An explainable-by-design ensemble learning system to detect unknown network attacks

Céline Minh*†, Kevin Vermeulen†, Cédric Lefebvre*, Philippe Owezarski†, William Ritchie*
*Custocy, Toulouse, France. Email: {cminh, clefebvre, writchie}@custocy.com
†LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France. Email: {celine.minh, kevin.vermeulen, owe}@laas.fr

***Abstract*—Machine learning (ML) is a promising technology for network intrusion detection systems. There is a wide range of ML algorithms that are potential candidates for network intrusion detection systems, as they exhibit very good detection accuracy in average. However, significant detection differences appear when facing different kinds of attacks, some being prone to better detect some particular attack types. They then often appear to complement each other. The challenge then lies in determining the accurate result when several ML models provide different results, and this without any explanation about their decision. To address this challenge, our system aims to reconstruct attack patterns from the outputs of these ML models and presenting them in an interpretable manner. For that, we propose an approach combining ensemble learning and stacking with a meta-learner that works on graphical representation of traffic flows, that then provides the required explainability level for the decisions made. The evaluation of our system, using the CSE-CIC-IDS2018 dataset, demonstrates a significant improvement achieved through the combination of multiple ML algorithms. Furthermore, we emphasize the importance of explainability in network intrusion detection systems and the need for accurate and interpretable models. Our system goes beyond traditional detection methods by reporting anomalous feature pairs and providing visual representations of attack patterns, empowering analysts to better understand and respond to network threats.**

***Index Terms*—Network security, unsupervised anomaly detection, explainable AI, ensemble learning, CNN**

I. Introduction

Machine learning (ML) is a promising technology for network intrusion detection systems (NIDSs) as it enables the detection of attacks on large amounts of data. However, a limitation of ML models is their tendency to produce conflicting results: different models may classify the same network flow as either an attack or benign. This lack of consensus among models raises a challenge in identifying the accurate result, as ML models are often considered as black boxes and lack transparent explanations about their detection results. Efforts must be made to address this challenge and find a solution that can determine the accurate result when ML models face particular attacks in different traffic environments.

This opens the road to ensemble learning [1], [2], which is an approach that combines multiple ML models, but that can provide different classification results on traffic flows. The problem is then to identify the appropriate model that provides the right class for traffic flows. Ensemble learning is nevertheless a promising approach to achieve a more accurate attack detection system. Specifically, *stacking* is an ensemble learning method that combines multiple models, called base-learners, that perform the same task with a decision process that finally tries to select the right model(s) with possibly the right decision e.g., by weighted majority voting. To cope with the limits of such relatively simple algorithms as majority or minority voting, a more sophisticated stacking method starts to be considered for ML based IDS. This new approach takes advantage of a meta-learner: the outputs of the multiple base-learners serve as input to a higher-level model - the meta-learner - that, based on appropriate learning methods, can select among all the base-learners the one with the right classification results for traffic flows. Our contribution develops this approach.

We then propose a novel ensemble approach that systematically presents the results of each base-learner in a graphical way for better visualization as well as explainability purposes to security analysts. The graphical approach is aimed at facilitating informed decision-making. Following the principle explained before, the decision is made by the meta-learner through the training of a combination of concise visual representations of network anomalies.

Explainability is the property of a system that makes its reasoning and results understandable by humans [3], [4]. Security analysts play a crucial role in making decisions based on NIDS analysis, and providing them with intelligible evidence of the ML models' detections is essential for building trust in the system. Explainability not only enhances collaboration between analysts and artificial intelligence (AI) systems but also helps engineers and researchers understand the strengths and weaknesses of the models, enabling them to design more accurate systems.

We propose a method that simultaneously allows users to take advantage of ensemble learning from multiple base-learners to enhance detections and visualize the results of all the base-learners to gain insights into how these detections are made. We introduce a visual representation of unsupervised learning (UL) detections over time that intends to both help security analysts understand what is happening on the network and allow our meta-learner – a convolutional neural networks (CNN) – to identify attack patterns [5], [6]. Our system is expected to preserve UL properties, including the detection of unknown attacks, because the layer that is supervised, the meta-learner, does not train on raw network traffic but on graphical features historically provided by base-learners, i.e.,

978-3-903176-59-1 ©2023 IFIP

meta-data.

Overall, our approach combines the advantages of ensemble learning and explainability to enhance the detection performance of NIDSs and empower security analysts in making informed decisions. We introduce an explainable-by-design system that analyzes and combines a set of UL models to detect network attacks. Our contributions can be summarized as follows:

1) A more transparent ML-based NIDS that enables security analysts to understand and trust the system's detections.
2) Visual representations of network anomalies that allows security analysts to interpret and gain insights into the detected network anomalies.
3) An ensemble learning method that uses a CNN as a meta-learner to combine base-learners.

Main results include that on our evaluation dataset, our system has only three false positives (FPR 0.0093) and one false negative (TPR 0.9898) that can be mitigated by the explainability provided by our system. To ensure explainability, our system provides intermediate and visual representations of the data that can be easily understood by an analyst.

The rest of the paper is organized as follows:

- In Section II, we will examine related work on ensemble learning, attack detection, and explainable AI, and position our contribution in such a context.
- In Section III, we will detail the design of our anomaly detection system. We will explain the different steps of the process, including network flow aggregation, anomaly scoring with base-learners, generation of visual representations of anomalies, and attack pattern recognition.
- Section IV will present the results of each component of our system. We will present the performance of each base-learner as well as the final model.
- Section V presents why our system is explainable and how it helps both the overall accuracy and the decision process of a network analyst.
- Finally, in Section VI, we summarize our main contributions and emphasize the advantages of our network attack detection system.

