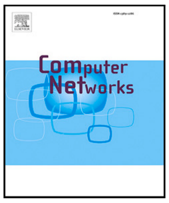
Computer Networks 265 (2025) 111264

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



HERO: From **H**igh-dimensional network traffic to z**ERO**-Day attack detection

Jesús F. Cevallos M., Alessandra Rizzardi[1], Sabrina Sicari[*,2], Alberto Coen-Porisini

Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, via O. Rossi 9, Varese (VA), 21100, Italy

A R T I C L E I N F O

Keywords:
Zero-day attack detection
Neural algorithmic reasoning
Raw traffic analysis

A B S T R A C T

Recent trends in zero-day attack (ZdA) detection use *collective* anomaly detection to give insights on out-of-distribution anomalies in a *zero-shot* fashion. Among these, existing frameworks propose the use of specialised labelling strategies to mimic a step-wise abstract anomaly detection algorithm that generalise ZdA-detection over low-dimensional traffic-flow statistics. To enlarge such applicative scenarios, this paper proposes HERO, which is compatible with **H**igh-dimensional raw-network traffic captures when performing z**ERO**-day attack detection. To reach convergence over such a high-dimensional and noisy input space, HERO decouples the *representation* task and the correspondent gradient updates from the *discriminative* task, following the *neural algorithmic reasoning* blueprint. Specifically, a neural processor is first trained on the discriminative task using synthetic data, and the weights are then frozen. A second training phase successfully optimises the encoding and decoding networks using raw-traffic captures and the algorithmically-aligned processor. Experiments with well-known intrusion detection datasets demonstrate the crucial advantage of using a two-stage training framework to achieve convergence. To the best of the authors' knowledge, HERO is the first deep learning-based instrument that performs collective anomaly detection and categorisation over raw network traffic on a zero-shot basis, i.e., without using labels.

1. Introduction

Zero-day attack (ZdA) detection is gaining momentum across the network intrusion detection research community, where Machine Learning (ML) has established as a *de facto* standard technique [3,4]. Among the vast plethora of ML-inductive biases used to detect ZdAs, *metric-based meta-learning* [5] has proven among the most effective in terms of training-data efficiency. In this meta-learning paradigm, samples from classes that were unseen during training are successfully classified, providing a handful of labelled samples for each new class at inference time [6,7]. However, the assumption of having labels for ZdA samples – at training or inference time – implicitly breaks the "zero-day" status of these samples, by definition [8–10]. Instead, realistic ZdA-detection should be done on a *zero-shot* basis, i.e., without labels [11].

Obviously, learning without labels is a particularly difficult task for any statistical learning-based approach[3]. For this reason, anomaly-based detection [12], where detecting specific types of attack is not necessary, has such a protagonistic role in current real-world Network Intrusion Detection (NID) systems [13]. Moreover, beyond detecting unusual traffic observations, a recent research trend focuses on clustering anomalies into different groups, in other words, performing *Collective* Anomaly Detection (CAD) [14]. After clustering anomalous traffic flows, any NID system can then make inferences about their benign or malign nature.[4]

Not being able to use labels, inferences about the benign/malign nature of collective anomalies can only be made based on other types of context knowledge. In this respect, common practices include importing insights from cyber-threat intelligence sources or relying on the *degree of novelty* (DoN) of the current anomalous cluster with respect to the known traffic patterns [8,15,16]. This work studies the second approach, where the main assumption made is that knowing the DoN of an anomalous cluster can help experts refine the likelihood of it being related to benign or malign traffic.

In this respect, the NERO framework [9] has recently been proven effective in obtaining inferences about the degree of novelty of ZdAs, using the *hint-labelling* strategies inherited from the *Neural Algorithmic*

* Corresponding author.
E-mail addresses: jf.cevallosmoreno@uninsubria.it (J.F. Cevallos M.), alessandra.rizzardi@uninsubria.it (A. Rizzardi), sabrina.sicari@uninsubria.it (S. Sicari), alberto.coenporisini@uninsubria.it (A. Coen-Porisini).
[1] Member, IEEE.
[2] Senior Member, IEEE.
[3] From an epistemic point of view, the task is in fact impossible if one does not have an exhaustive-enough world-model backing up some form of *concept synthesis [1,2].*
[4] Not every anomalous traffic observation necessarily corresponds to an attack.

https://doi.org/10.1016/j.comnet.2025.111264
Received 28 December 2024; Received in revised form 11 March 2025; Accepted 24 March 2025
Available online 21 April 2025

Reasoning (NAR) [17] paradigm. More specifically NERO performs CAD and DoN-classification on test-time samples whose classes were not included in training data and that *remain* unlabelled at test-time.

