecurity experts, although this represents less than 10% of the uncertain predictions. Therefore, there is a need to find a correct trade-off between performance increase and human workload, or to find other methods with a lower need for human efforts.

4.3.2 Automatically correct predictions

While it is possible to improve performance by pointing out errors and performing more investigations by cybersecurity experts to correct IDS mistakes, it is also possible to automatically correct predictions of the IDS, albeit with a lower increase in performance than human corrections.

However, to do so, it is not enough to find an incomplete or incorrect information. It is also required to identify, in case of an error, which class would be the most likely to be the correct class. Fortunately, LIME offers a possible solution: it is possible, for a single instance, to obtain explanations and thus feature importances with regard to all possible classes. Therefore, it is also possible to compute Correctness and Completeness for these

different explanations to find the most correct and most complete explanation. This can, in turn, point towards the actual class of the explained instance.

However, to extend Completeness and Correctness computations to be able to find the most likely class, there are a few obstacles. First, since attacks are generally more complex than normal traffic, Completeness is biased towards normal traffic: even if the actual class is that of an attack, it is highly probable that the most complete explanation would be the explanation for normal traffic. Secondly, how to decide if both the most complete explanation and the most correct explanation correspond to a different class, that is also different from the IDS's prediction? While it might tell that the prediction of the IDS is indeed erroneous, it does not help in finding which class is the correct one. While Completeness shows which class is best characterized by only a few features, Correctness merely allows to ensure that important features are indeed important to the decision. Therefore, we decided to give priority to the class corresponding to the most complete explanation in case of a disagreement. Completeness was prioritized because in case of disagreement, Correctness alone tended to often incorrectly change the prediction, whereas Completeness tended to more conservatively change predictions, and was more often correct. The final algorithm used to perform the automatic correction of predictions leveraging Correctness and Completeness is shown in Algorithm 4.

Results obtained using this fully automated method are shown in Figure 4.8. On WADI, similarly to the more manual method, the algorithm is able to achieve a better performance and can correct most of the remaining errors. Similarly, for CIC-IDS2017, there is a notable increase in performance, at virtually no cost, although it increases to a lesser extent than the manual method. Finally, contrary to the other datasets, there is also a decrease in performance on UNSW-NB15. While the automated correction allows to better detect `Fuzzers` and `Analysis`, there is an almost symmetrical decrease in performance for `Shellcode` and `Backdoor`. This shows that features that are significant between these classes, although this does not seem to lead to the IDS initially mistaking one for the other (confusion matrices does not show confusion between these classes).

In this case, the ability to correct predictions is fully automated and only induces a computation overhead, yet is still able to notably increase performance of the IDS. Most importantly, it could possibly be combined with the "manual" method to possibly improve performance further by relegating potential errors that are difficult to automatically correct to cybersecurity experts, thus improving performance with a manageable human workload.

Algorithm 3: Algorithm to find and point out errors

Data: IDS the trained IDS,
x a sample of the dataset to explain,
`Completeness` the function to compute Completeness,
`Correctness` the function to compute Correctness,
$features_{imp}$ the important features returned by the explanation

$complete \leftarrow$ `Completeness`(IDS, x, $features_{imp}$)
$correct \leftarrow$ `Correctness`(IDS, x, $features_{imp}$)
if *not correct and not complete* **then**
  `// Point out error for investigation`
else
  `// Do nothing`
end

Algorithm 4: Algorithm to automatically find and correct errors

Data: IDS the trained IDS,
`md` the median of the training data,
x a sample of the dataset to explain,
`Most_cmp_expl` the function to find the class with the most complete explanation,
`Most_cor_expl` the function to find the class with the most correct explanation,
$A_{features_{imp}}$ the array containing the important features returned by the explanation for each class

$pred \leftarrow$ IDS(x)
$median\_pred \leftarrow$ IDS(`md`) `// Generally is normal traffic`

$pred\_complete \leftarrow$ `Most_cmp_expl`(IDS, x, $A_{features_{imp}}$)
$pred\_correct \leftarrow$ `Most_cor_expl`(IDS, x, $A_{features_{imp}}$)
$new\_pred \leftarrow pred$
if $pred\_complete \neq median\_pred$ *and* $pred\_complete \neq pred$ **then**
  $new\_pred \leftarrow pred\_complete$
end
else if $pred\_correct \neq pred$ **then**
  $new\_pred \leftarrow pred\_correct$
end
return new_pred

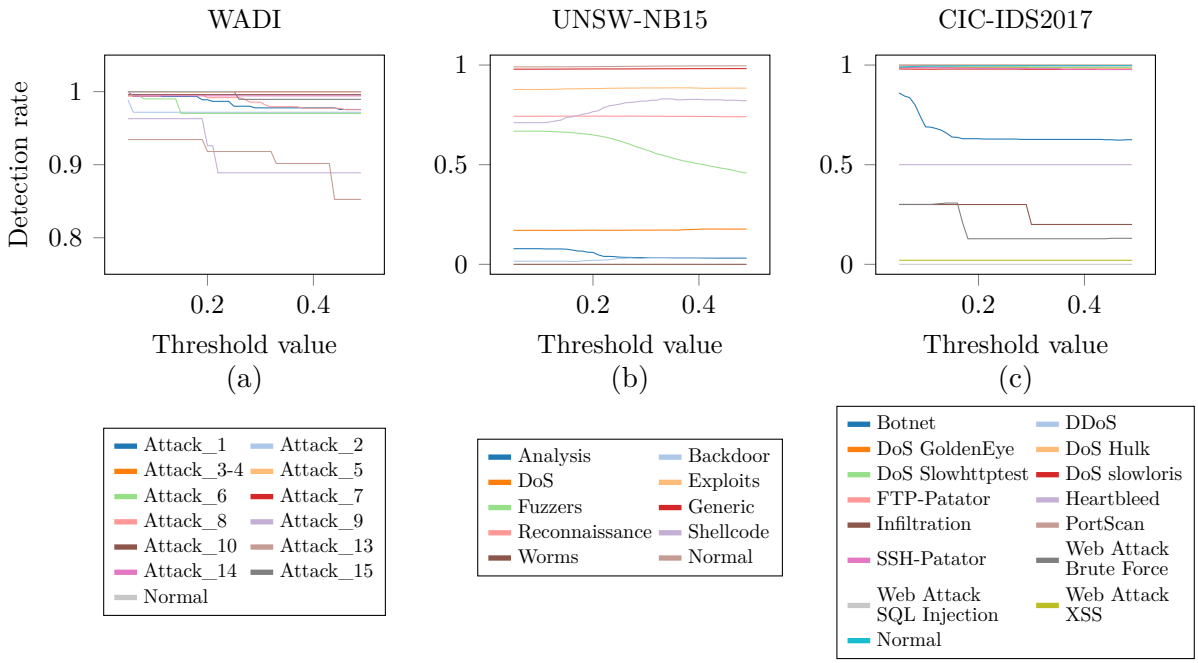


Figure 4.8 – Possible gains in Accuracy for each class, with XAI used to automatically validate or correct predictions.

4.4 Conclusion

XAI has made significant progress in the last decade, but is not yet mature enough to properly explain the decision process of IDSs. One of the main reasons comes from the fact that commonly used XAI methods rely, in one form or another, on dataset features. Unfortunately, while this works in some application domains, e.g., images, where an area of the image can represent a meaningful concept, there is often a gap between human knowledge of cyberattacks and how the dataset features are constructed for IDS datasets. This leads to XAI providing explanations that are more difficult to understand for humans, including for cybersecurity experts. Therefore, to leverage XAI methods in order to provide explanations of a decision, it should be necessary to reflect on how to appropriately construct datasets, with XAI in mind. Another possibility could be to rely on textual explanations or more generally generative AI, but as has been shown with ChatGPT, this could lead to another set of security problems [52].

Going further, XAI is still lacking a proper evaluation framework, depending on the application and who receives the explanation. Further research on this topic could definitely improve XAI and lead to more useful research.

Experiments performed in the context of this thesis also showed that explanations are heavily influenced by the performance of the explained ML algorithm. In our case, cyberattacks that were hard to detect generally lead to both incomplete and incorrect explanations, which highlights the fact that the IDS is predicting somewhat hazardously.

Finally, we showed that although not yet ready to explain behaviors of an IDS, XAI can still be leveraged to verify its decisions and improve its performance. Two methods, one more manual and the other more automatic, lead to a significant increase in performance. While the manual method requires a significant workload of human experts to investigate and correct potential errors, it also leads to near perfect detection of cyberattacks in many cases on some public datasets. The automatic method, although leading to a lower increase in performance, does not require any interaction and is able to correct erroneous predictions at the cost of more computation time. Finally, although not developed in this thesis, it might be interesting to focus on a hybrid methodology, relying mostly on automated corrections while pointing out really uncertain predictions for human experts to investigate. This could lead to a more significant increase in performance, with a manageable human workload.

Chapter 5

Zero-day attack detection and classification

While there are different methods to make Machine Learning (ML)-based Intrusion Detection Systems (IDSs) more adapted to the cybersecurity domain and more trustable, such methods will probably not completely replace signature-based approaches, because of their differences and respective advantages. As such, both systems could definitely be used concurrently, to benefits from both approaches' advantages and cover for the other approach's drawbacks.

One of the main drawbacks of current signature-based approaches is their perceived inability to detect Zero-Day Attacks (ZDAs). Therefore, developing ML-based IDSs that are able to better detect ZDAs could definitely benefit the current state of IDSs. However, current ML-based IDSs rely mainly on supervised ML methods because of their high performance, but this simultaneously impairs their ability to detect anything unknown [96]. While unsupervised methods allow to detect ZDAs, their ability to detect attacks, and especially differentiate them, is generally much lower than that of supervised methods.

More recently, the development of semi-supervised methods allowed to leverage both labeled and unlabeled data to achieve performance similar to supervised methods, while being more flexible and allowing to detect anomalies.

5.1 Paradigms for the detection of Zero-Day Attacks

As stated previously in Section 2.5.1 (Page 36), ML methods can be separated into different types of methods, among which some are naturally more suited for the detection of ZDAs.

5.1.1 Anomaly detection methods

The first methods employed to detect ZDAs were mainly anomaly detection methods. They could be either unsupervised or supervised methods. The goal of these methods is to distinguish outliers, or anomalies, from a normal behavior. Since there is a single anomaly class, detection of ZDAs is often as good as detection of possibly known attacks. However, the overall detection ability falls short compared to the detection ability of supervised methods for known classes.detection of all attacks is often much below other supervised methods performing classification. In [119, 20], multiple unsupervised anomaly detection methods were used: One-Class Support Vector Machine (OC-SVM), Isolation Forest (IF), Local Outlier Factor (LOF), Auto-Encoders (AEs). While they are able to detect anomalies, it is also shown that Recall is often much higher than Precision which would lead to a higher rate of false alarms [119].

In [190], a thorough evaluation of different anomaly detection approaches, both supervised and unsupervised, is performed on 11 datasets. These datasets are used to create dataset variants with each having excluded a different attack class from the training dataset. Algorithms tested include: Support Vector Machine (SVM), k-Nearest Neighbors, Logistic Regression, k-Means, LOF, OC-SVM, IF, and ensembles of these algorithms. Performance on both known and unknown attacks is reported, and it is shown that supervised methods are unsurprisingly better at detecting known attacks, while unsupervised methods are better at detecting unknown attacks. Results averaged over all dataset variants show that the best performing algorithms are able to detect a bit over half of the unknown attacks. While this might be interesting in the case of ZDAs, their performance in detecting known attacks remain much lower than state-of-the-art supervised classification algorithms [1].

Unsupervised anomaly detection approaches are interesting to detect attacks, including ZDAs, when no labels are available. However, they are unable to detect and differentiate between attacks, and they usually produce a high number of false alarms. As such, approaches able to perform multi-class classification, while having the ability to detect unknown attacks, are preferable to improve detection of known attacks and reduce the quantity of false alarms.

1. In [190], unsupervised methods achieve around 60% Recall for most unsupervised methods while supervised methods can reach above 70% REcall.

5.1.2 Open-Set and Open-World Learning

The problem of a trained classifier being able to detect and classify new unknown classes is a very difficult problem that has been first formalized in [18], and is named Open-World Learning (OWL). It extends Open-Set Learning (OSL) [144] that considered all unknowns as a single anomaly class by considering multiple unknown classes. It also extends it by applying Incremental Learning to incrementally add in a supervised manner the multiple new classes that were detected. In [28], it is shown that although small steps are taken in the direction of OWL, mainly with advances in OSL, this is not sufficient and much remains to be done. Since then, recent works on image applications [31, 158] have shown that most recent ML methods are slowly gaining the ability to detect and distinguish unknown classes. Both OSL and OWL are preferable to anomaly detection approaches since they are able to correctly recognize known attacks with a much higher performance, while simultaneously having the ability to detect ZDAs, and possibly differentiate them in the case of OWL.

In intrusion detection and detection of new classes, much of the work has focused on OSL. In [143, 92, 39, 79], it is shown that using OSL methods can lead to detection of unknown classes with a rate ranging between 20% and sometimes up to 90%. However, these approaches only consider a single anomaly class, and sometimes are tested on relatively small datasets. Furthermore, they are always tested by leaving out one attack of the dataset, which restricts the distribution of unknown attacks to that of a single class. As such, this is unclear if and how these methods would scale when considering multiple different new classes, even if considering them as a single anomaly class.

Currently, approaches used to detect ZDAs have achieved a high performance on known attacks and are able to distinguish them from unknown attacks. They are, however, only able of classifying possibly different ZDAs as a single anomaly class. In order to properly update IDSs, and retain a high performance, once ZDAs are detected, they still need to be differentiated. While relying on human annotation is possible, this is also costly. Therefore, approaches that detect ZDAs and are also able to correctly classify them would significantly increase the update efficiency.

5.1.3 Contrastive Learning

As mentioned in Section 2.5.1 (Page 37), Contrastive Learning (CL) appears to be a possible solution to achieve a high detection of known classes while being able to also de-

tect and differentiate multiple unknown classes. However, it also requires some adaptations to properly conform to specificities of the intrusion detection problem. One such adaptation is the difficulty to define augmentations, and thus perform self-supervised learning, leading to the use of a supervised contrastive loss [90]. This supervised contrastive loss $\mathcal{L}_{supcon}$ is shown in Equation (5.1), where $i \in I \equiv \{1...N\}$ is the index of a sample in a batch of size N and represents the chosen anchor, y_i is the label of sample i and z_i is the output of the model for instance i. $A(i) \equiv I \setminus \{i\}$, and $P(i) \equiv \{p \in A(i) : y_p = y_i\}$ represents the set of positives (samples of the same class). z_p thus represent positives with regard to z_i and z_a are all outputs excluding z_i. Finally, τ is a temperature hyperparameter. The goal of this loss is to penalize when negative samples are closer to an anchor than positive samples. In order to compute distance between samples, the dot product is used in this paper.

$$\mathcal{L}_{supcon} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)} \tag{5.1}$$

As with any CL approach, the selection of positives and negatives can be further refined according to different heuristics. Current approaches generally try to select hard negatives (the negative closest to the anchor) [87], or hard negatives and easy positives (the positive closest to the anchor) [177]. Since the objective is to push negatives away from the anchor, hard negatives will be pushed further than other negatives, and thus contribute more to the contrastive objective and can accelerate learning. The approach developed in this thesis used the method described in [177]. The closer the hardest negative is from the anchor compared to the easiest positive, the higher the loss, which in turn changes more the internal representation of the Contrastive Encoder (CE).