II. Related Work

Our research investigates mechanisms to reconstruct attack patterns based on multiple unsupervised learning detection techniques. Our work addresses issues related to (1) ensemble learning, (2) attack patterns detection, and (3) explainable AI.

a) Ensemble learning: Regarding ensemble learning systems for network anomaly detection, Vanerio and Casas [1] compared multiple *meta-learners* for detecting attacks. A meta-learner combines outputs of a set of base-learners to return a more accurate detection. As an example of meta-learner, the authors considered a weighted-majority voting algorithm where the weights were defined depending on the accuracy of the base-learner. We have taken a different approach in which the meta-learner is another ML layer, a CNN, that takes as input a visual representation of the base-learners' outputs and detects attack patterns on them. Mirsky et al. [7] proposed Kitsune, an ensemble of autoencoders for detecting network anomalies. The system relies on autoencoders, which are often considered as unsupervised learning techniques because they use unlabeled data, but autoencoders still require a training phase on benign data. Kitsune stacks autoencoders, by using another autoencoder as a meta-learner to process their anomaly scores. The approach to combine base-learners is close to ours, except that we address heterogeneous algorithms and data features combinations.

b) Attack patterns detection: Regarding attack patterns detection, Zhou et al. [8] proposed a system using LSTM to detect multi-stage attacks. Their model treats sequences of alarms generated by the NIDS Snort and addresses the problem of long-term dependency between the alarms. Ghafir et al. [9] proposed a system for the detection and prediction of advanced persistent threats (APT). The system uses the Hidden Markov Model (HMM) to detect the most probable APT scenario given the raised alarms. Then, it forecasts the next step of the ongoing APT. A significant difference with our work is that their system is trained on a predefined attack lifecycle [10], [11], whereas our system learns attack patterns directly from the data, without prior knowledge of specific attack lifecycles. Wang et al. [6] converted raw traffic data, specifically pcap files, directly into images. They also observed and identified attack patterns using this image-based representation.

c) Explainable AI: Wei et al. [12] observed that current general-purpose explanation methods, such as SHAP [13] and LIME [3], are not suitable for NIDSs because they do not handle dependencies of network flows' features. To overcome this issue, the authors proposed data-driven explanation methods for Deep Learning (DL)-based NIDSs, that are based on history inputs. Based on the extracted feature importance, the system generates defense rules to block malicious activities. Han et al. [4] addressed ML explainability by proposing a system to interpret existing unsupervised, DL-based NIDS. The system analyzed a given model detection by providing the most important features and describing their meaning so that a security expert can understand them. Instead of relying on *ad hoc* descriptions of the features, we design a more transparent system and provide a visual representation of traffic anomalies that can be easily interpreted by the security analyst.

III. System design

A. System overview

Our system takes networks flows as input (e.g., a pcap capture), splits them in time frames of Δ_T, and outputs, for each time frame, whether there was an attack or not during that time frame. Figure 1 shows an overview of our system.

First, the system aggregates flows by source and by destination on shorter time intervals Δ_t. It computes features from these aggregates, such as, for instance, for an aggregate by source, the number of destinations that the source exchanged traffic with during that time interval. These features are similar to what was used in prior work [14], [15], and a complete list is given in Table I. This corresponds to the step 1 of Figure 1, which is detailed in Section III-B.

Then, the system gives these features to a set of well selected base-learners. One base-learner can for instance use Isolation Forest [16], whereas another one can use Local Outlier Factor [17]. Each of these base-learners outputs an anomaly score for each aggregate. As we are using three base-learners (Section III-C2), each aggregate can be represented as a vector of three values in the three RGB color channels. Concatenated together, they form color-encoded segments representing the anomaly scores of a source or a destination over time intervals of Δ_t. This corresponds to step 2 in Figure 1, whic is detailed in Section III-C.

Finally, these color-encoded segments are put side-by-side to form an image that represents network anomalies during a larger timeframe Δ_T. Each line in the image corresponds to the evolution of anomaly scores for aggregates of a specific source or destination over the timeframe Δ_T. This image is given to a CNN, which is a layer of supervised learning that computes whether the image contains an attack or not. This is shown in step 3 of Figure 1, which is detailed in Section III-D.

The rest of the section will detail, for each step, the motivation, challenges, and design choices in building each of these components.

B. Aggregation of network flows

One classic question in designing anomaly detection techniques is how to represent the input traffic data. For instance, the CIC-CSE-IDS2018 dataset [18], that we use in that paper to evaluate our system, proposes 83 network flow features to evaluate ML-based NIDSs, where a network flow is identified by the 5-tuple (source IP, destination IP, source port, destination port, protocol). A traditional approach is to directly give these features to a machine learning algorithm that will compute whether a flow is anomalous or not. There are two problems with this approach: (1) The number of features tends to make the anomaly detection problem hard, also called curse of dimensionality [14], [15] (2) The result is rarely explainable. For instance, it is hard to understand why a deep neural network did classify a flow as anomalous [7]. To overcome these problems, we reduce the number of features by aggregating the flows by source and by destination (Table I), reducing the number of features from 83 to 9. This reduction also helps for improving the explainability of our system (Section III-C).

We choose to aggregate the flows by source and by destination, because our intuition is that some types of attacks are better identified by aggregating by source, and some others are better identified by aggregating by destination. For instance, certain types of a DDoS attack will involve a lot of traffic sent to a particular destination, so we will probably observe an anomalous value in the n_src_ip feature. We show that aggregates by source and by destination are complementary, and give better results than aggregates by source *or* by destination taken individually (Section IV-C).