Extending NERO *to classify high-dimensional data.* The main limitation of the NERO framework is its reliance on low-dimensional traffic statistics, e.g., mean payload per packet, mean packet rate, etc. This assumption makes it difficult to deploy NERO in network scenarios where such information may not be available for each traffic flow. In this respect, the question that arises is if *the* NERO *framework can be extended to operate with raw network traffic, which is high-dimensional*. This paper is an affirmative answer to such a question.

The "zero-touch" vs. "fast" network intrusion detection trade-off. The research landscape in network intrusion detection techniques has one major non-functional requirement: timeliness [18]. A direct approach to reduce detection latency consists of, for example, creating more efficient neural architectures [9,19] or traffic collection technology [20]. A less immediate approach for latency reduction, more strictly tightened to DL-based approaches, is instead focused on reducing the preprocessing overhead and exploiting the automatic feature extraction capabilities of deep Artificial Neural Networks (ANN) [10]. Without loss of generality, this paper aims to contribute to the NID research community by exploring the latter class of techniques. More specifically, HERO is tested on the classification of raw high-dimensional traffic traces, letting aside the exploration of efficient traffic statistics extraction tools [21].

(a) Main contribution: This paper presents HERO, a Deep Learning (DL) pipeline that takes in input **H**igh-dimensional raw network traffic and performs z**ERO** day attack detection in the form of *hierarchical* collective anomaly detection where anomalies can have different degrees of novelty. The main novelty of HERO with respect to its predecessor, NERO, is that, beyond hint labelling, HERO takes advantage of the full power of the NAR paradigm in terms of training procedure. More specifically, HERO divides the training of the pipeline on two main phases. In the first phase, the *neural processor* is trained to detect ZdAs using low-dimensional synthetic data. Then, on a second phase, the weights of the processor are frozen, and a neural encoder and decoder are attached to it to converge to the same task over raw network bytes. To the best of the authors' knowledge, HERO is the first DL-based instrument that performs collective anomaly detection and categorisation over raw network traffic on a zero-shot basis, i.e., without labels[5]

(b) Outline: This paper is organised as follows. Section 2, explores preliminary concepts. The problem of zero-shot ZdA detection is then formalised in Section 3. Successively, the main contributions of the HERO framework are presented in Section 4. The experiments and results are presented in Section 5, while Section 6 discusses related works. Finally, conclusive remarks and future work direction are given in Section 7. All the abbreviations used in this work are enlisted in Table 1.

2. Preliminaries

(a) Collective Anomaly Detection: Canonical anomaly-based intrusion detection algorithms model a binary classification task where each sample is deemed normal or anomalous. However, anomalous observations may come from a heterogeneous set of unknown classes of traffic. In this respect, the focus of any CAD algorithm is to group anomalous observations into similar groups [14].[6]

In the context of NID, CAD can help efficiently manage anomalous traffic flows by associating automatic blocking or accepting rules to clusters as opposed to per-flow monitoring [22]. Moreover, for any deep learning pipeline, CAD is less prone to overfitting with respect to anomaly-based NID because, rather than polarising the latent space into two zones, it separates each cluster in the manifold [10].

(b) Metric-based meta-learning:

Detecting ZdAs by learning their distribution is impossible for any statistical learning instrument because ZdAs are not among training data by definition. In other words, ZdAs correspond to *out-of-distribution* (OOD) [23,24] samples.[7] In this respect, instead of learning the distribution of each class in the training set, metric-based meta-learning (Mb-ML) focuses on learning the distribution of intra-class *consistency* and inter-class *separation*, and thus be more amenable to generalise clustering over unseen classes [25]. In other words, Mb-ML learns a representational space whose *geometry is congruent* with the semantics of the inferences the model needs to output. In its basic form [26], metric-based meta-learning learns an optimal similarity ranking between unlabelled *-query-* samples and the class-specific centroids of a set of labelled *-support-* samples. The main benefit of this paradigm is that, by focusing on the relative similarities between queries and support samples, the inference results are decoupled from class-specific distributions, and can thus be portrayed to Few-Shot classification [27] or Zero-shot clustering of unknown classes, as done in HERO.