An example of how CL works is shown in Figure 5.1. As training progresses, same class instances are brought closer and closer while they are separated from other classes. During a training step, the dataset is separated in different batches, and in each batch, every instance is used as an anchor (the three black points in Figure 5.1 illustrate going over every instance), with new positives and negatives being computed accordingly. The loss is then summed over the batch to update the CE's representation.

Finally, since CL is originally self-supervised method, it is also possible to leverage both unlabeled and labeled data to perform semi-supervised learning. While semi-supervised learning can be framed as in [67] to reduce the amount of labeled samples, it is also possible to consider semi-supervised learning when there are classes that are present in

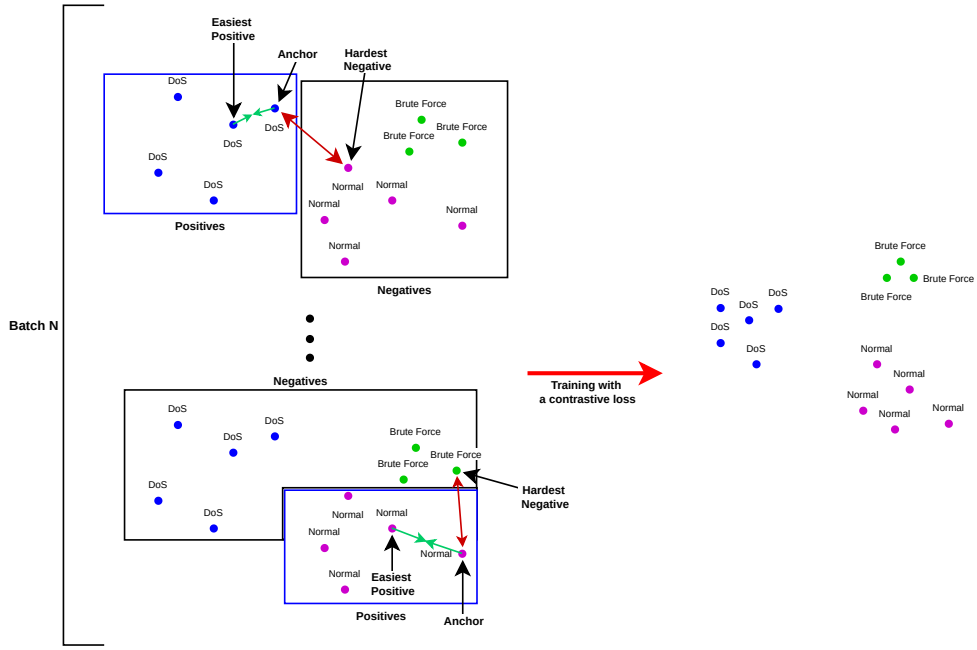


Figure 5.1 – Example of contrastive learning training on a single batch with anchors, positives and negatives.

the data but are unlabeled.

5.2 Practical implementation for IDSs

When using ML-based IDSs to detect multiple ZDAs in a real-world application, there are two possible scenarios:

— **Scenario 1**: They are completely new. In this case, the traffic never existed and the attacks can only be detected during testing.

— **Scenario 2**: They have been present for some time, and have remained unidentified. In this case, the traffic exists, is unlabeled but can still be used to train ML-based IDSs.

Both of these scenarios are very distinct and, ideally, both should be considered from the perspective of training ML-based IDSs. While ZDAs are more representative of the first scenario, they can also be present the second. Furthermore, in scenario 2, an IDS that was already trained using a labeled dataset should be able to be updated with unlabeled data to better differentiate ZDAs. This is similar to incremental learning, where new classes are incrementally learned, but extends incremental learning to use unlabeled data.

The approach developed in this thesis, SECL (for Sepmix rEgularized Contrastive Learning), consists of a CL algorithm to learn better representations that will be used by a combination of a k-Means and a k-Nearest Neighbors algorithm to cluster and assign a label to these representations. Additionally, it will use a three different regularization methods: dropout, Sepmix (for Separation through Mixup), and Von Neumann Entropy (VNE). These regularization methods help the approach in better detecting new classes.

The difficulty of the task in scenario 1 comes from the fact that an unknown number of classes do not exist in the training dataset, and they only appear during testing, as shown in Figure 5.2. The goal is thus to, along with correctly classifying known classes, detect all unknown classes and be able to classify them. In scenario 2, however, this unknown number of classes is present in the training dataset but is unlabeled.

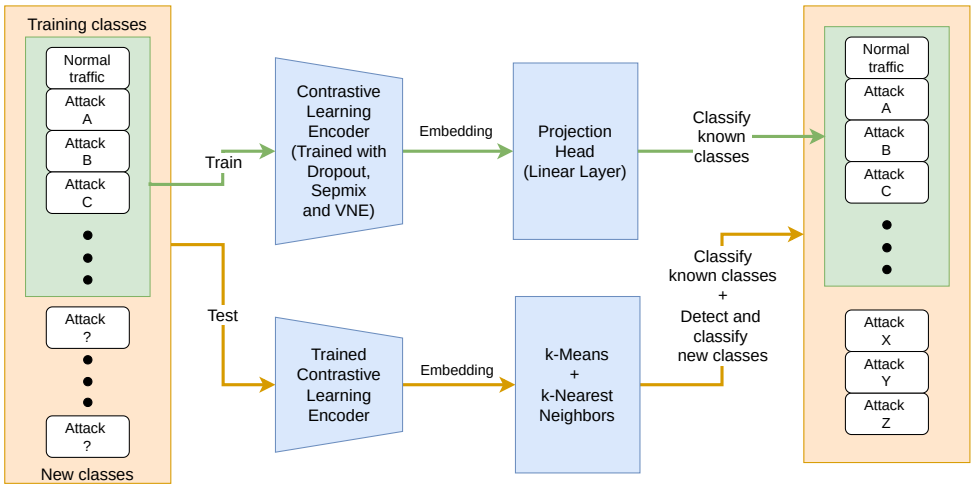


Figure 5.2 – Proposed approach with CL, Sepmix and VNE. The number of new classes is unknown during training. From [50].

To train Deep Learning (DL) methods, data goes through Neural Networks (NNs) as batches of multiple samples. CL approaches are very dependent on the size of batches, this influences the number of positives and negatives that can be used for computation. Batches are generally very big to increase diversity of samples for each class within a single batch. Unfortunately, the proportion of attack traffic is relatively low in intrusion detection datasets, which leads to batches often having a single sample or none at all of a specific attack. In these cases, the attack does not contribute to the loss and is much more difficult to learn because there are no positives within the same batch. Increasing the size of batches can quickly become impossible because of resource constraints. Therefore, a solution is to use a memory bank of samples that retains representations (outputs of the

CL model) of the samples of each class, as in [69, 120]. However, using memory in this way is not trivial, because this requires training of an additional encoder and is both slower and much more complex to train. Therefore, it has been chosen to use a simpler method for a more stable training process: update the memory with a given probability. Memory needs to be updated frequently to reflect the changes in the CE brought by the learning process, but also cannot be too frequent, because this, along with other regularization methods, can make training more unstable. After successive experiments, we settled on updating memory with a probability of 0.1 for each class. Memory is filled until a sufficient number N of samples of each class is in the memory bank. Experiments performed led to $N = 20$ to ensure stability of the training process.

During training, the proposed approach uses the supervised contrastive loss to circumvent the need for data augmentation, a memory bank to compensate for the high imbalance in the data and ensure that we can find a positive for every instance in a batch, Sepmix and VNE. The contrastive model is composed of a CE and a classification head. Since training is done using known classes with a supervised contrastive loss, a classification head (a simple linear layer) is used to project the representations into a layer that has the same dimension as the number of known classes, similarly to supervised DL algorithms. It is also conjectured in [36] that using a contrastive loss induces a loss of information. As such, using a classification head allows for the information to be lost mainly in the classification head, thus creating richer representations before projection.

During testing, this classification head is removed and replaced by a combination of a k-Means and a k-Nearest Neighbors algorithm to detect and classify an unknown number of classes. While a k-Means algorithm alone is able to detect and classify by assigning labels with the computed clusters, there is a need to replace these cluster labels by class labels. Therefore, the addition of k-Nearest Neighbors allows to assign labels to a cluster by determining the labels of samples closest to cluster centers and performing a majority vote. In a realistic context, this requires finding labels of a much lower number of instances to assign a class label to clusters.

5.2.1 Enhancing the ability to generalize: regularization methods

While a supervised contrastive loss benefits an IDS, by both allowing to circumvent the need to define data augmentation to perform CL, as well as allowing to achieve

performance equal to supervised approaches, it also greatly reduces the ability to detect ZDAs. One of the main reasons is that normal traffic being much more prevalent and its distribution often overlapping with the distribution of other classes, any ML approach tends to overfit on known attacks and defaults to identifying anything unknown as normal traffic. This will be mitigated using Sepmix, as described later in this section. Furthermore, learned representations tend to be of insufficient quality to properly differentiate unknown classes from known classes. Therefore, it is important to use regularization techniques in order to improve the ability of CL-based IDSs to properly detect new unknown classes. Both dropout and VNE will be used to increase the quality of learned representation by making the use of neurons inside the NN more balanced.

The mechanism of dropout [155] is a regularization method commonly used to train NNs. It randomly zeroes out different neurons in NNs at each step of the training process which reduces co-adaptation of neurons, i.e., the fact that neurons are activated by the same information. For CL, this essentially reduces the risk of representation collapse as well as improves the quality of all intermediate representations. In the proposed approach, as is often used in the literature, a dropout of 0.2, which removes around 20% of the links between neurons in the NN architecture, has shown the best performance.

VNE has been used in [91] and shows a high effectiveness in enforcing the use of all neurons in a NN, especially in the last layer, thus improving generalization. This effectively forces the CE to learn representations of higher quality that will be able to better differentiate unknown from known classes. Otherwise, unknown classes sometimes collapse into a single known class: normal traffic. In order to compute VNE, there are two steps involved. First, the autocorrelation $\mathcal{C}_{auto}$ of the representation matrix $Z = \{z_1, z_2, ..., z_N\}$, with z_i being the representation of x_i through CE and the classification head, is computed. Then, with λ_i being the i-th eigenvalue of $\mathcal{C}_{auto}$, the VNE loss is computed as in Equation (5.2).

$$\begin{aligned} \mathcal{C}_{auto} &= Z^T Z/N \\ \mathcal{L}_{VNE} &= -\sum_j \lambda_j \; log \; \lambda_j \end{aligned} \tag{5.2}$$

To reduce overfitting of CL approaches, Sepmix, that is inspired from Mixup is used. Mixup [184] is a method that helps in having a more linear behaviour in the space between different classes. Originally, Mixup creates new representations $\tilde{x}$ from the CE representations x, as well as new targets $\tilde{y}$ from labels y, as shown in Equation (5.3), by selecting

randomly two indices i and j, with $\lambda \sim Beta(\alpha, \alpha)$ being sampled for each pair of indices, with $Beta$ being the Beta distribution and $\alpha \in [0, \infty]$ being a hyperparameter.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \tag{5.3}$$

While Mixup encourages the NN model to behave linearly between samples and can improve generalization, this is not the reason it is used here. The goal here is to better separate different classes, and especially normal traffic from different attack classes. As such, Mixup is repurposed into Sepmix as shown in Equation (5.4), where i is the index of a randomly selected sample, and c is the closest sample that is of a different class. $\lambda \in [0, 1]$ is also fixed instead of being sampled from a $Beta$ distribution and becomes a hyperparameter. This decision was taken because our own experiments showed no notable increase in performance by sampling λ from a $Beta$ distribution in scenario 1, and reduced performance of SECL in scenario 2. The goal is thus to bring closer to $\tilde{x}$ any sample x_j if $y_j = y_i$ and push further away any sample x_k if $y_k \neq y_i$.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_k \\ \tilde{y} &= y_i \end{aligned} \tag{5.4}$$

Sepmix essentially leverages the memory of class instances to create virtual samples between classes that will be used in the supervised contrastive loss to make representations more compact. Figure 5.3 shows how memory is used to create new virtual instances that can change which data serves as easiest positives and hardest negatives. In scenarios similar to scenario 2, unlabeled instances can be present in the data and can be leveraged by sepmix to create new instances. Different color codes are used to show instances from different classes coming either from the memory or created through Sepmix. Once instances in the memory are added to batch data, closest instances to those from memory, but from a different class (either labeled or not), are used to create virtual instances of the same class as those from memory. As a consequence, this also allows to artificially increase the contrastive loss, and thus, through training using such a loss, increases the distance between samples of different classes, especially in the case of normal traffic that often overlaps with all other classes. In the case of scenario 2 where unidentified attacks exist in the training data but are unlabeled, they can be used as the closest sample from a different class with Sepmix, as shown in Figure 5.3 to create a virtual `DoS` instance, to improve performance and generalization. Therefore, Sepmix allows to leverage unlabeled

data to both make known classes more compact, and better separate them from unknown classes.

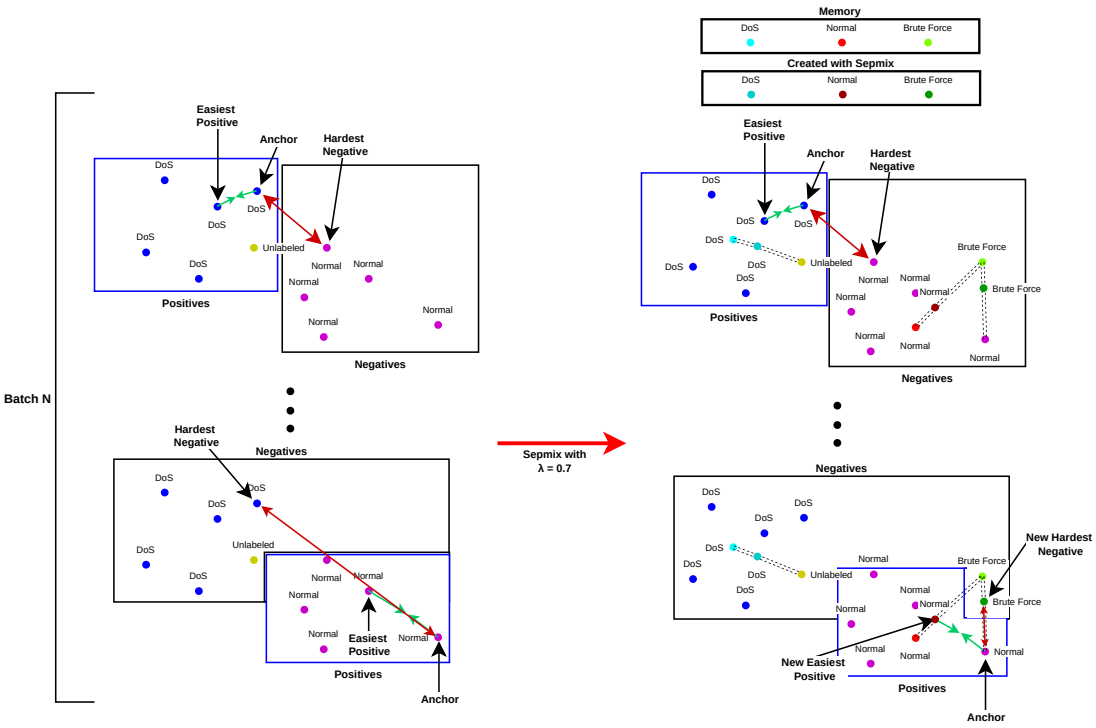


Figure 5.3 – Illustration of how Sepmix creates new instances using a memory.