TABLE I: Aggregates features

Feature	Aggregation key	Description
n_dst_ip	IPsrc	Number of destination IP addresses
n_src_ip	IPdst	Number of source IP addresses
n_dst_ports	IPsrc & IPdst	Number of destination ports
n_src_ports	IPsrc & IPdst	Number of source ports
n_fwd_pkts	IPsrc & IPdst	Number of forward packets
n_bwd_pkts	IPsrc & IPdst	Number of backward packets
sum_flx_dur	IPsrc & IPdst	Sum of flows duration
tot_flx	IPsrc & IPdst	Number of flows
sum_pkts_size	IPsrc & IPdst	Sum of packets size
std_pkt_size	IPsrc & IPdst	Standard deviation of packets size

These aggregates are computed over time intervals of $\Delta_t = 2$ minutes. We make this decision to mitigate the impact of legitimate but sudden changes in network traffic (such as variations on weekdays or weekends). Through empirical evaluation, we determined that a time interval of 2 minutes provides effective results (Section IV-B).

Moreover, to further reduce the dimensionality of our data, we only compute the aggregates on IP addresses that are internal to the network under consideration (i.e., the machines belonging to an enterprise network). To be clear, there is no aggregates per destination for public destinations in the Internet.

In summary, aggregating network flows by source IP address and destination IP address provides both a dimensionality reduction that benefits unsupervised machine learning algorithms used in Section III-C and a better explainability for security analysts.

C. Ensemble anomaly scoring

This component of the system (Step 2 on Figure 1) takes as input the aggregates computed in the previous section, i.e., aggregates by source and by destination over time interval of $\Delta_t = 2$ minutes. Its goal is to obtain an anomaly score for each input aggregate. Our idea is to use ensemble learning with multiple base-learners to achieve this goal, and we identify two challenges: (1) For our system to be explainable, how can we compute a score that both represent a degree of anomaly and is easily understandable by an analyst? (2) Which base-learners should we select to maximize the performance?

1) Unsupervised anomaly scoring: Different base-learners can have very different approaches to compute the anomaly scores, and this represents a challenge for us. Indeed, we cannot run a base-learner directly on the features computed from the aggregates, as the anomaly score would probably not be meaningful for an analyst. For instance, the clusters computed by DBSCAN [19] in high-dimensional data and thus the anomalies are hard to interpret. Instead, we run each base-learner on subspaces of $k = 2$ features among the $n = 9$ aggregate features of Table I, similar to what was done in the UNADA prior work [14], [15]. Each base-learner is therefore run on $\binom{2}{9} = 36$ pairs of features. The anomaly score of an aggregate is then, for each base-learner, the number of pairs of features that this base-learner considered as anomalous,