(c) Novelty-degree of Zero-day Attacks: Assume to have a two-level taxonomy of traffic classes, i.e., macro-categories that contain micro-categories. Then, anomalous traffic flows will correspond to samples of unknown micro-classes. In this context, a *type-A* anomaly is defined as an unknown pattern whose macro-class is also unknown, while a *type-B* anomaly is said to have a known macro-class [15]. The goal of this paper and related works on zero-shot ZdA detection is to correctly classify anomalies as type-A and type-B anomalies. This classification is assumed to give useful information about the nature (malign or benign) of anomalous observations at test-time [8,16]. Note that, from an epistemic point of view, type-A has a higher DoN with respect to type-B anomalies if the taxonomy is well defined.

Table 1
Abbreviations used in this paper.

Abbreviation	Meaning
ANN	Artificial Neural Network
AUC	Area Under the Curve
CAD	Collective Anomaly Detection
CE	Cross-Entropy
DDoS	Distributed-Denial-of-Service
DoN	Degree of novelty
DoS	Denial-of-Service
DL	Deep Learning
GAT	Graph Attention Network
GNN	Graph Neural Network
HPO	Hyper-Parameter Optimisation
IoT	Internet of Things
Mb-ML	Metric-based meta-learning
MITM	Man-in-the-middle
ML	Machine Learning
NAR	Neural Algorithmic Reasoning
NID	Network Intrusion Detection
OOD	Out of Distribution
OS	Operating System
PCA	Principle-component Analysis
PR	Precision-Recall
RoC	Receiver Operating Characteristic
TNP	True Negative-Proportion
TPP	True Positive-Proportion
XSS	Cross-site scripting
ZdA	Zero-day Attacks

[5] The HERO source-code, neural modules, and experiment-replay scripts are available at https://github.com/QwertyJacob/HERO .

[6] Many techniques exist for CAD. In this paper, a DL-based solution is used to perform clustering in linear time with respect to the number of input flows.

[7] In probabilistic modelling terms, ZdAs must be anomalies generated from a different distribution with respect to the distribution that generates training data anomalies.