The new supervised contrastive loss then uses the new created samples to represent positives and negatives. This new loss is defined similarly to Equation (5.1) and is shown in Equation (5.5), where mI represents the combination of batch data, data added from the memory and new data created through Sepmix, and $mP(i) \equiv \{p \in mI : y_p = y_i\}$ represents the set of positives. z_p and z_a can thus include data coming from memory or created through Sepmix.

$$\mathcal{L}_{nsupcon} = \sum_{i \in I} \frac{-1}{|mP(i)|} \sum_{p \in mP(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in mI} exp(z_i \cdot z_a / \tau)} \tag{5.5}$$

Finally, adding VNE to the loss shown in Equation (5.5), the complete loss formulation is shown in Equation (5.6), with α influencing the impact of $\mathcal{L}_{VNE}$. If $\alpha > 0$, this forces the method to increase the rank of Z (this increases independence of the different z_i). Practically, this reduces the number of neurons in CE that will be equal to 0, which leads

to representations of higher quality.

$$\mathcal{L}_{SECL} = \mathcal{L}_{nsupcon} - \alpha \mathcal{L}_{VNE} \tag{5.6}$$

5.2.2 Connecting the different building blocks

An algorithm describing a single training step of a batch (a group) of samples is shown in Algorithm 5, with all three hyperparameters' (λ, α_{mem} and α_{VNE}) values being found using grid search.

Algorithm 5: A training step (for a single batch)

Data: X, y the data and labels from a batch,
$\mathcal{F}$ the Contrastive Encoder,
g the classification head,
λ the hyperparameter for Sepmix,
α_{mem} the hyperparameter for the memory bank,
α_{VNE} the hyperparameter for VNE

```
/* Compute the representation matrix                                  */
H ← F(X)      // Dropout is integrated in the NN and used here
/* If memory is full, it is updated following the method described
   in Section 5.2                                                     */
memory ← save to memory bank(H, α_mem)
/* New representations are added using the memory and batch
   representations and using the λ hyperparameter, as described in
   Equation (5.4)                                                     */
new_H, new_y ← get memory samples(H, memory, λ)
/* Get positives and negatives used for loss computation              */
p, n ←get positives and negatives(H, y, new_H, new_y)
/* The loss is computed using the projection of representations,
   positives and negatives of each sample, as in Equation (5.5)       */
loss ← L_supcon(g(H), g(p), g(n))
/* VNE is computed as in Equation (5.2) added to the loss, as in
   Equation (5.6)                                                     */
L_SECL ← loss − α L_VNE(g(H))
/* Update F and g                                                     */
backward(L_SECL)
```

5.3 Experimental results

Datasets used and pre-processing steps were the same as in Section 4.2.2 (Page 78).

Unknown classes setup In order to simulate new unknown classes and still retain the ability to easily evaluate the performance of the tested approaches, the easiest method is to simulate the presence of unknown classes by removing them from labeled train sets. For the scenario 1, the unknown classes were completely removed from the train sets. For the scenario 2, unknown classes data was kept but labels were removed by assigning the label -1. During training, all samples with a label of -1 can only be mixed with other samples that possess a correct label by using Sepmix, as was shown in Figure 5.3.

Evaluation methodology To evaluate the performance of the proposed approach in both scenarios, results were averaged on two runs with different initializations of the three datasets, i.e., different classes were randomly selected to be removed. In order to keep a sufficient number of classes for training, only up to a third of the attack classes were removed: 5 for WADI and CIC-IDS2017, and 3 for UNSW-NB15. This is a big difference compared to the evaluation of OSL approaches that limit themselves to leaving out a single class from the dataset, which makes it unclear if their approach would be able to detect multiple ZDAs, even without distinguishing them.

In both scenarios, the proposed approach was compared to a supervised baseline. This is a NN using a cross-entropy loss with the same architecture as CE in the proposed approach (without the classification head) *trained knowing all classes (even those unknown for SECL). The comparison to this supervised baseline gives an upper bound to the performance that the approach has to get close to.*

The different regularization approaches are model-agnostic, and the contrastive approach can be applied to any kind of NN. As such, the approach developed in this thesis can be used with any kind of NN, which represents a big family of models. While the NN architecture used could be improved, a similar increase in performance could be expected with other NN architectures by using the methodology presented here.

Metrics used to evaluate performance are the same as what is generally used in multi-class supervised learning. Accuracy will be used to give a general idea of the performance. F1-score is also chosen because it relays information about two metrics important in intrusion detection: the Detection Rate (or Recall) and the False Alarm Rate (the opposite of Precision). Therefore, the higher the F1-score, the higher the Detection Rate and the

lower the False Alarm Rate. Results given for F1-scores in Tables will be micro-averaged, i.e., averaged while taking into account class proportions to better compare both the supervised baseline and SECL to other state-of-the-art methods, while F1-scores in Figures will be macro-averaged to properly show performance even on underrepresented classes.

Hyperparameter tuning Multiple hyperparameters are used in SECL and need to be adjusted for optimal performance. In this case, there are six hyperparameters that have the biggest impact on performance and need to be carefully selected:

— The contrastive loss temperature, τ (Equation (5.5)). Varying this hyperparameter will have an impact on the loss, artificially increasing or decreasing it, thus also impacting computed gradients in the same way. The lower the temperature, the higher the loss, and the more representations will be pushed together in the case of positives and pushed apart in the case of negatives. Experiments showed that a temperature around 0.1 was optimal.

— The architecture of the CE. It needs to be complex enough to learn rich representations, while not being too complex that training becomes too unstable. The problem of instability is further amplified by regularization mechanisms and thus the architecture has to be chosen carefully. Experiments showed that a five-layered network (512, 1024, 2048, 4096, 2048), with dropout layers and ReLU non-linearities between each layer produced rich enough representations with enough stability to train successfully.

— The λ used for Sepmix (Equation (5.4)). The selected λ influences how close the created sample is to the original sample: the higher the λ, the closer it is. Experiments showed that λ lower than 0.5 tends to impede learning because created samples are too far from same class samples and the training process becomes too complex. Values ranging between 0.6 and 0.9 have a similar impact, depending on the chosen α for VNE.

— The α used for VNE. Both positive and negatives values are possible, but only positive values force the CE to learn richer representations. Values ranging between 0.1 and 0.3 tend to have a similar impact, depending on the chosen λ.

— Both k for k-Means and k-Nearest Neighbors. Experiments showed that fixing k-Nearest Neighbors' k to a small number, e.g., 5 allowed to reduce the impact of normal traffic's high prevalence. For k-Means, the k value can be obtained by finding the k giving the highest silhouette coefficient.

When using a lower value for Sepmix λ, e.g., 0.3, this will increase the effect of regularization and VNE's α is best chosen smaller, e.g., 0.1.

The hyperparameter values used for experiments shown in Section 5.3.1 are: 0.1 for temperature, (512, 1024, 2048, 4096, 2048) for the CE architecture, 0.75 for λ, 0.3 for α, 5 for the k of k-Nearest Neighbors and the k for k-Means has been obtained via computing the silhouette coefficient, a measure of how well clusters are constructed. In practice, this number ends up being higher than the sum of the number of known and unknown classes, mainly because of the diversity present in the normal traffic.

5.3.1 Results

First, in order to properly evaluate the performance of the proposed approaches, the supervised baseline *trained knowing all classes* need to be evaluated. Additionally, a Dummy baseline that only predicts the most prevalent class, which is normal traffic in the three datasets, has been added to show how the prevalence of normal traffic impacts results, especially when they are micro-averaged. Any method should at least be able to improve over this Dummy baseline. Results obtained are shown in Table 5.1.

Table 5.1: Accuracy of baselines on all datasets

Dataset	**Baseline**	
	Dummy	Supervised
WADI	0.9895	0.9994
UNSW-NB15	0.8735	0.9882
CIC-IDS2017	0.8030	0.9955

Values were rounded to the fourth decimal

It can be seen from the Dummy baseline that normal traffic is highly prevalent, especially for the WADI dataset. Although detrimental to performance and evaluation, this high imbalance is a fundamental characteristic of intrusion detection problems and should be expected from IDS datasets to properly represent realistic traffic. Because the high prevalence of normal traffic could bias results, we chose to remove this class from the micro-averaged results in the following sections, which explains the difference in results of the supervised baseline in the following results compared to Table 5.1.

Scenario 1

As a reminder, the scenario 1 is the scenario in which new attacks are completely unknown, and thus are not in the training dataset. Therefore, results in this Section show the ability of SECL to detect multiple ZDAs after an initial training using a supervised dataset.

From Table 5.2, it is visible that considering both known classes and unknown attacks, SECL is consistently close to the supervised baseline, and sometimes even better for WADI.

Table 5.2: F1-score of SECL and the supervised baseline for scenario 1 on all datasets, depending on the number of unknown classes

Dataset	Model	Number of unknown classes				
		1	2	3	4	5
WADI	Baseline	0.9974	0.9974	0.9974	0.9974	0.9974
	SECL	0.9985	0.9987	0.9971	0.9950	0.9941
UNSW-NB15	Baseline	0.9874	0.9874	0.9874	X	X
	SECL	0.9840	0.9824	0.9784	X	X
CIC-IDS2017	Baseline	0.9950	0.9950	0.9950	0.9950	0.9950
	SECL	0.9943	0.9749	0.9869	0.9800	0.9825

Values were rounded to the fourth decimal. Experiments stopped at three classes for UNSW-NB15.

In order to get a better picture of the actual performance and ability to generalize of SECL, F1-score of SECL and the supervised baseline on the attacks known and unknown to SECL (which excludes performance on normal traffic) are shown respectively in Figure 5.4 and Figure 5.5.

It can be seen in Figure 5.4 that results shown in Table 5.2 are confirmed, and SECL is able to achieve performance similar to a supervised approach on known attacks. However, it is important to ensure that the ability to detect unknown classes is not impaired by this high performance on known classes.

From Figure 5.5a and Figure 5.5c, it is shown that SECL is able to detect part of the new attacks consistently, although not at the level of a fully supervised model. In Figure 5.5b, SECL almost completely missed unknown attacks when 1 attack was removed.

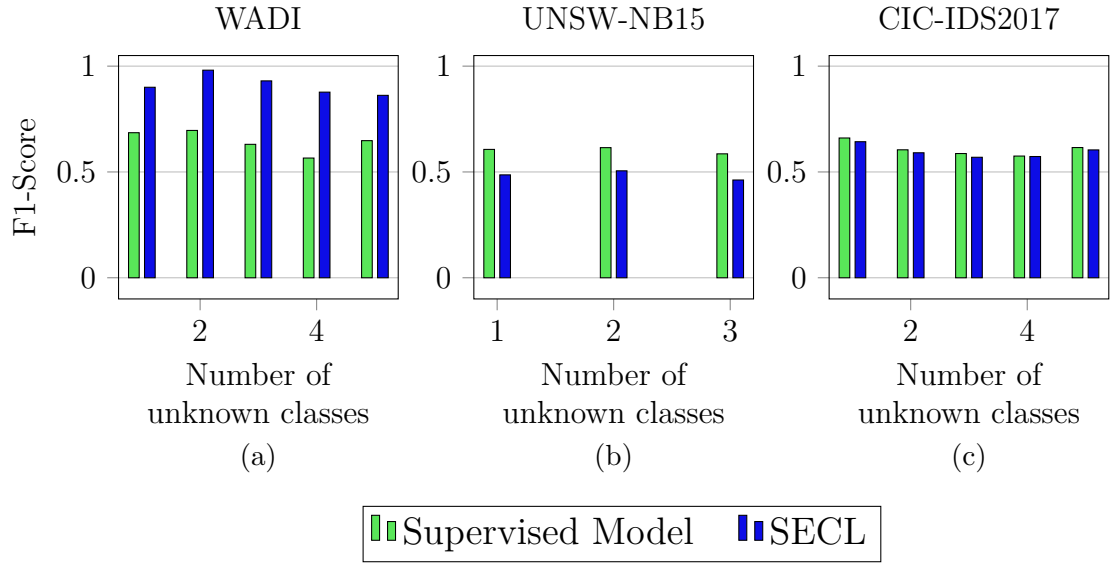


Figure 5.4 – F1-score of SECL and the supervised baseline for known classes in scenario 1.

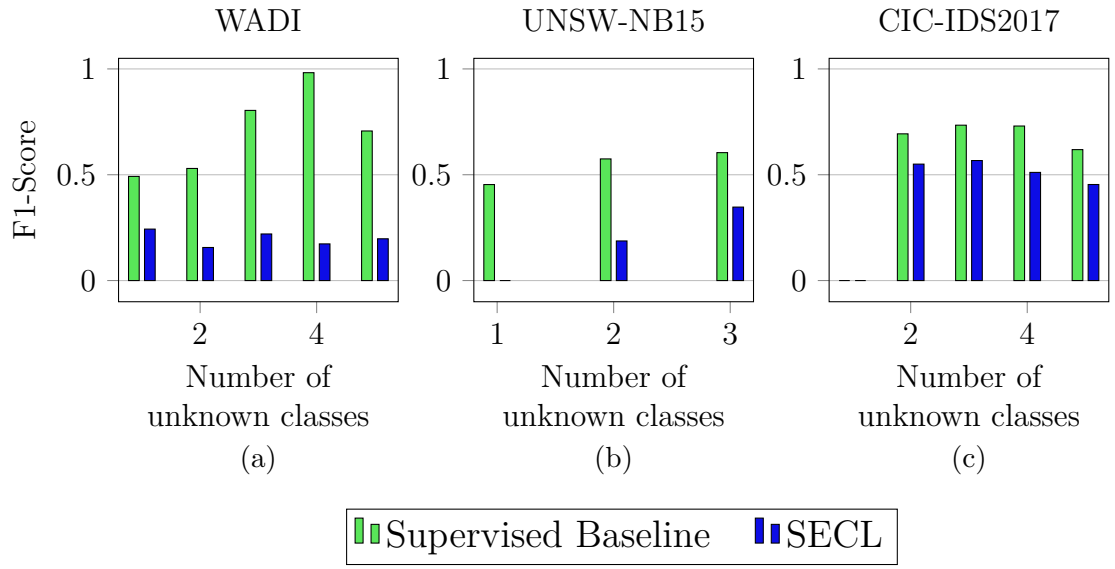


Figure 5.5 – F1-score of SECL and the supervised baseline for unknown classes in scenario 1. From [50].

This might partly be a bias induced by the fact that particularly hard to detect classes were removed, as exemplified by the lower F1-score of the supervised baseline. In Figure 5.5c, both SECL and the supervised baseline missed the unknown attacks, showing that SECL also struggles to detect unknown attacks if they are hard to learn for supervised methods.

While SECL shows a consistent ability to detect ZDAs, it frequently struggles to detect classes that are also hard to detect in a supervised setting except in cases where the supervised baseline also struggles in detecting attacks it learned. Furthermore, it can even be better at detecting known attacks than a supervised model that is trained knowing all classes. Although most important to detect ZDAs, regularization can also impact and improve detection of known classes, especially when these classes have outliers.

Scenario 2

As a reminder, the scenario 2 is the one in which new attacks are present in the training dataset, but are unlabeled. This scenario shows how an IDS initially trained with a supervised dataset would be able to incrementally learn using unlabeled data, collected while in operation, to better detect both known classes and ZDAs. While this is harder to learn on these attacks than by relying on labels, it allows to leverage their data through Sepmix to potentially learn to differentiate them without the need to identify them. If a sample from a known class and a sample from a ZDA are used by Sepmix to create a virtual sample, all other than this known class will try to move away from the virtual sample, thus improving detection of ZDAs without actually needing to identify them.