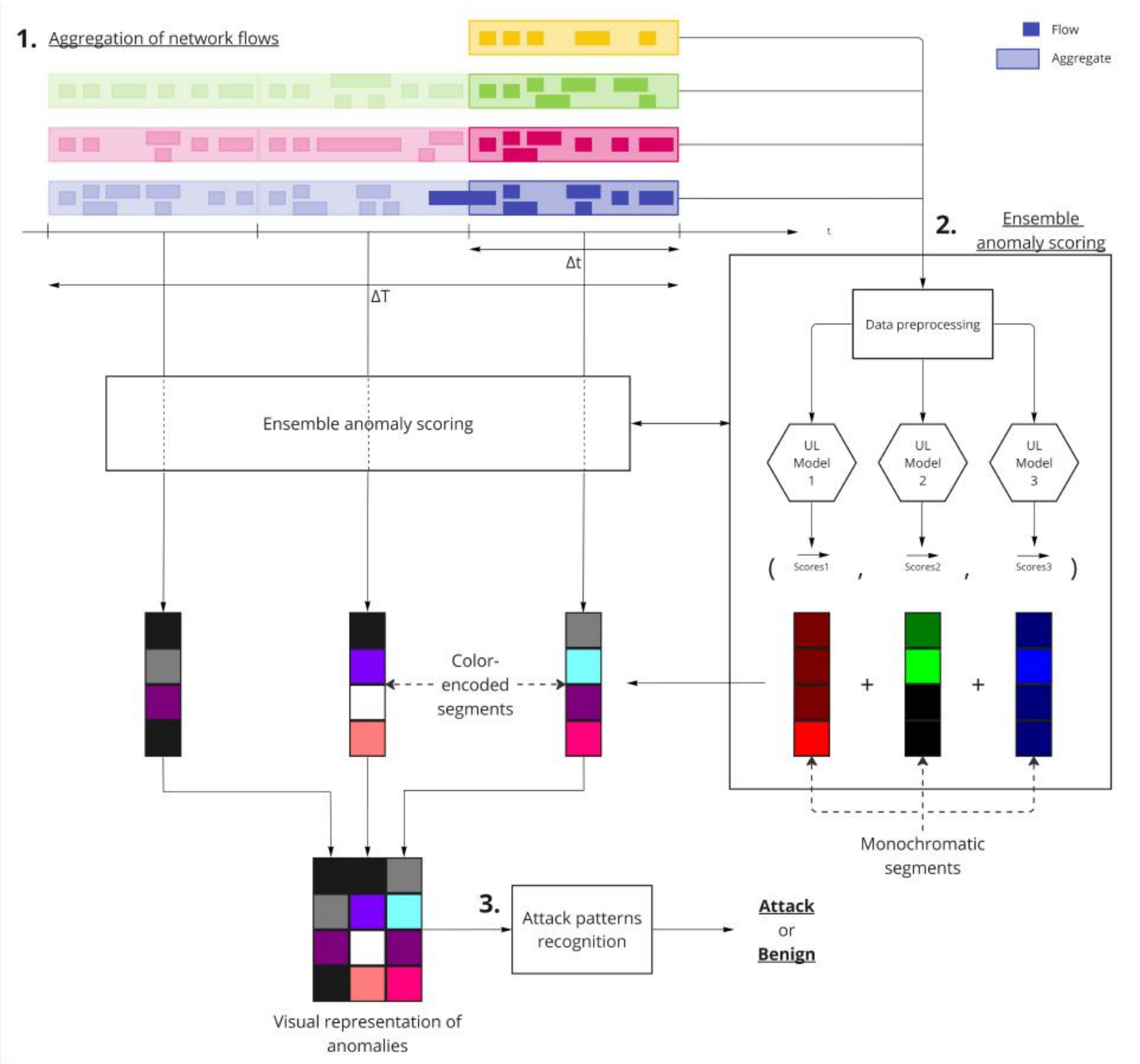


Fig. 1: System overview. The system analyzes a traffic capture over a period Δ_T. First, it aggregates network flows from the capture that belong to the same time interval Δ_t and computes their features. Next, it applies a set of three unsupervised base-learners to assign anomaly scores to the aggregates. After that, the system generates a visual representation of the detected anomalies, enabling a visualization of potential attack patterns. Finally, these representations are analyzed by an attack pattern recognition module, here a CNN, which determines whether the network is under attack during the period Δ_T.

ranging from 0 to 36. The way a pair of features is considered anomalous is specific to each base-learner, depending on the algorithm used. This choice facilitates the interpretation, as one can retrieve which pairs of network features were identified as anomalous, which are meaningful for a network analyst.

2) Base-learners selection: We select three base-learners among unsupervised anomaly detection algorithms, to be able to enlarge the scope of attacks that can be detected. We use three as their output can then be converted into images, each base-learner representing a color in the RGB channels (Section III-D). This choice improves explainability as the analyst can benefit from a visual representation of the anomaly scores. Prior work has also showed that adding more base-learners does not necessarily improve performance [20].

Our approach to select these three algorithms is data driven: we evaluate the performance of all the possible combinations of three algorithms among the algorithms available in the popular machine learning libraries scikit-learn [21] and PyOD [22] on the CIC-CSE-IDS2018 dataset [18], and select the best combination, using the standard metrics of true postive rate (TPR) and false positive rate (FPR) (Section IV-B).

These algorithms have different approaches, and our intuition is that ensemble learning would work well to improve

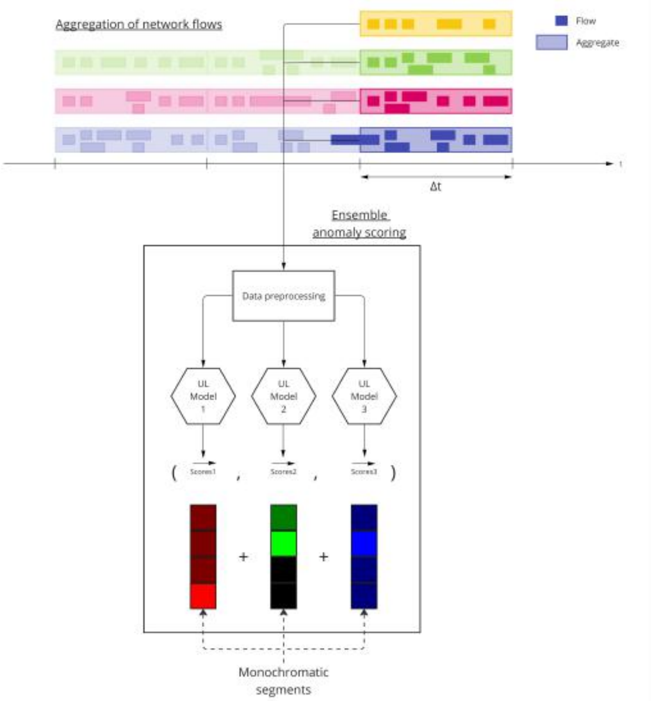


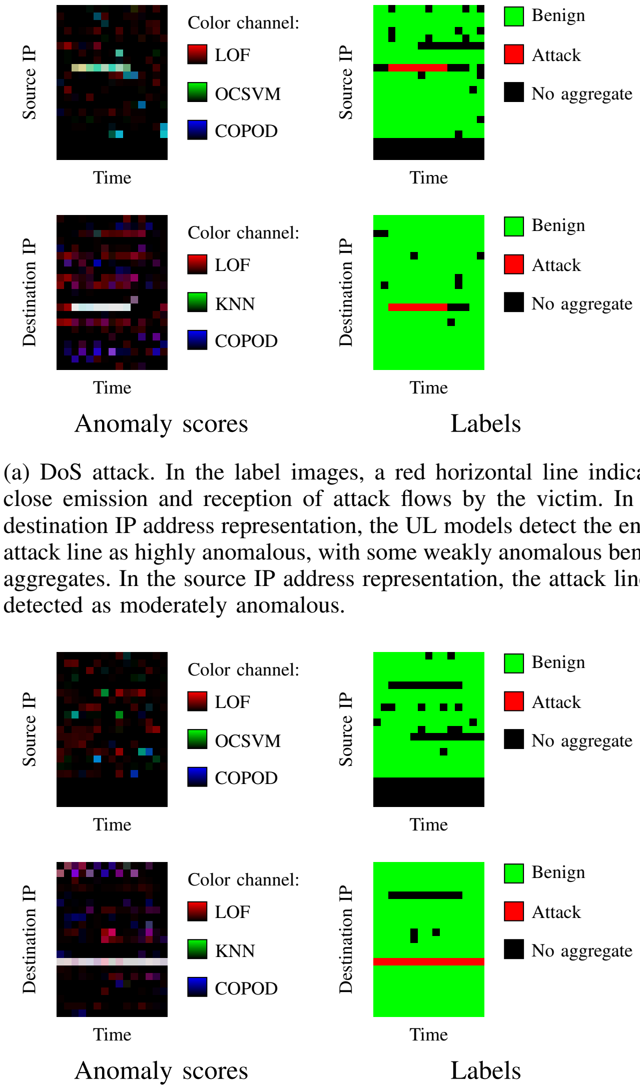
Fig. 2: Anomaly scoring by three unsupervised base-learners during a time interval Δ_t. This analysis generates a vertical color-encoded segment that is part of the visual representation of anomalies generated in Figure 4. Four network flows are grouped into aggregates during a time interval Δ_t. Then, the features of the aggregates are extracted and prepared before being evaluated by the three unsupervised base-learners. Each base-learner assigns an anomaly score to each aggregate. A color is assigned to each model, allowing the representation of anomaly scores on a monochromatic segment. By overlaying the monochromatic segments from the base-learners, we obtain a color-encoded segment that represents the anomalies detected by the set of base-learners.