Table 5.3 shows that SECL is consistently better than the supervised baseline for WADI, and quite close for the other two datasets. This means that being able to leverage unknown attacks during training, even without labels, might allow to learn representations of much higher quality and increases the ability of the approach to generalize and detect ZDAs.

Similarly to scenario 1, performance of SECL on known attacks (Figure 5.6) is very similar to the supervised model, although this time, performance is slightly lower on WADI and slightly higher for UNSW-NB15 and CIC-IDS2017.

For unknown attacks however, the situation is different from scenario 1, as shown in Figure 5.7.

From Figure 5.7a, it is shown that SECL is almost always better at detecting unknown attacks that the supervised baseline that was trained knowing the labels. It means that SECL was able to create rich representations that are able to facilitate the task of separat-

Table 5.3: F1-score of SECL and the supervised baseline for scenario 2 on all datasets, depending on the number of unknown classes

Dataset	Model	Number of unknown classes				
		1	2	3	4	5
WADI	Baseline	0.9974	0.9974	0.9974	0.9974	0.9974
	SECL	0.9994	0.9992	0.9992	0.9989	0.9984
UNSW-NB15	Baseline	0.9874	0.9874	0.9874	X	X
	SECL	0.9852	0.9838	0.9845	X	X
CIC-IDS2017	Baseline	0.9950	0.9950	0.9950	0.9950	0.9950
	SECL	0.9947	0.9944	0.9905	0.9930	0.9927

Values were rounded to the fourth decimal. Experiments stopped at three classes for UNSW-NB15.

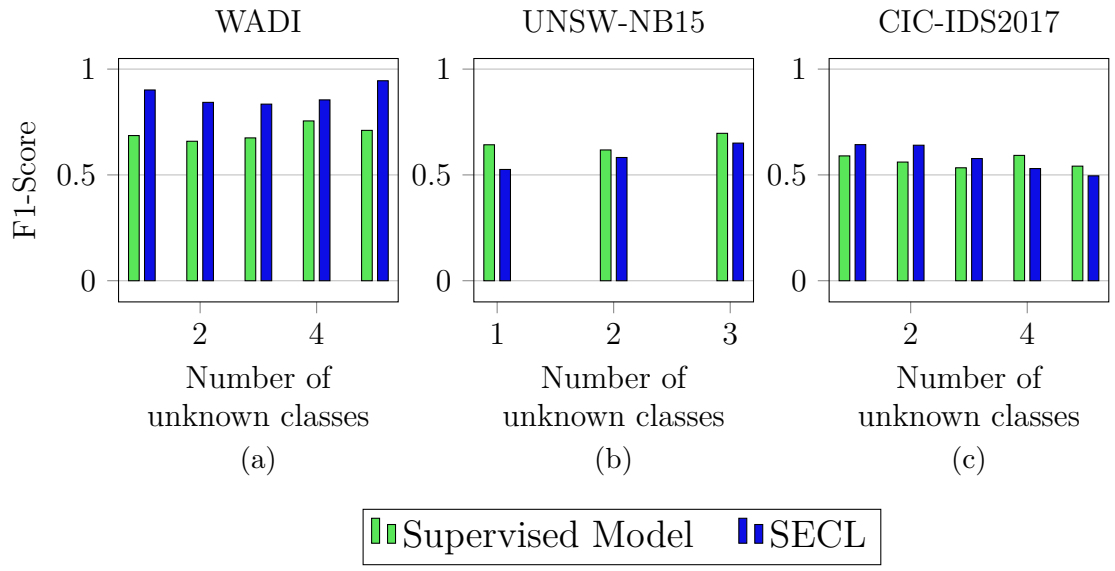


Figure 5.6 – F1-score of SECL and the supervised baseline for known classes in scenario 2.

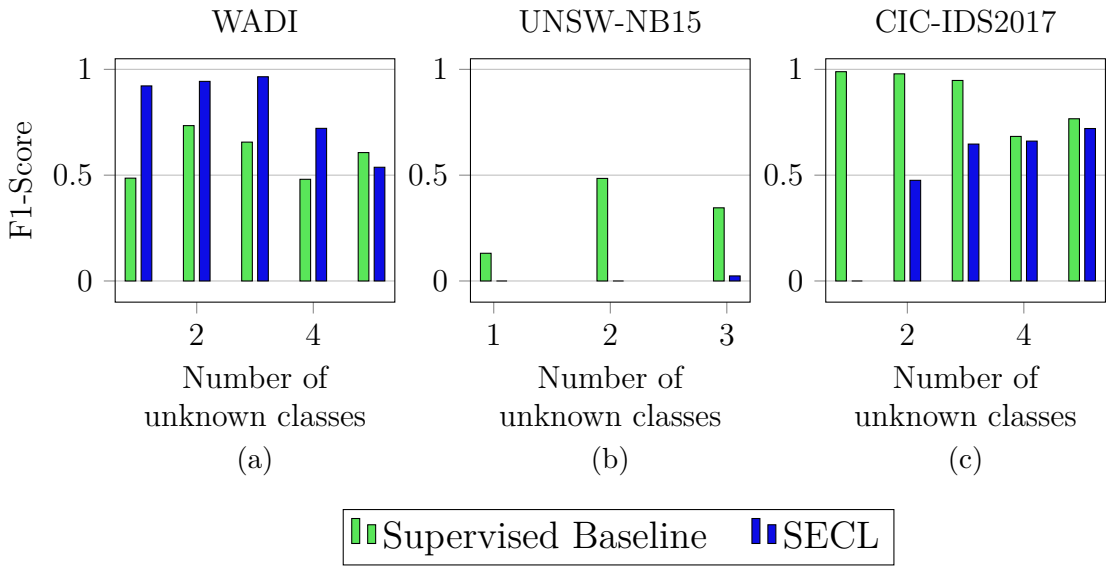


Figure 5.7 – F1-score of SECL and the supervised baseline for unknown classes in scenario 2. From [50].

ing both known and unknown classes. In Figure 5.7c, SECL also seems quite effective at detecting unknown attacks, although results are below or close to the supervised baseline. On the contrary, in Figure 5.7b, SECL seems unable to detect unknown attacks. It might be caused by a high similarity between known and unknown attacks, hence causing the approach to incorrectly leverage unlabeled data during training.

A conjecture about why SECL is able to better detect both known and unknown attacks on WADI than a supervised method is that SECL overfits less and is thus more robust to outliers. Because the WADI dataset resembles time series, traffic at the beginning of an attack is much closer to normal traffic than when the attack has completely impacted the system. As such, if a supervised method is unable to learn from the beginning of the attack, it will generally miss it because of overfitting. SECL, however, will still be able to distinguish the beginning of attacks because they are simulated through Sepmix which can mix normal traffic and attack traffic. Dropout and VNE further help in reducing overfitting.

Finally, results obtained by SECL on all classes show that the performance seems even more stable as the number of unlabeled attacks increases than it was in scenario 1. This is a very promising result, because this means that SECL might be able to leverage a very high number of unlabeled classes to further enrich the learned representations and better detect both known classes and ZDAs.

5.3.2 Ablation study of the regularization method

To determine the influence of each component of the new regularization method combining dropout, VNE and Sepmix, experiments were run removing either one of the three components.

Table 5.4: F1-score of SECL and time overhead, removing different components of the regularization method. On WADI with three unlabeled classes.

Ablation	**Attacks**		**Average training time/epoch**
	Unknown	All	
Bare NN	–	–	49
No VNE	0.7877	0.7793	216
No Sepmix	0.8968	0.7867	184
No dropout	0.9452	0.8227	212
SECL	0.9906	0.9083	227

Values were rounded to the fourth decimal.
Time reported is in seconds.
Bare NN refers to the supervised baseline, without contrastive learning or regularizations.

Table 5.4 shows results on detection of ZDAs on WADI with three classes unlabeled in the training set (scenario 2). It can clearly be seen that removing any component of the combined regularization method greatly decreases performance, be it for detection of ZDAs or for all attacks.

All three components seem to have a significant impact on the F1-score for detecting both unknown attacks and all classes. Of the three components, dropout seems to have the lowest impact. This is expected because it is known to help regularizing in-distribution samples. While it also seems to help in detecting ZDAs, its effect is more limited than both Sepmix and VNE. Although Sepmix and VNE seem to have a similar impact on all classes, Sepmix seems to have less impact on detecting ZDAs than VNE. An hypothesis is that increasing the quality of the learned representations have more effect than actually making classes more compact and further apart. While Sepmix allows to both make known classes more compact and separate them from ZDAs, this does not actually increase the quality of the representations. On the opposite, VNE that explicity forces SECL to learn

richer representations has a much higher impact on detecting ZDAs.

It can be seen from the average training time per epoch that the proposed approach does induce a significant computation overhead. However, interestingly, almost all the overhead is caused by the CL approach. The three different regularizations used in this approach have an almost negligible computation overhead. The most expensive regularization among the three is Sepmix, which was expected since it has to create virtual instances, a computationally expensive process.

5.4 Conclusion

Contrary to state-of-the-art IDSs based on OSL that are able to only detect a single unknown class, we developed SECL, a method designed to detect and classify ZDAs using CL and a new regularization method that combines dropout, Sepmix, and VNE. All three components are differently but conjointly helping SECL to better detect ZDAs while retaining a performance on known classes similar to supervised methods.

Two scenarios were considered: unknown attacks are completely absent during training, and unknown attacks are present in the training data, but were never identified, and thus are unlabeled. Results on scenario 1 show that the proposed approach is quite consistently able to detect unknown attacks, even as their number increases. For scenario 2, the proposed approach is able to leverage unlabeled attacks during training and is consistently close to, if not even better, in performance, than a fully supervised method trained knowing all classes.

Furthermore, results obtained by SECL on scenario 2 suggest that SECL might be able to incrementally learn using a high number of unlabeled classes to further increase performance in detecting both known attacks and ZDAs. Therefore, SECL provides a first step towards building next-generation IDSs, able to detect both known and ZDAs by relying on the ever-increasing volume of traffic to incrementally train.

Chapter 6

Conclusion

This thesis focuses on the subject of ML applied to the detection of cyberattacks. Our objective was to investigate the different building blocks relative to how an IDS using ML should work to help decision-making of cybersecurity operators, which obstacles are existing and could be overcome as well as identify future avenues of research. Although autonomous systems using ML are emerging in various fields, we believe ML should not be used in critical applications or situations without a human making the final decisions, which also influenced the research performed during this thesis.

We thus endeavored to develop methods to increase trust in ML-based IDSs. We first developed new metrics based on the Common Vulnerability Scoring System (CVSS) score to adapt ML metrics to the cybersecurity landscape in order to better inform cybersecurity experts of the actual ability of an IDS. These new metrics also allowed to better reflect an IDS's performance according to the system is has to protect. Finally, we further integrated cybersecurity knowledge by training IDSs using CVSS scores, to help an IDS in focusing more on attacks with higher severity.

We then researched the domain of Explainable AI (XAI) in order to explain how IDSs reach decisions, in order to assist human experts in their investigations. While we found that the field is not yet mature when it comes to intrusion detection, we developed a method leveraging XAI in order to verify and correct IDS predictions. Furthermore, both the method developed can be both manual, by showing potential errors for human experts to investigate, or fully automated.

Finally, we focused on improving abilities of ML-based IDSs that complement current signature-based IDSs. We thus developed an approach based on contrastive learning and different regularization methods that is able to detect and differentiate ZDAs.

While we made steps towards building trustable IDSs, there are still various factors fundamental to the intrusion detection problem that can be improved upon.

6.1 Datasets and metrics: fundamental building blocks

The problem of building ML-based IDSs is highly dependent on the quality of datasets and metrics. As has been illustrated in Chapter 2 and Chapter 3, both the quality of datasets and the metrics used totally define how useful a given approach would be.

6.1.1 Datasets

While new datasets are created, the cybersecurity landscape is evolving much faster. Therefore, a better solution than publicly available datasets would be publicly available testbeds, or simulations for faster iterations of new datasets, allowing to include recent attack methodologies much faster [89]. Moreover, it is important to keep in mind the fact that existing datasets can be faulty (Section 2.6.2), and although some corrected versions might exist, e.g., for CIC-IDS2017 ([106, 97]), this also impairs comparison with existing state-of-the-art that used faulty datasets.

Finally, best performance, especially considering the intrusion detection problem as a multi-class problem, is often achieved by supervised ML methods which also leads to create datasets from simulated environments, because they are much easier to label. However, as has been shown in Chapter 5, more recent methods categorized as semi-supervised learning methods are able to leverage unlabeled data to improve the training process. Therefore, leveraging unsupervised data from real traffic to enrich existing datasets could definitely be a great avenue of research.

6.1.2 Metrics

Metrics used to evaluate the performance of ML models also need to be properly reflected upon. IDS problems are heavily impacted by the inherent imbalance in the data which causes results to often be overly optimistic. A good solution would be to focus on full results, such as the full unnormalized confusion matrix, or to rely on macro-averaged results that properly reflect performance on rarer classes, especially for multi-class problems. Furthermore, existing ML metrics are generic and do not particularly take into account specificities of the cybersecurity domain. In Chapter 3, we proposed new metrics integrating CVSS scores to better represent the ability of ML-based IDSs to properly defend different systems. However, while we believe this to be a step in the right direction, this does not provide a perfect solution. Future metrics could lean more

towards the actual cost of a cyberattack, taking into account personnel time, material and financial costs.

Finally, it might be needed to redefine how the problem is formulated. As was raised in [145], data, be it represented in packets or flows, separate a single behavior in multiple parts. It thus raises the question of the necessity to detect all packets or flows belonging to an attack when an early detection might be enough to prevent an attack from impacting a system. Therefore, it might be more interesting to detect anomalous data as soon as possible rather than finding all anomalous data.

6.2 The need for explainability: what can we expect?

In many cases, being able to explain how any ML model behaves is relevant for legal reasons or questions about responsibility. Even though this mainly applies to autonomous Artificial Intelligence (AI) systems, this might still be needed when ML is used to help in decision-making. However, as has been illustrated in Chapter 4, XAI is still a relatively recent field of research and should be further developed before being fully usable.

Many XAI methods have been developed for image applications and can explain how ML models behave because parts of images themselves possess semantic meanings. On the other hand, for IDSs, while existing methods could provide insights about the way an IDS behaves, this is still far from comparable to a human's way of thinking and there is still a need to bridge this gap. The situation could be improved by curating datasets to use features that are meaningful to human experts or by providing textual explanations similarly to current generative AI models (e.g., ChatGPT). However, we still need to ensure that textual explanations truly reflect an IDS's behavior and are not simply some convincing hallucination.

In the meantime, as has been shown in Chapter 4, XAI can still provide value, for example by improving the performance of IDSs via an additional decision process. It could also increase the trust in an IDS's decision by ensuring that the decision is conform to the IDS's general behavior.

6.3 Maintainability of ML-based IDSs

While signature-based IDSs require significant work from human experts to maintain and create rules for new threats, ML-based IDSs similarly require significant work in the

form of dataset curation and model training.