the accuracy of the system [20]. Namely, we tested:

- *Isolation Forest (IF)* [16] detects anomalies using isolation. The algorithm recursively constructs a random feature selection and split values to isolate data samples. The anomaly score is determined by the number of splittings required for isolation.
- *Local Outlier Factor (LOF)* [17] compares the local density of a data sample with that of its neighbors to detect anomalies.
- *DBSCAN* [19] groups data samples into clusters based on their proximity. If a group contains enough data samples, it forms a cluster; otherwise, they are classified as outliers.
- *One-Class SVM (OCSVM)* [23] defines a hyperplane to separate data samples and detects anomalies based on their distance from the hyperplane.
- *Unsupervised KNN* [24] is a proximity-based model that uses the distance to the kth nearest neighbor as an outlier score.
- *COPOD* [25] is a probabilistic model for anomaly detection.

Although our choice of the base-learners depends on the dataset, in practice, an enterprise could reuse our methodology to choose the base-learners that are the best suited to its traffic.

D. Attack patterns recognition



(a) DoS attack. In the label images, a red horizontal line indicates close emission and reception of attack flows by the victim. In the destination IP address representation, the UL models detect the entire attack line as highly anomalous, with some weakly anomalous benign aggregates. In the source IP address representation, the attack line is detected as moderately anomalous.

(b) Brute-force attack. In the label images, the victim is shown to receive but not emit attack flows. The attack is only visible in the destination IP address representation, where the UL models detect the entire attack line as highly anomalous, along with some weakly anomalous benign aggregates.

Fig. 3: Anomaly Representations and Labels. The figures show anomaly representations and labels for a brute-force attack and a DoS. The vertical axis represents IP addresses, while the horizontal axis represents time intervals. The left images display base-learner outputs using color channels, while the right images show attack and benign labels (red and green pixels, respectively).

In the previous section, we selected a set of base-learners which detect anomalous aggregates and output an explainable result. One problem is that they might disagree on an aggregate, some of them might consider it as anomalous, whereas some others might not, and there is no simple rule to decide which one is right. One reason that could make them disagree is that some base-learners are better than others at identifying some types of attacks, and the challenge is to take advantage of their complementarity. Our idea is to use a meta-learner on top of our three base-learners that will learn which base-learner is more suited for which type of attack. Indeed, we show in Section IV-B that we cannot simply select the decision of the base-learner having the best performance [1].

To translate this idea into a design, we map the anomaly scores of the three base-learners into colors of the RGB channels, with the intensity of a color being proportional to the anomaly score. These pixels, representing aggregates over time intervals of two minutes (Section III-B), are then put together to form an image of 30 minutes, where a line corresponds to a source or destination IP address, depending on whether the aggregation is made by source or destination, and each column corresponds to a time interval of two minutes. The ideas behind building these images are two-fold: (1) Most of the attacks last more than 2 minutes, and moreover some of them can have complex patterns. By building sequences of anomaly scores, we hope that our meta-learner will perform better at detecting these complex attacks. (2) The images improve explainability as they provide a useful visualization of the anomalous IP addresses and can be easily translated to the original anomalous networking features.

This image is then given to a CNN. This CNN is used as a meta-learner to identify attack patterns. The CNN takes as input the two matrices, one representing the aggregates by source and the other the aggregates per destination, and outputs a decision of whether there was an attack during the 30 minutes represented by the image.

The CNN requires a labeled dataset for training and validation. We choose to label an image as an *attack* if this image contains at least one pixel representing an attack, i.e., the corresponding aggregate is labeled as malicious traffic in our dataset [18] (Section IV-A). Otherwise, the image is considered as benign.

An example of how this component works is given in Figure 3, which represents a DoS and a brute-force attacks. The output of the base-learners are represented on the images of the left. A white pixel means that all the base-learners diagnosed the aggregate as highly anomalous, whereas a black pixel means that none of the base-learners considered that aggregate as anomalous. The images on the right represent our ground truth labeled dataset [18], with a green pixel represents benign traffic, whereas a red pixel represents an attack. Black pixels represent just an absence of traffic for an IP address during the 2 minutes. We observe that for both the DoS and the brute-force attack, the anomaly scores given by the base-learners are generally high as they are closer to the white color than to the black. However, notice that on the image on the top left of the DoS attack, not all the base-learners had high anomalous scores, as some pixels are turquoise and further from white, showing the necessity of having a meta-learner to find the right decision.

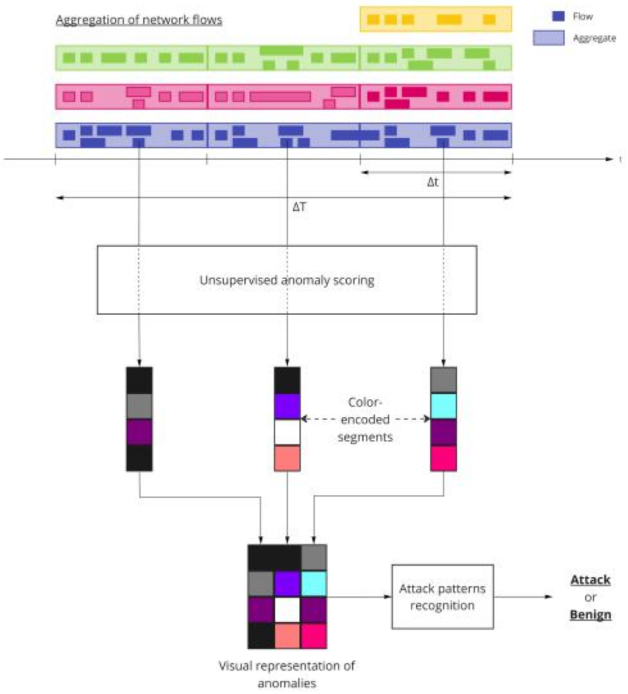

Aggregation of network flows
Flow
Aggregate
t
Δt
ΔT
Unsupervised anomaly scoring
Color-encoded segments
Attack patterns recognition
Attack
or
Benign
Visual representation of anomalies

Fig. 4: Attack pattern recognition on a visual representation of the detected anomalies during a timeframe Δ_T. The timeframe Δ_T consists of, in this example, 3 consecutive time intervals Δ_t. The network flows from each time interval Δ_t are analyzed as shown in Figure 2. All the anomalies detected by the $N = 3$ base-learners within a specific time interval Δ_t are visually represented using color-coded segments. By arranging the 3 segments side by side, we obtain a visual representation of network anomalies detected by the $N = 3$ base-learners during the timeframe Δ_T.

IV. Results

Our system achieves an overall performance of a True Positive Rate 0.9898 and a False Positive Rate of 0.0093, corresponding to three false positives and one false negative (Section IV-C) that we can mitigate thanks to explainability (Section V). We achieve this performance by having the following intermediate results: (1) A good combination of three base learners is better than only always trusting the best of the three (Section III-C). (2) Adding the CNN as a meta-learner on top of the base-learners greatly improves the performance (Section III-D) (3) Computing the aggregates by source and by destination has a better performance than only computing by source or destination (Section III-D). Finally, we describe how explainability helps the system to be more accurate in Section V, as well as how explainability can improve the decision process for a network security analyst.

A. Dataset

We evaluated our system on the CIC-CSE-IDS2018 dataset [18], which, to our knowledge, is the only recent, representative, labelled dataset that is publicly available for evaluating NIDSs. This dataset represents a realistic enterprise network with 450 machines, and an attacking infrastructure consisting of 50 machines. The dataset contains 10 days of network captures, covering various simulated attack scenarios including brute-force attacks, denial-of-service (DoS) attacks, distributed denial-of-service (DDoS) attacks, web attacks, infiltration attacks, and botnet attacks. In particular, the dataset provides the hours of the attacks and the source IP addresses making the attack and the destination IP addresses receiving it, so we labeled the network flows accordingly. An aggregate by source or by destination is labeled as an attack if the source or the destination emits or receive traffic from or to an attacking IP address. Finally, an image is labeled as an attack if one aggregate in the image is labeled as an attack.

1) Aggregation of network flows: Table II describes the distribution of aggregates for each traffic category. There are fewer aggregates by source IP addresses compared to destination IP addresses, indicating that there are more receiving machines than emitting machines in our dataset. In addition, the density of attacks is higher in the aggregates by destination IP than in those by source IP.

2) Transforming the anomaly scores into images: After putting together the aggregates into images representing the anomaly scores obtained during time intervals of 30 minutes, we obtain 4214 pairs of images (one for aggregation by source and one by destination).

B. Ensemble Anomaly Scoring

The unsupervised anomaly scoring component generates visual intermediate representations of network anomalies based on a set of base-learner detections (Figure 3).

Table III shows the performance of the top five combinations of base-learners, also called *stacked models*, when using aggregates by source and by destination. It shows their true positive rates (TPR) and false positive rates (FPR) on the aggregates by source and destination. We consider that a combination labels an aggregate as an attack if one of its base-learners detects the aggregate as an attack.

The top combinations are (LOF, OCSVM, COPOD) for analyzing the aggregates by source IP address, and (LOF, KNN, COPOD) for analyzing the aggregates by destination IP address. If we empirically select the combination with the best results, our hypothesis is that these combinations work well in practice because they have different methodology to detect anomalies. Specifically, LOF is a local density-based algorithm, OCSVM and KNN rely on distance-based approaches, and COPOD is probabilistic based.

We retrieve now a known result in ensemble learning that it is better to use a stacked model rather than always using the best base-learner [20]. Table IV shows the TPR and the FPR of each base-learner and the best stacked model. The LOF, which has the best TPR of the base-learners with 0.8923 on the aggregates by source, does not detect some of the attacks identified by the stacked model, which has a TPR of 0.9878. However, the stacked model has a higher TPR. The process of reducing this TPR is made by the meta-learner (Section IV-C).