As signature-based IDSs evolve, additional rules become more and more complex to ensure that different rules do not overlap. Similarly, ML-based IDSs can suffer from concept drift, where some known classes slowly shift for a reason or another. For example, adding some new services could potentially lead the IDS to flag anything from these services as anomalies, completely overloading any alert system. Therefore, it is necessary to frequently update the IDS, although training could take a long time. Research focused on semi-supervised learning, as well as incremental learning, offers potential solutions, but also requires advances to solve their own problems, such as incremental learning's catastrophic forgetting.

However, contrary to signature-based IDSs, ML-based IDSs offer the possibility to detect zero-day attacks with relatively high performance, as was shown in Chapter 5, which could potentially lengthen the time they can operate without retraining, as well as potentially facilitate the task of updating training datasets.

Finally, while most ML-based IDSs focus on either detecting anomalies or performing multi-class classification with a limited number of attacks, it is still unclear how such IDSs would fare trying to recognize hundreds or thousands of attacks.

6.4 Outlook and remaining obstacles

While ML-based IDSs have improved over the years, this is mostly linked to improvements in ML learning methods and did not develop much taking into account the problem's specificities. More effort should be put into properly delineating the topic of ML-based IDS, to adapt ML and its components to the problem of intrusion detection instead of the opposite.

Despite that, the field has matured and research is increasingly paying attention to critical subjects such as the quality of datasets, how to properly evaluate ML-based IDSs, or which network features can bias an IDS and should be handled differently.

Since datasets are so important to the quality of IDSs, different approaches were developed to enrich the training process without necessarily needing to aggregate huge datasets. Federated Learning allows to share results of training between multiple entities, without compromising confidentiality of the data. It could, used concurrently with Incremental Learning approaches, allow for much faster update of IDSs to different threats, including threats not personally encountered.

More recently, advances in Graph Neural Networks (GNNs) brought new possibilities with graph being a natural representation of network topologies. In this case, GNNs could also bring changes about how the problem is formulated and solved. For example, it could be more interesting to find malicious nodes or edges of the graph, rather than detecting occurring cyberattacks. However, it also brings about its own challenges with dynamic graphs being more memory intensive, although much more representative of real networks.

Additionally, as is the case with most ML applications, there is a question about the resistance of such methods to adversarial attacks, although some research tried to provide solutions to this problem [85, 104]. Our own research about XAI showed that in some cases, possessing the correct information, it is very easy for an attacker to bypass IDSs by crafting attack packets, with for example, some specific flags. There are also questions about how much information could be gleaned from an IDS, whether it be an IDS's parameters, or even the data used to train it, by having the possibility to interact with it.

To conclude, we tried, in this thesis, to develop approaches that are as general as possible, in order to be used with any ML approach. While training leveraging CVSS scores (Section 3.4) and using a contrastive loss (Chapter 5) are specific to NNs, other methods presented in this manuscript (Section 3.1, Section 3.3, Chapter 4) are usable with any ML approach. Finally, all approaches target different capabilities or characteristics of an IDS and can be combined to create a more trustable IDS.

BIBLIOGRAPHY

[1] *1999 DARPA Intrusion Detection Evaluation Dataset*, [Online]. Available:https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset, 1999, (visited on 03/24/2022).

[2] Mahmoud Abdallah et al., « A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs », *in*: *Proceedings of the 16th International Conference on Availability, Reliability and Security*, Aug. 2021, DOI: 10.1145/3465481.3469190, URL: http://dx.doi.org/10.1145/3465481.3469190.

[3] Amina Adadi and Mohammed Berrada, « Peeking Inside the Black-Box: a Survey on Explainable Artificial Intelligence (XAI) », *in*: *IEEE Access* 6 (2018), pp. 52138–52160, DOI: 10.1109/access.2018.2870052, URL: https://doi.org/10.1109/access.2018.2870052.

[4] Julius Adebayo et al., « Debugging Tests for Model Explanations », *in*: *CoRR* (2020), arXiv: 2011.05429 [cs.CV], URL: http://arxiv.org/abs/2011.05429v1.

[5] Iftikhar Ahmad et al., « Enhancing Svm Performance in Intrusion Detection Using Optimal Feature Subset Selection Based on Genetic Principal Components », *in*: *Neural Computing and Applications* 24.7-8 (2013), pp. 1671–1682, DOI: 10.1007/s00521-013-1370-6, URL: http://dx.doi.org/10.1007/s00521-013-1370-6.

[6] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur, « WADI », *in*: *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2017, DOI: 10.1145/3055366.3055375, URL: http://dx.doi.org/10.1145/3055366.3055375.

[7] Tanzeela Altaf et al., « Ne-Gconv: a Lightweight Node Edge Graph Convolutional Network for Intrusion Detection », *in*: *Computers & Security* 130 (2023), p. 103285, DOI: 10.1016/j.cose.2023.103285, URL: http://dx.doi.org/10.1016/j.cose.2023.103285.

[8] Ons Aouedi et al., « Federated Semisupervised Learning for Attack Detection in Industrial Internet of Things », *in*: *IEEE Transactions on Industrial Informatics* 19.*1* (2023), pp. 286–295, DOI: 10.1109/tii.2022.3156642, URL: http://dx.doi.org/10.1109/TII.2022.3156642.

[9] Giovanni Apruzzese, Pavel Laskov, and Johannes Schneider, « Sok: Pragmatic Assessment of Machine Learning for Network Intrusion Detection », *in*: *CoRR* (2023), arXiv: 2305.00550 [cs.CR], URL: http://arxiv.org/abs/2305.00550v1.

[10] Paulo Freitas de Araujo-Filho et al., « Intrusion Detection for Cyber-Physical Systems Using Generative Adversarial Networks in Fog Environment », *in*: *IEEE Internet of Things Journal* 8.*8* (2021), pp. 6247–6256, DOI: 10.1109/jiot.2020.3024800, URL: http://dx.doi.org/10.1109/JIOT.2020.3024800.

[11] Alejandro Barredo Arrieta et al., « Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible Ai », *in*: *CoRR* (2019), arXiv: 1910.10045v2 [cs.AI], URL: http://arxiv.org/abs/1910.10045v2.

[12] Yosef Ashibani and Qusay H. Mahmoud, « Cyber Physical Systems Security: Analysis, Challenges and Solutions », *in*: *Computers and Security* 68 (2017), pp. 81–97, DOI: 10.1016/j.cose.2017.04.005, URL: https://doi.org/10.1016/j.cose.2017.04.005.

[13] Julien Aussibal and Laurent Gallon, « A New Distributed IDS Based on CVSS Framework », *in*: *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, Nov. 2008, pp. 701–707, DOI: 10.1109/sitis.2008.115, URL: http://dx.doi.org/10.1109/SITIS.2008.115.

[14] Stefan Axelsson, « The Base-Rate Fallacy and the Difficulty of Intrusion Detection », *in*: *ACM Transactions on Information and System Security* 3.*3* (2000), pp. 186–205, DOI: 10.1145/357830.357849, URL: http://dx.doi.org/10.1145/357830.357849.

[15] Sikha Bagui and Kunqi Li, « Resampling Imbalanced Data for Network Intrusion Detection Datasets », *in*: *Journal of Big Data* 8.*1* (2021), p. 6, DOI: 10.1186/s40537-020-00390-x, URL: http://dx.doi.org/10.1186/s40537-020-00390-x.

[16] Millot Baptiste, Francq Julien, and Sicard Franck, « Systematic and Efficient Anomaly Detection Framework using Machine Learning on Public ICS Datasets », *in*: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, July 2021, pp. 292–297, DOI: 10.1109/csr51186.2021.9527911, URL: http://dx.doi.org/10.1109/CSR51186.2021.9527911.

[17] Pieter Barnard, Nicola Marchetti, and Luiz A. DaSilva, « Robust Network Intrusion Detection Through Explainable Artificial Intelligence (XAI) », *in*: *IEEE Networking Letters* 4.*3* (2022), pp. 167–171, DOI: 10.1109/lnet.2022.3186589, URL: http://dx.doi.org/10.1109/LNET.2022.3186589.

[18] Abhijit Bendale and Terrance Boult, « Towards Open World Recognition », *in*: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, DOI: 10.1109/cvpr.2015.7298799, URL: https://doi.org/10.1109/cvpr.2015.7298799.

[19] Y. Bengio, P. Simard, and P. Frasconi, « Learning Long-Term Dependencies With Gradient Descent Is Difficult », *in*: *IEEE Transactions on Neural Networks* 5.*2* (1994), pp. 157–166, DOI: 10.1109/72.279181, URL: http://dx.doi.org/10.1109/72.279181.

[20] Gustavo de Carvalho Bertoli et al., « Generalizing Intrusion Detection for Heterogeneous Networks: a Stacked-Unsupervised Federated Learning Approach », *in*: *CoRR* (2022), arXiv: 2209.00721 [cs.CR], URL: http://arxiv.org/abs/2209.00721v3.

[21] Umang Bhatt, Adrian Weller, and José M. F. Moura, « Evaluating and Aggregating Feature-Based Model Explanations », *in*: *CoRR* (2020), arXiv: 2005.00631 [cs.LG], URL: http://arxiv.org/abs/2005.00631v1.

[22] Monowar H. Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal Kumar Kalita, « Towards Generating Real-Life Datasets for Network Intrusion Detection », *in*: *International Journal of Network Security* 17 (2015), pp. 683–701.

[23] Holman Bolivar et al., « Multi-criteria Decision Making Model for Vulnerabilities Assessment in Cloud Computing regarding Common Vulnerability Scoring System », *in*: *2019 Congreso Internacional de Innovación y Tendencias en Ingenieria (CONIITI)*, 2019, pp. 1–6.

[24] Hamid Bostani and Mansour Sheikhan, « Hybrid of Binary Gravitational Search Algorithm and Mutual Information for Feature Selection in Intrusion Detection Systems », *in*: *Soft Computing* 21.*9* (2015), pp. 2307–2324, DOI: 10.1007/s00500-015-1942-8, URL: http://dx.doi.org/10.1007/s00500-015-1942-8.

[25] Iñigo López-Riobóo Botana et al., « Explanation Method for Anomaly Detection on Mixed Numerical and Categorical Spaces », *in*: *CoRR* (2022), arXiv: 2209.04173 [cs.LG], URL: http://arxiv.org/abs/2209.04173v1.

[26] Antoine Boudermine, « A dynamic attack graphs based approach for impact assessment of vulnerabilities in complex computer systems », Theses, Institut Polytechnique de Paris, Dec. 2022, URL: https://theses.hal.science/tel-03947453.

[27] Imed Eddine Boukari et al., « StrucTemp-GNN: An Intrusion Detection Framework in IoT Networks Using Dynamic Heterogeneous Graph Neural Networks », *in*: *Mobile, Secure, and Programmable Networking*, Mobile, Secure, and Programmable Networking, Springer Nature Switzerland, 2024, pp. 17–39, DOI: 10.1007/978-3-031-52426-4_2, URL: http://dx.doi.org/10.1007/978-3-031-52426-4_2.

[28] T. E. Boult et al., « Learning and the Unknown: Surveying Steps Toward Open World Recognition », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), pp. 9801–9807, DOI: 10.1609/aaai.v33i01.33019801, URL: https://doi.org/10.1609/aaai.v33i01.33019801.

[29] Leo Breiman, « Random Forests », *in*: *Machine Learning* 45.*1* (2001), pp. 5–32, DOI: 10.1023/a:1010933404324, URL: http://dx.doi.org/10.1023/A:1010933404324.

[30] Leo Breiman et al., *Classification And Regression Trees*, [] Routledge, 2017, nil, DOI: 10.1201/9781315139470, URL: http://dx.doi.org/10.1201/9781315139470.

[31] Kaidi Cao, Maria Brbic, and Jure Leskovec, « Open-World Semi-Supervised Learning », *in*: *International Conference on Learning Representations*, 2022, URL: https://openreview.net/forum?id=O-r8LOR-CCA.

[32] Evan Caville et al., « Anomal-E: a Self-Supervised Network Intrusion Detection System Based on Graph Neural Networks », *in*: *CoRR* (2022), arXiv: 2207.06819 [cs.LG], URL: http://arxiv.org/abs/2207.06819v4.

[33] Fabien Charmet et al., « Explainable artificial intelligence for cybersecurity: a literature survey », *in*: *Annals of Telecommunications* 77.*11-12* (2022), pp. 789–812, DOI: 10.1007/s12243-022-00926-7, URL: http://dx.doi.org/10.1007/s12243-022-00926-7.

[34] Nitesh V. Chawla et al., « Smote: Synthetic Minority Over-Sampling Technique », *in*: *J. Artif. Int. Res.* 16.*1* (June 2002), pp. 321–357, ISSN: 1076-9757.

[35] Tianqi Chen and Carlos Guestrin, « Xgboost: a Scalable Tree Boosting System », *in*: *CoRR* (2016), arXiv: 1603.02754 [cs.LG], URL: http://arxiv.org/abs/1603.02754v3.

[36] Ting Chen et al., « A Simple Framework for Contrastive Learning of Visual Representations », *in*: *CoRR* (2020), arXiv: 2002.05709v3 [cs.LG], URL: http://arxiv.org/abs/2002.05709v3.

[37] Anton Cherepanov, *Win32/Industroyer, A new threat for industrial control systems*, 2017.

[38] Davide Chicco, « Ten Quick Tips for Machine Learning in Computational Biology », *in*: *BioData Mining* 10.*1* (2017), p. 35, DOI: 10.1186/s13040-017-0155-3, URL: http://dx.doi.org/10.1186/s13040-017-0155-3.

[39] Andrea Corsini and Shanchieh Jay Yang, « Are Existing Out-Of-Distribution Techniques Suitable for Network Intrusion Detection? », *in*: *2023 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2023, pp. 1–9, DOI: 10.1109/cns59707.2023.10288685, URL: http://dx.doi.org/10.1109/CNS59707.2023.10288685.

[40] Andrea Corsini, Shanchieh Jay Yang, and Giovanni Apruzzese, « On the Evaluation of Sequential Machine Learning for Network Intrusion Detection », *in*: *CoRR* (2021), arXiv: 2106.07961 [cs.CR], URL: http://arxiv.org/abs/2106.07961v1.

[41] Corinna Cortes and Vladimir Vapnik, « Support-Vector Networks », *in*: *Machine Learning* 20.*3* (1995), pp. 273–297, DOI: 10.1007/bf00994018, URL: http://dx.doi.org/10.1007/BF00994018.

[42] T. Cover and P. Hart, « Nearest Neighbor Pattern Classification », *in*: *IEEE Transactions on Information Theory* 13.*1* (1967), pp. 21–27, DOI: 10.1109/tit.1967.1053964, URL: http://dx.doi.org/10.1109/TIT.1967.1053964.

[43] Gideon Creech and Jiankun Hu, « Generation of a new IDS test dataset: Time to retire the KDD collection », *in*: *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 4487–4492, DOI: 10.1109/wcnc.2013.6555301, URL: http://dx.doi.org/10.1109/WCNC.2013.6555301.

[44] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano, « Triton: the First Ics Cyber Attack on Safety Instrument Systems », *in*: *Proc. Black Hat USA* 2018 (2018), pp. 1–26.

[45] Hongwei Ding et al., « Imbalanced Data Classification: a Knn and Generative Adversarial Networks-Based Hybrid Approach for Intrusion Detection », *in*: *Future Generation Computer Systems* 131 (2022), pp. 240–254, DOI: 10.1016/j.future.2022.01.026, URL: http://dx.doi.org/10.1016/j.future.2022.01.026.