C. Attack Patterns Recognition

This last section of the evaluation looks into the performance of our meta-learner, the CNN (Section III-D). Our dataset of images is split into training, validation, and evaluation sets with an 80-10-10 ratio, respectively. The training set was used to train the CNN, while the validation set is used for hyperparameter tuning [26]. The evaluation set is reserved for the final assessment of the attack patterns recognition module's performance.

Table VI shows the F-score and confusion matrix of our meta-learner, i.e., our CNN, which analyzes aggregates by source and destination IP address, called the combined model. There are three main results from this table: (1) The overall performance of the system is good: it only has three false positives, and one false negative, but we show that explainability can help understand why the system did not detect this attack and that it would have been detected in practice thanks to explainability (Section V). (2) The FPR of the meta-learner is way better than the FPR of the best combination of base-learners, with 0.0093 versus 0.6763, while the TPR is similar (0.9898 versus 0.9878), showing the added value of the meta-learner. (3) The two representations, aggregates by source and destination, are complementary. The combined model has a higher F-score than the two other models, only using aggregates by source or destination, although the F-score are close.

It is worth noting that our system does not require the attack to last throughout the entire capture to detect it, even if attacks that have been ongoing for a long time are easier to detect and visualize. In fact, our system demonstrates the capability to detect most attacks within 2 minutes, as soon as the first attack aggregate appears in the capture. Further research could explore reducing the time required for effective attack detection, potentially by implementing time-sliding windows [27].

V. Explainability improves accuracy

The goal of our system is not to achieve the best and perfect accuracy blindly, only relying on statistical methods, like most prior work. Instead, we aim a perfect accuracy by involving the end user of an NIDS, i.e., a human analyst who needs to understand the output of the NIDS to decide whether to trigger some actions if one considers that their network is under attack. Our system is explainable-by-design, as each successive component does not hinder the possibility to easily retrieve which network features triggered the anomaly detection. A report is generated whenever the system labels an image with an attack, easing the process of an analyst. To illustrate why we can in fact obtain a perfect accuracy on our dataset thanks to explainability, we looked into the image corresponding to the undetected attack in Table VI. What we

TABLE II: Distribution of aggregates by source and destination IP address for each traffic category

	Source IP		Destination IP	
Traffic Category	**Count**	**Proportion (%)**	**Count**	**Proportion (%)**
Brute-force	41	0.004455	97	0.009547
Denial of service	13	0.001412	70	0.004231
Web	0	0	133	0.013089
Infiltration	32	0.003477	76	0.007480
Bot	1500	0.162978	0	0
DDoS	58	0.006302	62	0.006102
Benign	918723	99.821376	1015651	99.956894
Total	**920367**	**100**	**1016089**	**438**

TABLE III: Top 5 combinations of 3 base-learners for aggregates by source and destination IP address. The combinations are ranked according to two criteria: the true positive rate (TPR), representing the percentage of accurately detected attacks, and the false positive rate (FPR), indicating the proportion of false alarms. The subsets of base-learners in these combinations have demonstrated the highest accuracy in identifying attacks while minimizing false positives.

Source IP			Destination IP		
Subset of base-learners	**TPR**	**FPR**	**Subset of base-learners**	**TPR**	**FPR**
(LOF, OCSVM, COPOD)	0.9878	0.6763	(LOF, KNN, COPOD)	0.9384	0.3994
(IF, LOF, OCSVM)	0.9811	0.6704	(LOF, OCSVM, COPOD)	0.9811	0.4437
(LOF, OCSVM, KNN)	0.9732	0.6762	(LOF, OCSVM, KNN)	0.9315	0.4312
(LOF, DBSCAN, OCSVM)	0.9726	0.6620	(LOF, DBSCAN, COPOD)	0.9292	0.3917
(IF, OCSVM, COPOD)	0.9690	0.4937	(IF, LOF, COPOD)	0.9292	0.3939

TABLE IV: True positive rate (TPR) and false positive rate (FPR) for the base-learners vs the stacked model.

Source IP			Destination IP		
Base-learner	**TPR**	**FPR**	**Base-learner**	**TPR**	**FPR**
LOF	0.8923	0.3754	LOF	0.9041	0.3366
OCSVM	0.8394	0.4634	KNN	0.8904	0.1862
COPOD	0.7652	0.1467	COPOD	0.8744	0.1795
Stacked Model	**0.9878**	**0.6763**	**Stacked Model**	**0.9384**	**0.3994**

TABLE V: Distribution of attack and benign images in the dataset split. The table presents the distribution of attack and benign images across the dataset split, which was divided into training, validation, and test datasets in an 80-10-10 ratio.

	Attack		Benign	
	Count	**Proportion (%)**	**Count**	**Proportion (%)**
Training	776	23.70	2498	76.30
Validation	102	24.23	319	75.77
Test	98	23.22	324	76.78
Total	**976**	**23.16**	**3238**	**76.84**

actually observed is that this image corresponded to the end of an attack, that would have been detected by prior images as the attack lasted over several images. So in practice, this attack would have been detected and we would have 0 false negative on our evaluation dataset.

VI. Conclusion

We have proposed an explainable-by-design network attack detection system. In our approach, we used unsupervised techniques to detect anomalies on aggregates based on source and destination IP addresses. The results produced by our system are interpretable and understandable for security analysts. The outputs include the IP addresses associated with the aggregates detected as anomalous, allowing analysts to relate them to specific machines in the network. Additionally, the anomalous feature pairs are also reported, providing an additional level of detail on the detected anomalies. To represent these anomalies, we used images generated from a set of unsupervised base-learners. These images enable security analysts to visually observe the attack patterns detected by the system. Finally, we analyzed these traffic representations using a CNN to recognize attack patterns.