[46] Hongwei Ding et al., « Tmg-Gan: Generative Adversarial Networks-Based Imbalanced Learning for Network Intrusion Detection », *in*: *IEEE Transactions on Information Forensics and Security* 19 (2024), pp. 1156–1167, DOI: 10.1109/tifs.2023.3331240, URL: http://dx.doi.org/10.1109/TIFS.2023.3331240.

[47] Finale Doshi-Velez and Been Kim, « Towards a Rigorous Science of Interpretable Machine Learning », *in*: *CoRR* (2017), arXiv: 1702.08608v2 [stat.ML], URL: http://arxiv.org/abs/1702.08608v2.

[48] Aeryn Dunmore et al., « A Comprehensive Survey of Generative Adversarial Networks (GANs) in Cybersecurity Intrusion Detection », *in*: *IEEE Access* 11 (2023), pp. 76071–76094, DOI: 10.1109/access.2023.3296707, URL: http://dx.doi.org/10.1109/ACCESS.2023.3296707.

[49] Robin Duraz et al., « Explainability-based Metrics to Help Cyber Operators Find and Correct Misclassified Cyberattacks », *in*: *Proceedings of the 2023 on Explainable and Safety Bounded, Fidelitous, Machine Learning for Networking*, Dec. 2023, DOI: 10.1145/3630050.3630177, URL: http://dx.doi.org/10.1145/3630050.3630177.

[50] Robin Duraz et al., « SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization », *in*: *Proceedings of the 19th International Conference on Availability, Reliability and Security*, July 2024, pp. 1–12, DOI: 10.1145/3664476.3664505, URL: http://dx.doi.org/10.1145/3664476.3664505.

[51] Robin Duraz et al., « Using Cvss Scores Can Make More Informed and More Adapted Intrusion Detection Systems », *in*: *JUCS - Journal of Universal Computer Science* 30.*9* (2024), pp. 1244–1264, DOI: 10.3897/jucs.131659, URL: http://dx.doi.org/10.3897/jucs.131659.

[52] Robin Emsley, « Chatgpt: These Are Not Hallucinations - They're Fabrications and Falsifications », *in*: *Schizophrenia* 9.*1* (2023), p. 52, DOI: 10.1038/s41537-023-00379-4, URL: http://dx.doi.org/10.1038/s41537-023-00379-4.

[53] Martin Ester et al., « A density-based algorithm for discovering clusters in large spatial databases with noise », *in*: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[54] Osama Faker and Erdogan Dogdu, « Intrusion Detection Using Big Data and Deep Learning Techniques », *in*: *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 86–93, DOI: 10.1145/3299815.3314439, URL: http://dx.doi.org/10.1145/3299815.3314439.

[55] Nicolas Falliere, Liam O Murch, and Eric Chien, *APT Case Study - W32.Stuxnet*, 2012.

[56] Mohamed Amine Ferrag et al., « Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study », *in*: *Journal of Information Security and Applications* 50.*102419* (2020), DOI: 10.1016/j.jisa.2019.102419, URL: http://dx.doi.org/10.1016/j.jisa.2019.102419.

[57] Mohamed Amine Ferrag et al., « Edge-Iiotset: a New Comprehensive Realistic Cyber Security Dataset of Iot and Iiot Applications for Centralized and Federated Learning », *in*: *IEEE Access* 10 (2022), pp. 40281–40306, DOI: 10.1109/access.2022.3165809, URL: http://dx.doi.org/10.1109/ACCESS.2022.3165809.

[58] Marcel Frigault et al., « Measuring the Overall Network Security by Combining CVSS Scores Based on Attack Graphs and Bayesian Networks », *in*: *Network Security Metrics*, Network Security Metrics, Springer International Publishing, 2017, pp. 1–23, DOI: 10.1007/978-3-319-66505-4_1, URL: http://dx.doi.org/10.1007/978-3-319-66505-4_1.

[59] Ni Gao, Yiyue He, and Beilei Ling, « Exploring Attack Graphs for Security Risk Assessment: a Probabilistic Approach », *in*: *Wuhan University Journal of Natural Sciences* 23.*2* (2018), pp. 171–177, DOI: 10.1007/s11859-018-1307-0, URL: http://dx.doi.org/10.1007/s11859-018-1307-0.

[60] Mossa Ghurab et al., « A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System », *in*: *Asian Journal of Research in Computer Science* 7.*4* (2021), pp. 14–33, DOI: 10.9734/ajrcos/2021/v7i430185, URL: http://dx.doi.org/10.9734/AJRCOS/2021/v7i430185.

[61] Zhongyuan Gong et al., « An Improved Hybrid Sampling Model for Network Intrusion Detection Based on Data Imbalance », *in*: *Artificial Intelligence Security and Privacy*, Artificial Intelligence Security and Privacy, Springer Nature Singapore, 2024, pp. 164–175, DOI: 10.1007/978-981-99-9788-6_14, URL: http://dx.doi.org/10.1007/978-981-99-9788-6_14.

[62] Ian J. Goodfellow et al., « Generative Adversarial Networks », *in*: *CoRR* (2014), arXiv: 1406.2661 [stat.ML], URL: http://arxiv.org/abs/1406.2661v1.

[63] Mikel Gorricho-Segura, Xabier Echeberria-Barrio, and Lander Segurola-Gil, « Edge-based Analysis for Network Intrusion Detection using a GNN Approach », *in*: *2023 JNIC Cybersecurity Conference (JNIC)*, June 2023, DOI: 10.23919/jnic58574.2023.10205384, URL: http://dx.doi.org/10.23919/JNIC58574.2023.10205384.

[64] Qiong Gu, Li Zhu, and Zhihua Cai, « Evaluation Measures of the Classification Performance of Imbalanced Data Sets », *in*: *Communications in Computer and Information Science*, Communications in Computer and Information Science, Springer Berlin Heidelberg, 2009, pp. 461–471, DOI: 10.1007/978-3-642-04962-0_53, URL: http://dx.doi.org/10.1007/978-3-642-04962-0_53.

[65] Asmaa Halbouni et al., « Cnn-Lstm: Hybrid Deep Neural Network for Network Intrusion Detection System », *in*: *IEEE Access* 10 (2022), pp. 99837–99849, DOI: 10.1109/access.2022.3206425, URL: http://dx.doi.org/10.1109/ACCESS.2022.3206425.

[66] Xueying Han et al., « Network Intrusion Detection Based on N-Gram Frequency and Time-Aware Transformer », *in*: *Computers & Security* 128 (2023), p. 103171, DOI: 10.1016/j.cose.2023.103171, URL: http://dx.doi.org/10.1016/j.cose.2023.103171.

[67] Kazuki Hara and Kohei Shiomoto, « Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder », *in*: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2020, pp. 1–8, DOI: 10.1109/noms47738.2020.9110343, URL: http://dx.doi.org/10.1109/NOMS47738.2020.9110343.

[68] Haibo He et al., « ADASYN: Adaptive synthetic sampling approach for imbalanced learning », *in*: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, DOI: 10.1109/ijcnn.2008.4633969, URL: http://dx.doi.org/10.1109/IJCNN.2008.4633969.

[69] Kaiming He et al., « Momentum Contrast for Unsupervised Visual Representation Learning », *in*: *CoRR* (2019), arXiv: 1911.05722v3 [cs.CV], URL: http://arxiv.org/abs/1911.05722v3.

[70] Mingshu He et al., « Deep-Feature-Based Autoencoder Network for Few-Shot Malicious Traffic Detection », *in*: *Security and Communication Networks* 2021 (2021), pp. 1–13, DOI: 10.1155/2021/6659022, URL: http://dx.doi.org/10.1155/2021/6659022.

[71] Anna Hedström et al., « Quantus: an Explainable Ai Toolkit for Responsible Evaluation of Neural Network Explanations and Beyond », *in*: *CoRR* (2022), arXiv: 2202.06861 [cs.LG], URL: http://arxiv.org/abs/2202.06861v3.

[72] Hanan Hindy et al., « A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems », *in*: *IEEE Access* 8 (2020), pp. 104650–104675, DOI: 10.1109/access.2020.3000179, URL: http://dx.doi.org/10.1109/ACCESS.2020.3000179.

[73] Sepp Hochreiter and Jürgen Schmidhuber, « Long Short-Term Memory », *in*: *Neural Computation* 9.8 (1997), pp. 1735–1780, DOI: 10.1162/neco.1997.9.8.1735, URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[74] Elike Hodo et al., « Shallow and Deep Networks Intrusion Detection System: a Taxonomy and Survey », *in*: *CoRR* (2017), arXiv: 1701.02145 [cs.CR], URL: http://arxiv.org/abs/1701.02145v1.

[75] Robert R. Hoffman et al., « Metrics for Explainable Ai: Challenges and Prospects », *in*: *CoRR* (2018), arXiv: 1812.04608 [cs.AI], URL: http://arxiv.org/abs/1812.04608v2.

[76] Hannes Holm, « Signature Based Intrusion Detection for Zero-Day Attacks: (Not) A Closed Chapter? », *in*: *2014 47th Hawaii International Conference on System Sciences*, 2014, DOI: 10.1109/hicss.2014.600, URL: http://dx.doi.org/10.1109/HICSS.2014.600.

[77] Zakaria Abou El Houda, Bouziane Brik, and Lyes Khoukhi, « "Why Should I Trust Your Ids?": an Explainable Deep Learning Framework for Intrusion Detection Systems in Internet of Things Networks », *in*: *IEEE Open Journal of the Communications Society* 3 (2022), pp. 1164–1176, DOI: 10.1109/ojcoms.2022.3188750, URL: http://dx.doi.org/10.1109/OJCOMS.2022.3188750.

[78] Shuokang Huang and Kai Lei, « Igan-Ids: an Imbalanced Generative Adversarial Network Towards Intrusion Detection System in Ad-Hoc Networks », *in*: *Ad Hoc Networks* 105 (2020), p. 102177, DOI: 10.1016/j.adhoc.2020.102177, URL: http://dx.doi.org/10.1016/j.adhoc.2020.102177.

[79] A. Hussain et al., « An NIDS for Known and Zero-Day Anomalies », *in*: *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*, Apr. 2023, pp. 1–7, DOI: 10.1109/drcn57075.2023.10108319, URL: http://dx.doi.org/10.1109/DRCN57075.2023.10108319.

[80] Félix Iglesias and Tanja Zseby, « Analysis of Network Traffic Features for Anomaly Detection », *in*: *Machine Learning* 101.*1-3* (2014), pp. 59–84, DOI: 10.1007/s10994-014-5473-9, URL: http://dx.doi.org/10.1007/s10994-014-5473-9.

[81] Auwal Sani Iliyasu and Huifang Deng, « N-Gan: a Novel Anomaly-Based Network Intrusion Detection With Generative Adversarial Networks », *in*: *International Journal of Information Technology* 14.*7* (2022), pp. 3365–3375, DOI: 10.1007/s41870-022-00910-3, URL: http://dx.doi.org/10.1007/s41870-022-00910-3.

[82] Ashish Jaiswal et al., « A Survey on Contrastive Self-Supervised Learning », *in*: *CoRR* (2020), arXiv: 2011.00362 [cs.CV], URL: http://arxiv.org/abs/2011.00362v3.

[83] Laszlo A. Jeni, Jeffrey F. Cohn, and Fernando De La Torre, « Facing Imbalanced Data–Recommendations for the Use of Performance Metrics », *in*: *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, 2013, pp. 245–251, DOI: 10.1109/acii.2013.47, URL: http://dx.doi.org/10.1109/ACII.2013.47.

[84] Kaiyuan Jiang et al., « Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network », *in*: *IEEE Access* 8 (2020), pp. 32464–32476, DOI: 10.1109/access.2020.2973730, URL: http://dx.doi.org/10.1109/ACCESS.2020.2973730.

[85] Houda Jmila and Mohamed Ibn Khedher, « Adversarial Machine Learning for Network Intrusion Detection: a Comparative Study », *in*: *Computer Networks* 214 (2022), p. 109073, DOI: 10.1016/j.comnet.2022.109073, URL: http://dx.doi.org/10.1016/j.comnet.2022.109073.

[86] Jonathan Goh and Sridhar Adepu and Khurum Nazir Junejo and Aditya Mathur, « A Dataset to Support Research in the Design of Secure Water Treatment Systems », *in*: *Critical Information Infrastructures Security*, Critical Information Infrastructures Security, Springer International Publishing, 2017, pp. 88–99, DOI: 10.1007/978-3-319-71368-7_8, URL: http://dx.doi.org/10.1007/978-3-319-71368-7_8.

[87] Yannis Kalantidis et al., « Hard Negative Mixing for Contrastive Learning », *in*: *CoRR* (2020), arXiv: 2010.01028 [cs.CV], URL: http://arxiv.org/abs/2010.01028v2.

[88] *KDD Cup 99 Data*, [Online]. Available:http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[89] A. Kenyon, L. Deka, and D. Elizondo, « Are Public Intrusion Datasets Fit for Purpose Characterising the State of the Art in Intrusion Event Datasets », *in*: *Computers & Security* 99 (2020), p. 102022, DOI: 10.1016/j.cose.2020.102022, URL: http://dx.doi.org/10.1016/j.cose.2020.102022.

[90] Prannay Khosla et al., « Supervised Contrastive Learning », *in*: *CoRR* (2020), arXiv: 2004.11362 [cs.LG], URL: http://arxiv.org/abs/2004.11362v5.

[91] Jaeill Kim et al., « VNE: An Effective Method for Improving Deep Representation by Manipulating Eigenvalue Distribution », *in*: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 3799–3810, DOI: 10.1109/cvpr52729.2023.00370, URL: http://dx.doi.org/10.1109/CVPR52729.2023.00370.

[92] Satoru Koda and Ikuya Morikawa, « OOD-Robust Boosting Tree for Intrusion Detection Systems », *in*: *2023 International Joint Conference on Neural Networks (IJCNN)*, June 2023, pp. 01–10, DOI: 10.1109/ijcnn54540.2023.10191603, URL: http://dx.doi.org/10.1109/IJCNN54540.2023.10191603.

[93] Igor Kononenko, « Semi-naive bayesian classifier », *in*: *Lecture Notes in Computer Science*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1991, pp. 206–219, DOI: 10.1007/bfb0017015, URL: http://dx.doi.org/10.1007/BFb0017015.

[94] Nickolaos Koroniotis et al., « Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-Iot Dataset », *in*: *Future Generation Computer Systems* 100 (2019), pp. 779–796, DOI: 10.1016/j.future.2019.05.041, URL: http://dx.doi.org/10.1016/j.future.2019.05.041.

[95] Gulshan Kumar, Krishan Kumar, and Monika Sachdeva, « The Use of Artificial Intelligence Based Techniques for Intrusion Detection: a Review », *in*: *Artificial Intelligence Review* 34.*4* (2010), pp. 369–387, DOI: 10.1007/s10462-010-9179-5, URL: http://dx.doi.org/10.1007/s10462-010-9179-5.

[96] Dominik Kus et al., « A False Sense of Security? », *in*: *Proceedings of the 8th ACM on Cyber-Physical System Security Workshop*, 2022, DOI: 10.1145/3494107.3522773, URL: http://dx.doi.org/10.1145/3494107.3522773.

[97] Maxime Lanvin et al., « Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes », *in*: *Lecture Notes in Computer Science*, Lecture Notes in Computer Science, Springer Nature Switzerland, 2023, pp. 18–33, DOI: 10.1007/978-3-031-31108-6_2, URL: http://dx.doi.org/10.1007/978-3-031-31108-6_2.

[98] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, « Deep Learning », *in*: *Nature* 521.*7553* (2015), pp. 436–444, DOI: 10.1038/nature14539, URL: http://dx.doi.org/10.1038/nature14539.

[99] Ding LI et al., *A Trustworthy View on XAI Method Evaluation*, 2022, DOI: 10.36227/techrxiv.21067438.v1, URL: http://dx.doi.org/10.36227/techrxiv.21067438.v1.

[100] Xiang Li et al., « Network Intrusion Detection With Edge-Directed Graph Multi-Head Attention Networks », *in*: *CoRR* (2023), arXiv: 2310.17348 [cs.CR], URL: http://arxiv.org/abs/2310.17348v1.

[101] Chin-Yew Lin and Eduard Hovy, « Automatic evaluation of summaries using N-gram co-occurrence statistics », *in*: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, 2003, nil, DOI: 10.3115/1073445.1073465, URL: http://dx.doi.org/10.3115/1073445.1073465.

[102] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai, « Cann: an Intrusion Detection System Based on Combining Cluster Centers and Nearest Neighbors », *in*: *Knowledge-Based Systems* 78 (2015), pp. 13–21, DOI: 10.1016/j.knosys.2015.01.009, URL: http://dx.doi.org/10.1016/j.knosys.2015.01.009.

[103] Zhong Qiu Lin et al., « Do Explanations Reflect Decisions? a Machine-Centric Strategy To Quantify the Performance of Explainability Algorithms », *in*: *CoRR* (2019), arXiv: 1910.07387 [cs.LG], URL: http://arxiv.org/abs/1910.07387v2.

[104] Zilong Lin, Yong Shi, and Zhi Xue, « IDSGAN: Generative Adversarial Networks for Attack Generation Against Intrusion Detection », *in*: *Advances in Knowledge Discovery and Data Mining*, vol. 13282, Advances in Knowledge Discovery and Data Mining, Springer International Publishing, 2022, chap. IDSGAN: Generative Adversarial Networks for Attack Generation Against Intrusion Detection, pp. 79–91, DOI: 10.1007/978-3-031-05981-0_7, URL: http://dx.doi.org/10.1007/978-3-031-05981-0_7.

[105] Zachary C. Lipton, « The Mythos of Model Interpretability », *in*: *CoRR* (2016), arXiv: 1606.03490v3 [cs.LG], URL: http://arxiv.org/abs/1606.03490v3.

[106] Lisa Liu et al., « Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018 », *in*: *2022 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2022, nil, DOI: 10.1109/cns56114.2022.9947235, URL: http://dx.doi.org/10.1109/CNS56114.2022.9947235.

[107] Xiaodong Liu et al., « A Gan and Feature Selection-Based Oversampling Technique for Intrusion Detection », *in*: *Security and Communication Networks* 2021 (2021), pp. 1–15, DOI: 10.1155/2021/9947059, URL: http://dx.doi.org/10.1155/2021/9947059.

[108] Yi Liu and Lanjian Wu, « Intrusion Detection Model Based on Improved Transformer », *in*: *Applied Sciences* 13.*10* (2023), p. 6251, DOI: 10.3390/app13106251, URL: http://dx.doi.org/10.3390/app13106251.

[109] Yue Liu et al., « Explainable Ai for Android Malware Detection: Towards Understanding Why the Models Perform So Well? », *in*: *CoRR* (2022), arXiv: 2209.00812 [cs.CR], URL: http://arxiv.org/abs/2209.00812v1.

[110] Wai Weng Lo et al., « E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT », *in*: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2022, DOI: 10.1109/noms54207.2022.9789878, URL: http://dx.doi.org/10.1109/NOMS54207.2022.9789878.

[111] Manuel Lopez-Martin et al., « Supervised Contrastive Learning Over Prototype-Label Embeddings for Network Intrusion Detection », *in*: *Information Fusion* 79 (2022), pp. 200–228, DOI: 10.1016/j.inffus.2021.09.014, URL: http://dx.doi.org/10.1016/j.inffus.2021.09.014.

[112] Scott Lundberg and Su-In Lee, « A Unified Approach To Interpreting Model Predictions », *in*: *CoRR* (2017), arXiv: 1705.07874 [cs.AI], URL: http://arxiv.org/abs/1705.07874v2.

[113] Laurens van der Maaten and Geoffrey Hinton, « Visualizing Data Using T-SNE », *in*: *Journal of Machine Learning Research* 9.*86* (2008), pp. 2579–2605, URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[114] Mayra Macas, Chunming Wu, and Walter Fuertes, « A Survey on Deep Learning for Cybersecurity: Progress, Challenges, and Opportunities », *in*: *Computer Networks* 212 (2022), p. 109032, DOI: 10.1016/j.comnet.2022.109032, URL: http://dx.doi.org/10.1016/j.comnet.2022.109032.

[115] J. MacQueen, « Some methods for classification and analysis of multivariate observations », *in*: 1967, URL: https://api.semanticscholar.org/CorpusID:6278891.

[116] Shraddha Mane and Dattaraj Rao, « Explaining Network Intrusion Detection System Using Explainable Ai Framework », *in*: *CoRR* (2021), arXiv: 2103.07110 [cs.CR], URL: http://arxiv.org/abs/2103.07110v1.

[117] Warren S. McCulloch and Walter Pitts, « A Logical Calculus of the Ideas Immanent in Nervous Activity », *in*: *The Bulletin of Mathematical Biophysics* 5.*4* (1943), pp. 115–133, DOI: 10.1007/bf02478259, URL: http://dx.doi.org/10.1007/BF02478259.

[118] John McHugh, « Testing Intrusion Detection Systems », *in*: *ACM Transactions on Information and System Security* 3.*4* (2000), pp. 262–294, DOI: 10.1145/382912.382923, URL: http://dx.doi.org/10.1145/382912.382923.

[119] Jorge Meira et al., « Performance Evaluation of Unsupervised Techniques in Cyber-Attack Anomaly Detection », *in*: *Journal of Ambient Intelligence and Humanized Computing* 11.*11* (2019), pp. 4477–4489, DOI: 10.1007/s12652-019-01417-9, URL: https://doi.org/10.1007/s12652-019-01417-9.

[120] Nicolas Michel et al., « Contrastive Learning for Online Semi-Supervised General Continual Learning », *in*: *CoRR* (2022), arXiv: 2207.05615 [cs.LG], URL: http://arxiv.org/abs/2207.05615v1.

[121] Tim Miller, « Explanation in Artificial Intelligence: Insights From the Social Sciences », *in*: *CoRR* (2017), arXiv: 1706.07269v3 [cs.AI], URL: http://arxiv.org/abs/1706.07269v3.

[122] Nour Moustafa, « A New Distributed Architecture for Evaluating Ai-Based Security Systems At the Edge: Network Ton_iot Datasets », *in*: *Sustainable Cities and Society* 72 (2021), p. 102994, DOI: 10.1016/j.scs.2021.102994, URL: http://dx.doi.org/10.1016/j.scs.2021.102994.

[123] Nour Moustafa and Jill Slay, « UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) », *in*: *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6, DOI: 10.1109/milcis.2015.7348942, URL: http://dx.doi.org/10.1109/milcis.2015.7348942.

[124] Rawshan Ara Mowri, Madhuri Siddula, and Kaushik Roy, « Interpretable Machine Learning for Detection and Classification of Ransomware Families Based on Api

Calls », *in*: *CoRR* (2022), arXiv: 2210.11235 [cs.CR], URL: http://arxiv.org/abs/2210.11235v2.

[125] Sowmya Myneni et al., « DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats », *in*: *Deployable Machine Learning for Security Defense*, Deployable Machine Learning for Security Defense, Springer International Publishing, 2020, chap. DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats, pp. 138–163, DOI: 10.1007/978-3-030-59621-7_8, URL: http://dx.doi.org/10.1007/978-3-030-59621-7_8.

[126] Meike Nauta et al., « From Anecdotal Evidence To Quantitative Evaluation Methods: a Systematic Review on Evaluating Explainable Ai », *in*: *CoRR* (2022), arXiv: 2201.08164 [cs.AI], URL: http://arxiv.org/abs/2201.08164v2.

[127] Subash Neupane et al., « Explainable Intrusion Detection Systems (X-IDS): a Survey of Current Methods, Challenges, and Opportunities », *in*: *CoRR* (2022), arXiv: 2207.06236 [cs.CR], URL: http://arxiv.org/abs/2207.06236v1.

[128] Hoang Nguyen and Rasha Kashef, « Ts-Ids: Traffic-Aware Self-Supervised Learning for Iot Network Intrusion Detection », *in*: *Knowledge-Based Systems* 279 (2023), p. 110966, DOI: 10.1016/j.knosys.2023.110966, URL: http://dx.doi.org/10.1016/j.knosys.2023.110966.

[129] Laisen Nie et al., « Intrusion Detection for Secure Social Internet of Things Based on Collaborative Edge Computing: a Generative Adversarial Network-Based Approach », *in*: *IEEE Transactions on Computational Social Systems* 9.*1* (2022), pp. 134–145, DOI: 10.1109/tcss.2021.3063538, URL: http://dx.doi.org/10.1109/TCSS.2021.3063538.

[130] Genki Osada, Kazumasa Omote, and Takashi Nishide, « Network Intrusion Detection Based on Semi-supervised Variational Auto-Encoder », *in*: *Computer Security - ESORICS 2017*, Computer Security - ESORICS 2017, Springer International Publishing, 2017, pp. 344–361, DOI: 10.1007/978-3-319-66399-9_19, URL: http://dx.doi.org/10.1007/978-3-319-66399-9_19.

[131] Kishore Papineni et al., « BLEU », *in*: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001, nil, DOI: 10.3115/1073083.1073135, URL: http://dx.doi.org/10.3115/1073083.1073135.

[132] Cheolhee Park et al., « An Enhanced Ai-Based Network Intrusion Detection System Using Generative Adversarial Networks », *in*: *IEEE Internet of Things Journal* 10.*3* (2023), pp. 2330–2345, DOI: 10.1109/jiot.2022.3211346, URL: http://dx.doi.org/10.1109/JIOT.2022.3211346.

[133] Gregory Plumb, Marco Tulio Ribeiro, and Ameet Talwalkar, « Finding and Fixing Spurious Patterns With Explanations », *in*: *CoRR* (2021), arXiv: 2106.02112 [cs.LG], URL: http://arxiv.org/abs/2106.02112v1.

[134] David Pujol-Perich et al., « Unveiling the Potential of Graph Neural Networks for Robust Intrusion Detection », *in*: *ACM SIGMETRICS Performance Evaluation Review* 49.*4* (2022), pp. 111–117, DOI: 10.1145/3543146.3543171, URL: http://dx.doi.org/10.1145/3543146.3543171.

[135] Lianyong Qi et al., « Fast Anomaly Identification Based on Multiaspect Data Streams for Intelligent Intrusion Detection Toward Secure Industry 4.0 », *in*: *IEEE Transactions on Industrial Informatics* 18.*9* (2022), pp. 6503–6511, DOI: 10.1109/tii.2021.3139363, URL: http://dx.doi.org/10.1109/TII.2021.3139363.

[136] Attiq Ur-Rehman et al., « Vulnerability Modelling for Hybrid Industrial Control System Networks », *in*: *Journal of Grid Computing* 18.*4* (2020), pp. 863–878, DOI: 10.1007/s10723-020-09528-w, URL: http://dx.doi.org/10.1007/s10723-020-09528-w.

[137] Shahbaz Rezaei and Xin Liu, « Deep Learning for Encrypted Traffic Classification: an Overview », *in*: *IEEE Communications Magazine* 57.*5* (2019), pp. 76–81, DOI: 10.1109/mcom.2019.1800819, URL: http://dx.doi.org/10.1109/MCOM.2019.1800819.

[138] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, « "Why Should I Trust You?": Explaining the Predictions of Any Classifier », *in*: *CoRR* (2016), arXiv: 1602.04938 [cs.LG], URL: http://arxiv.org/abs/1602.04938v3.

[139] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, « Anchors: High-Precision Model-Agnostic Explanations », *in*: *AAAI*, vol. 32, 1, 2018.

[140] Markus Ring et al., « A Survey of Network-Based Intrusion Detection Data Sets », *in*: *Computers & Security* 86 (2019), pp. 147–167, DOI: 10.1016/j.cose.2019.06.005, URL: http://dx.doi.org/10.1016/j.cose.2019.06.005.

[141] Martin Roesch and Stanford Telecommunications, « Snort - Lightweight Intrusion Detection for Networks », *in*: *LISA '99: Proceedings of the 13th USENIX conference on System administration*, 1999, pp. 229–238.

[142] Lukas Ruff et al., « A Unifying Review of Deep and Shallow Anomaly Detection », *in*: *Proceedings of the IEEE* 109.*5* (2021), pp. 756–795, DOI: 10.1109/jproc.2021.3052449, URL: https://doi.org/10.1109/jproc.2021.3052449.

[143] Mohanad Sarhan et al., « From Zero-Shot Machine Learning To Zero-Day Attack Detection », *in*: *International Journal of Information Security* 22.*4* (2023), pp. 947–959, DOI: 10.1007/s10207-023-00676-0, URL: http://dx.doi.org/10.1007/s10207-023-00676-0.

[144] W. J. Scheirer et al., « Toward Open Set Recognition », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.*7* (2013), pp. 1757–1772, DOI: 10.1109/tpami.2012.256, URL: https://doi.org/10.1109/tpami.2012.256.

[145] Max Schrötter, Andreas Niemann, and Bettina Schnor, « A Comparison of Neural-Network-Based Intrusion Detection Against Signature-Based Detection in Iot Networks », *in*: *Information* 15.*3* (2024), p. 164, DOI: 10.3390/info15030164, URL: http://dx.doi.org/10.3390/info15030164.

[146] Gesina Schwalbe and Bettina Finzel, « A Comprehensive Taxonomy for Explainable Artificial Intelligence: a Systematic Survey of Surveys on Methods and Concepts », *in*: *Data Mining and Knowledge Discovery* (2023), DOI: 10.1007/s10618-022-00867-8, URL: http://dx.doi.org/10.1007/s10618-022-00867-8.

[147] Ramprasaath R. Selvaraju et al., « Grad-Cam: Visual Explanations From Deep Networks Via Gradient-Based Localization », *in*: *International Journal of Computer Vision* 128.*2* (2019), pp. 336–359, DOI: 10.1007/s11263-019-01228-7, URL: https://doi.org/10.1007/s11263-019-01228-7.

[148] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, « Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization », *in*: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116, DOI: 10.5220/0006639801080116, URL: https://doi.org/10.5220/0006639801080116.

[149] Iman Sharafaldin et al., « Towards a Reliable Intrusion Detection Benchmark Dataset », *in*: *Software Networking* 2017.*1* (2017), pp. 177–200, DOI: 10.13052/jsn2445-9739.2017.009, URL: http://dx.doi.org/10.13052/jsn2445-9739.2017.009.

[150] Ali Shiravi et al., « Toward Developing a Systematic Approach To Generate Benchmark Datasets for Intrusion Detection », *in*: *Computers & Security* 31.*3* (2012), pp. 357–374, ISSN: 0167-4048, DOI: 10.1016/j.cose.2011.12.012, URL: https://www.sciencedirect.com/science/article/pii/S0167404811001672.