Our approach aims to design a more transparent attack detection system that allows security analysts to understand the decisions made by the system. The evaluation of our system on the CIC-CSE-IDS2018 dataset [18] demonstrated reasonable accuracy, but more importantly, the errors made by the system are easily identifiable and can be analyzed by security analysts.

Acknowledgments

We gratefully acknowledge the financial support provided by Custocy through the CIFRE (Industrial Agreement for Training through Research) program. We also thank our colleagues at Custocy for their support in using the Netsens tool. Furthermore, we extend our appreciation to the team responsible for the LAAS-CNRS pfcalcul platform for granting us access and providing technical support.

TABLE VI: Performance metrics and confusion matrix of CNN on aggregates by source IP address, destination IP address, and combined CNN

	TPR	**FPR**	**F-score**	**Confusion Matrix** $\begin{bmatrix}\textbf{TP} & \textbf{FP}\\ \textbf{FN} & \textbf{TN}\end{bmatrix}$
CNN Source	0.9796	0.0062	0.9905	$\begin{bmatrix}96 & 2\\ 2 & 322\end{bmatrix}$
CNN Destination	0.9796	0.0093	0.9882	$\begin{bmatrix}96 & 3\\ 2 & 321\end{bmatrix}$
Combined CNN	**0.9898**	**0.0093**	**0.9906**	$\begin{bmatrix}\mathbf{97} & \mathbf{3}\\ \mathbf{1} & \mathbf{321}\end{bmatrix}$

REFERENCES

[1] J. Vanerio and P. Casas, “Ensemble-learning Approaches for Network Security and Anomaly Detection,” in *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. Los Angeles CA USA: ACM, Aug. 2017, pp. 1–6.

[2] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, “An Adaptive Ensemble Machine Learning Model for Intrusion Detection,” *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144.

[4] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, “DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 3197–3217.

[5] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, “Detection of advanced persistent threat using machine-learning correlation analysis,” *Future Generation Computer Systems*, vol. 89, pp. 349–359, Dec. 2018.

[6] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, “Malware traffic classification using convolutional neural network for representation learning,” in *2017 International Conference on Information Networking (ICOIN)*, Jan. 2017, pp. 712–717.

[7] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection,” in *Proceedings 2018 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, Feb. 2018.

[8] P. Zhou, G. Zhou, D. Wu, and M. Fei, “Detecting multi-stage attacks using sequence-to-sequence model,” *Computers & Security*, vol. 105, p. 102203, Jun. 2021.

[9] I. Ghafir, K. G. Kyriakopoulos, S. Lambotharan, F. J. Aparicio-Navarro, B. Assadhan, H. Binsalleeh, and D. M. Diab, “Hidden Markov Models and Alert Correlations for the Prediction of Advanced Persistent Threats,” *IEEE Access*, vol. 7, pp. 99 508–99 520, 2019.

[10] E. Hutchins, M. Cloppert, and R. Amin, “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains,” in *ICIW2011-Proceedings of the 6th International Conference on Information Warfare and Secuirty: ICIW*. Academic Conferences Limited, 2011, pp. 113–125.

[11] D. McWhorter, “Mandiant Exposes APT1 — One of China’s Cyber Espionage Units & Releases 3,000 Indicators,” *Mandiant, February*, 2013.

[12] F. Wei, H. Li, Z. Zhao, and H. Hu, “xNIDS: Explaining deep learning-based network intrusion detection systems for active intrusion responses,” in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 4337–4354.

[13] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[14] P. Casas, J. Mazel, and P. Owezarski, “UNADA: Unsupervised Network Anomaly Detection Using Sub-space Outliers Ranking,” in *10th IFIP Networking Conference (NETWORKING)*, vol. LNCS-6640. Springer, May 2011, pp. 40–51.

[15] J. Dromard, G. Roudière, and P. Owezarski, “Online and Scalable Unsupervised Network Anomaly Detection Method,” *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, Mar. 2017.

[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in *2008 Eighth IEEE International Conference on Data Mining*, Dec. 2008, pp. 413–422.

[17] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 93–104.

[18] I. Sharafaldin, A. Habibi Lashkari, and A. 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*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116.

[19] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD’96)*, vol. 96, no. 34, 1996, pp. 226–231.

[20] S. Džeroski and B. Ženko, “Is Combining Classifiers with Stacking Better than Selecting the Best One?” *Machine Learning*, vol. 3, no. 54, pp. 255–273, 2004.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, “Scikit-learn: Machine Learning in Python,” *MACHINE LEARNING IN PYTHON*, p. 6, 2011.

[22] Y. Zhao, Z. Nasrullah, and Z. Li, “PyOD: A Python Toolbox for Scalable Outlier Detection,” *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[23] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, “Support Vector Method for Novelty Detection,” in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 1999.

[24] F. Angiulli and C. Pizzuti, “Fast Outlier Detection in High Dimensional Spaces,” in *European conference on principles of data mining and knowledge discovery*, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Berlin, Heidelberg: Springer, 2002, pp. 15–27.

[25] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, “COPOD: Copula-Based Outlier Detection,” in *2020 IEEE international conference on data mining (ICDM)*. IEEE, Nov. 2020.

[26] T. O’Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, “KerasTuner,” 2019.

[27] J. Dromard and P. Owezarski, “Integrating short history for improving clustering based network traffic anomaly detection,” in *International Workshop on Autonomic Systems for Big Data Analytics (ASBDA 2017)*, Tucson, United States, Sep. 2017, p. 8p.