[151] Kashif Siddiqui and Thomas E. Doyle, « Trust Metrics for Medical Deep Learning Using Explainable-AI Ensemble for Time Series Classification », *in*: *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, Sept. 2022, pp. 370–377, DOI: 10.1109/ccece49351.2022.9918458, URL: http://dx.doi.org/10.1109/CCECE49351.2022.9918458.

[152] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, « Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps », *in*: *CoRR* (2013), arXiv: 1312.6034v2 [cs.CV], URL: http://arxiv.org/abs/1312.6034v2.

[153] Daniel Smilkov et al., « Smoothgrad: Removing Noise By Adding Noise », *in*: *CoRR* (2017), arXiv: 1706.03825v1 [cs.LG], URL: http://arxiv.org/abs/1706.03825v1.

[154] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz, « Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation », *in*: *Lecture Notes in Computer Science*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 1015–1021, DOI: 10.1007/11941439_114, URL: http://dx.doi.org/10.1007/11941439_114.

[155] Nitish Srivastava et al., « Dropout: a Simple Way To Prevent Neural Networks From Overfitting », *in*: *Journal of Machine Learning Research* 15.*56* (2014), pp. 1929–1958, URL: http://jmlr.org/papers/v15/srivastava14a.html.

[156] Sédrick Stassin et al., « An Experimental Investigation Into the Evaluation of Explainability Methods », *in*: *CoRR* (2023), arXiv: 2305.16361 [cs.LG], URL: http://arxiv.org/abs/2305.16361v1.

[157] Pengfei Sun et al., « Dl-Ids: Extracting Features Using Cnn-Lstm Hybrid Network for Intrusion Detection System », *in*: *Security and Communication Networks* 2020 (2020), pp. 1–11, DOI: 10.1155/2020/8890306, URL: http://dx.doi.org/10.1155/2020/8890306.

[158] Yiyou Sun and Yixuan Li, « Opencon: Open-World Contrastive Learning », *in*: *Transactions on Machine Learning Research* (2023), ISSN: 2835-8856, URL: https://openreview.net/forum?id=2wWJxtpFer.

[159] Radoslava Švihrová and Christian Lettner, « A semi-supervised approach for network intrusion detection », *in*: *Proceedings of the 15th International Conference on Availability, Reliability and Security*, Aug. 2020, DOI: 10.1145/3407023.3407073, URL: http://dx.doi.org/10.1145/3407023.3407073.

[160] Mateusz Szczepanski et al., « Achieving Explainability of Intrusion Detection System by Hybrid Oracle-Explainer Approach », *in*: *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8, DOI: 10.1109/ijcnn48605.2020.9207199, URL: http://dx.doi.org/10.1109/IJCNN48605.2020.9207199.

[161] Barbara G. Tabachnick and Linda S. Fidell, *Using Multivariate Statistics (5th Edition)*, USA: Allyn & Bacon, Inc., 2006, ISBN: 0205459382.

[162] Md. Alamin Talukder et al., « Machine Learning-Based Network Intrusion Detection for Big and Imbalanced Data Using Oversampling, Stacking Feature Embedding and Feature Extraction », *in*: *Journal of Big Data* 11.*1* (2024), p. 33, DOI: 10.1186/s40537-024-00886-w, URL: http://dx.doi.org/10.1186/s40537-024-00886-w.

[163] Shuting Tao et al., « Supervised Contrastive Learning With Tree-Structured Parzen Estimator Bayesian Optimization for Imbalanced Tabular Data », *in*: *CoRR* (2022), arXiv: 2210.10824 [cs.LG], URL: http://arxiv.org/abs/2210.10824v2.

[164] Mahbod Tavallaee et al., « A detailed analysis of the KDD CUP 99 data set », *in*: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, DOI: 10.1109/cisda.2009.5356528, URL: http://dx.doi.org/10.1109/cisda.2009.5356528.

[165] Abebe Tesfahun and D. Lalitha Bhaskari, « Intrusion Detection Using Random Forests Classifier with SMOTE and Feature Reduction », *in*: *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, Nov.

2013, DOI: 10.1109/cube.2013.31, URL: http://dx.doi.org/10.1109/cube.2013.31.

[166] Amjad M. Al Tobi and Ishbel Duncan, « Kdd 1999 Generation Faults: a Review and Analysis », *in*: *Journal of Cyber Security Technology* 2.*3-4* (2018), pp. 164–200, DOI: 10.1080/23742917.2018.1518061, URL: http://dx.doi.org/10.1080/23742917.2018.1518061.

[167] Farhan Ullah et al., « Ids-Int: Intrusion Detection System Using Transformer-Based Transfer Learning for Imbalanced Network Traffic », *in*: *Digital Communications and Networks* 10.*1* (2024), pp. 190–204, DOI: 10.1016/j.dcan.2023.03.008, URL: http://dx.doi.org/10.1016/j.dcan.2023.03.008.

[168] Oleksandra Vereschak, Gilles Bailly, and Baptiste Caramiaux, « How To Evaluate Trust in Ai-Assisted Decision Making? a Survey of Empirical Methodologies », *in*: *Proceedings of the ACM on Human-Computer Interaction* 5.*CSCW2* (2021), pp. 1–39, DOI: 10.1145/3476068, URL: http://dx.doi.org/10.1145/3476068.

[169] Oleksandra Vereschak et al., « Trust in AI-assisted Decision Making: Perspectives from Those Behind the System and Those for Whom the Decision is Made », *in*: *Proceedings of the CHI Conference on Human Factors in Computing Systems*, May 2024, nil, DOI: 10.1145/3613904.3642018, URL: http://dx.doi.org/10.1145/3613904.3642018.

[170] R. Vinayakumar et al., « Deep Learning Approach for Intelligent Intrusion Detection System », *in*: *IEEE Access* 7 (2019), pp. 41525–41550, DOI: 10.1109/access.2019.2895334, URL: http://dx.doi.org/10.1109/ACCESS.2019.2895334.

[171] Hoang V. Vo, Hanh P. Du, and Hoa N. Nguyen, « Apelid: Enhancing Real-Time Intrusion Detection With Augmented Wgan and Parallel Ensemble Learning », *in*: *Computers & Security* 136 (2024), p. 103567, DOI: 10.1016/j.cose.2023.103567, URL: http://dx.doi.org/10.1016/j.cose.2023.103567.

[172] Syed Wali and Irfan Khan, *Explainable AI and Random Forest Based Reliable Intrusion Detection system*, 2021, DOI: 10.36227/techrxiv.17169080.v1, URL: http://dx.doi.org/10.36227/techrxiv.17169080.v1.

[173] Maonan Wang et al., « An Explainable Machine Learning Framework for Intrusion Detection Systems », *in*: *IEEE Access* 8 (2020), pp. 73127–73141, DOI: 10.1109/access.2020.2988359, URL: https://doi.org/10.1109/access.2020.2988359.

[174] Wei Wang et al., « Robust Unsupervised Network Intrusion Detection With Self-Supervised Masked Context Reconstruction », *in*: *Computers & Security* 128 (2023), p. 103131, DOI: 10.1016/j.cose.2023.103131, URL: http://dx.doi.org/10.1016/j.cose.2023.103131.

[175] Chunyuan Wu et al., « Feature-oriented Design of Visual Analytics System for Interpretable Deep Learning based Intrusion Detection », *in*: *2020 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, 2020, pp. 73–80, DOI: 10.1109/tase49443.2020.00019, URL: http://dx.doi.org/10.1109/TASE49443.2020.00019.

[176] Yali Wu et al., « An Active Learning Framework Using Deep Q-Network for Zero-Day Attack Detection », *in*: *Computers & Security* 139 (2024), p. 103713, DOI: 10.1016/j.cose.2024.103713, URL: http://dx.doi.org/10.1016/j.cose.2024.103713.

[177] Hong Xuan, Abby Stylianou, and Robert Pless, « Improved Embeddings with Easy Positive Triplet Mining », *in*: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020, pp. 2463–2471, DOI: 10.1109/wacv45572.2020.9093432, URL: http://dx.doi.org/10.1109/WACV45572.2020.9093432.

[178] Yanqing Yang et al., « Improving the Classification Effectiveness of Intrusion Detection By Using Improved Conditional Variational Autoencoder and Deep Neural Network », *in*: *Sensors* 19.*11* (2019), p. 2528, DOI: 10.3390/s19112528, URL: http://dx.doi.org/10.3390/s19112528.

[179] Yanqing Yang et al., « Network Intrusion Detection Based on Supervised Adversarial Variational Auto-Encoder With Regularization », *in*: *IEEE Access* 8 (2020), pp. 42169–42184, DOI: 10.1109/access.2020.2977007, URL: http://dx.doi.org/10.1109/ACCESS.2020.2977007.

[180] Mahmood Yousefi-Azar et al., « Autoencoder-based feature learning for cyber security applications », *in*: *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2017, DOI: 10.1109/ijcnn.2017.7966342, URL: https://doi.org/10.1109/ijcnn.2017.7966342.

[181] Yawei Yue et al., « Contrastive Learning Enhanced Intrusion Detection », *in*: *IEEE Transactions on Network and Service Management* 19.*4* (2022), pp. 4232–4247, DOI: 10.1109/tnsm.2022.3218843, URL: http://dx.doi.org/10.1109/TNSM.2022.3218843.

[182] Tahmina Zebin, Shahadate Rezvy, and Yuan Luo, « An Explainable Ai-Based Intrusion Detection System for Dns Over Https (DoH) Attacks », *in*: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 2339–2349, DOI: 10.1109/tifs.2022.3183390, URL: http://dx.doi.org/10.1109/TIFS.2022.3183390.

[183] Hongpo Zhang et al., « An Effective Convolutional Neural Network Based on Smote and Gaussian Mixture Model for Intrusion Detection in Imbalanced Dataset », *in*: *Computer Networks* 177 (2020), p. 107315, DOI: 10.1016/j.comnet.2020.107315, URL: http://dx.doi.org/10.1016/j.comnet.2020.107315.

[184] Hongyi Zhang et al., « Mixup: Beyond Empirical Risk Minimization », *in*: *CoRR* (2017), arXiv: 1710.09412 [cs.LG], URL: http://arxiv.org/abs/1710.09412v2.

[185] Yichi Zhang et al., « Intrusion Detection of Industrial Internet-Of-Things Based on Reconstructed Graph Neural Networks », *in*: *IEEE Transactions on Network Science and Engineering* 10.*5* (2023), pp. 2894–2905, DOI: 10.1109/tnse.2022.3184975, URL: http://dx.doi.org/10.1109/TNSE.2022.3184975.

[186] Zhao Zhang et al., « A Scalable Network Intrusion Detection System Towards Detecting, Discovering, and Learning Unknown Attacks », *in*: *International Journal of Machine Learning and Cybernetics* 12.*6* (2021), pp. 1649–1665, DOI: 10.1007/s13042-020-01264-7, URL: http://dx.doi.org/10.1007/s13042-020-01264-7.

[187] Zhibo Zhang et al., « Explainable Artificial Intelligence Applications in Cyber Security: State-Of-The-Art in Research », *in*: *IEEE Access* 10 (2022), pp. 93104–93139, DOI: 10.1109/access.2022.3204051, URL: http://dx.doi.org/10.1109/ACCESS.2022.3204051.

[188] Meihui Zhong et al., « A Survey on Graph Neural Networks for Intrusion Detection Systems: Methods, Trends and Challenges », *in*: *Computers & Security* 141 (2024), p. 103821, DOI: 10.1016/j.cose.2024.103821, URL: http://dx.doi.org/10.1016/j.cose.2024.103821.

[189] Qiuming Zhu, « On the Performance of Matthews Correlation Coefficient (MCC) for Imbalanced Dataset », *in*: *Pattern Recognition Letters* 136 (2020), pp. 71–80, DOI: 10.1016/j.patrec.2020.03.030, URL: http://dx.doi.org/10.1016/j.patrec.2020.03.030.

[190] Tommaso Zoppi et al., « Which Algorithm Can Detect Unknown Attacks? Comparison of Supervised, Unsupervised and Meta-Learning Algorithms for Intrusion Detection », *in*: *Computers & Security* 127 (2023), p. 103107, DOI: 10.1016/j.cose.2023.103107, URL: http://dx.doi.org/10.1016/j.cose.2023.103107.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Apprentissage Automatique de confiance pour les Systèmes de Detection d'Intrusion

Mot clés : Apprentissage Automatique, Détection d'intrusion, Métriques, XAI, Nouvelles Classes

Résumé : Les systèmes de détection d'intrusion sont des composants essentiels à la défense de notre écosystème numérique. Récemment, les avancées en apprentissage automatique ont permis de développer de nouveaux types de système de détection d'intrusion, permettant de s'éloigner du besoin de créer des règles de détection de plus en plus complexes. Ces systèmes de détection utilisant l'apprentissage automatique sont capables d'apprendre de façon autonome différent comportements, à la condition d'avoir un jeu de données bien calibré. Le contexte de la cybersécurité amène des besoins spécifiques, des besoin qui sont différents des tâches d'apprentissage automatique les plus courantes : la reconnaissance d'images et le traitement du langage. Cela implique qu'il faut adapter les différents procédés utilisés en apprentissage automatique pour répondre à ces attentes. Étant dans un environnement ayant d'importants enjeux, les systèmes de détection d'intrusion devraient plutôt être utilisés pour aider à la prise de décisions, mais cela reste essentiel de pouvoir leur faire confiance. De ce fait, dans cette thèse, nous avons dans un premier temps développé une nouvelle métrique basée sur les scores CVSS, permettant d'intégrer des connaissances de cybersécurité dans le processus d'évaluation des systèmes de détection d'intrusion. Nous nous sommes ensuite concentrés sur le besoin de comprendre des décisions qui ne sont pas compréhensibles. Même si le domaine de l'explicabilité n'est pas assez avancé pour correctement expliquer ces décisions, cela reste possible de vérifier la confiance en celles-ci d'une façon plus robuste, amenant à examiner ou corriger d'éventuelles erreurs. Finalement, nous nous sommes efforcés de compléter les approches existantes en utilisant des techniques d'apprentissage automatique récentes pour augmenter la capacité à détecter de nouvelles cyberattaques. Toutes ces méthodes contribuent donc à créer des systèmes de détection d'intrusion utilisant l'apprentissage automatique qui sont plus fiables.

Title: Trustable Machine Learning for Intrusion Detection Systems

Keywords: Machine Learning, Intrusion Detection, Metrics, XAI, New classes

Abstract: Intrusion detection systems are essentiel components to defend our digital ecosystem. Recently, the advent of machine learning allowed to develop new types of intrusion detection systems, breaking away from the need to carefully craft more and more complex detection rules. These detection systems based on machine learning are able to autonomously learn to recognize different behaviors, given a sufficiently well designed dataset. The context of cybersecurity brings specific requirements to the task at hand, requirements that are different from machine learning's most developed tasks: image recognition and natural language processing. This implies adapting the different mechanisms employed in machine learning to cater to these requirements. Being used in a high stake environment, intrusion detection systems should be used to help in decision-making, yet it is still fundamental to be able to trust them. Therefore, in this thesis, we first developed a new metric based on CVSS scores, allowing to integrate cybersecurity knowledge into the evaluation process of intrusion detection systems. We then focused on how to increase confidence in otherwise incomprehensible decisions. While explainability has yet to be mature enough to properly explain decisions, it can still allow to check the confidence in the decision in a more robust way, leading to investigate or correct mistakes. Finally, we endeavored to complement current approaches, by increasing the ability to detect and differentiate new cyberattacks, leveraging novel machine learning techniques. All these methods thus contribute in making intrusion detection systems based on machine learning more trustable.