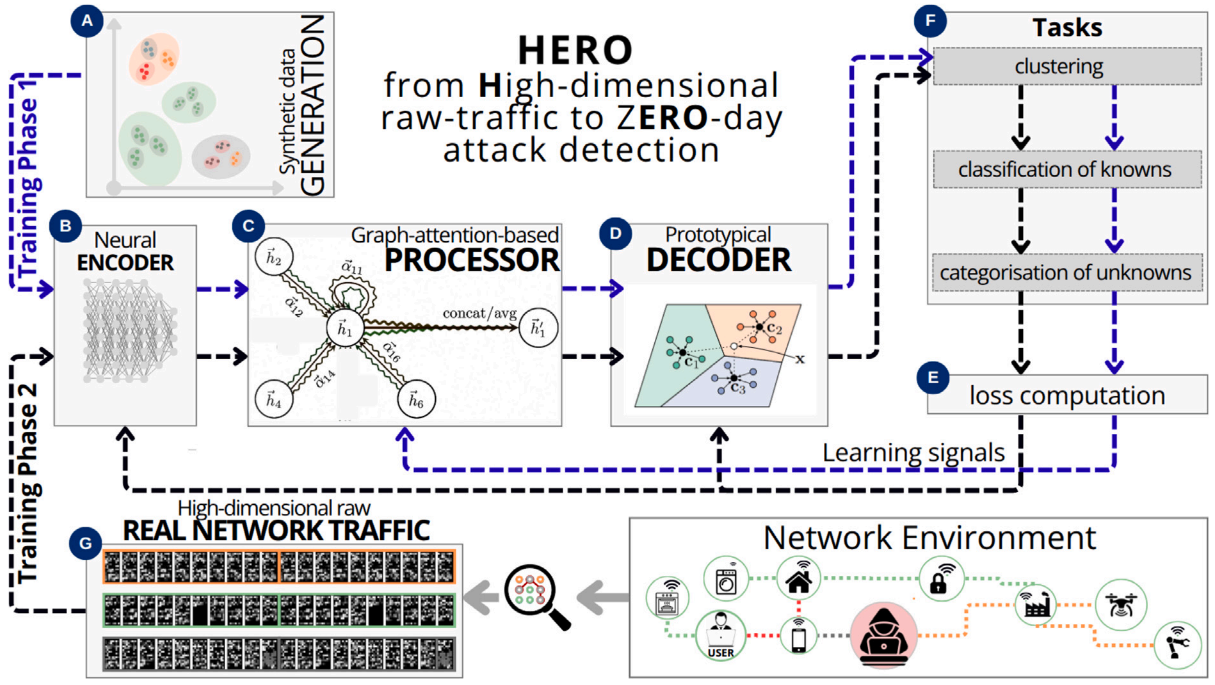

HERO
from High-dimensional raw-traffic to ZERO-day attack detection
A
Synthetic data GENERATION
Training Phase 1
B
Neural ENCODER
C
Graph-attention-based PROCESSOR
concat/avg
D
Prototypical DECODER
F
Tasks
clustering
classification of knowns
categorisation of unknowns
E
loss computation
Learning signals
Training Phase 2
G
High-dimensional raw REAL NETWORK TRAFFIC
Network Environment
USER

Fig. 1. Schematic representation of the HERO pipeline for ZdA-detection over high-dimensional raw-network traffic. The blue flow indicates the processor's pre-training on the algorithmic task using synthetic data. Once pre-trained, the processor weights are frozen, and suitable encoder and decoders are trained (red flow) using such a processor to converge to known-attack classification and collective anomaly detection and categorisation over high-dimensional data. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(d) Neural Algorithmic Reasoning:

The vector transformations over most DL-based classification pipelines can be conceptually decomposed into an automatic feature-engineering or *representation learning* task and a goal-oriented or *discriminative* task [28]. The first consists in transforming the feature space into a latent space or *manifold* while the second is more focused on establishing decision boundaries over such a manifold.[8] In this respect, as the number of neural parameters and the complexity of the discriminative task grows, the gradients of the discriminative learning signal could potentially be in conflict or not *synchronised* with the manifold learning one, potentially hindering the overall convergence [28].

In lots of OOD-sample classification tasks, such as the one modelled in this paper, the discriminative goal's complexity can be handled by metric-based meta-learning. However, besides the complexity of the discriminative task, another important factor that hinders the convergence of DL pipelines is the dimensionality of the feature space and the noise in training data, which are the precise context factors faced when processing raw network traffic.

In this respect, the neural algorithmic reasoning, is a DL framework that *decouples* the representation learning gradients from the discriminative ones by creating different training phases and neural modules with different goals: a NAR pipeline is generally composed of a stacked *encoder*, *processor* and *decoder* neural networks that are trained separately [17]. First, the processor network is trained to align to the discriminative task using low-dimensional de-noised data. Successively, the processor's weights are frozen, and the encoder and decoder networks are trained to map high-dimensional noisy data to a manifold aligned with the processor's operations and the processor's outputs to the output space, respectively.

The NAR paradigm often-times creates specialised labels, namely, hint-labelling strategies, to train the processor to align to the discriminative goal.[9] However, the sole application of such a labelling scheme is not enough for convergence when using high-dimensional and noisy raw-network traffic samples, which is the main goal of HERO. For this reason, also the two-stage training paradigm in NAR is applied in this paper. Namely, a low-dimensional synthetic dataset is created to train a processor network on the envisioned task. Subsequently, an encoder and decoder are respectively trained to learn the best-accommodating manifold and output mapping transformations for high-dimensional noisy inputs to be correctly processed by the pre-trained processor. The following section explains such a design choice in detail.

3. Problem and curricula formalisation

This section and the following explain in detail the problem formalisation, architectural, and training inductive biases of the HERO framework. To this end, these explanations follow the schemes in Figs. 1 and 2.

(a) Modelling the ZdA-detection task as an OOD hierarchical-CAD task:

The degree-of-novelty criterion is assumed to be useful information for experts to infer the benign or malign nature of anomalous traffic flows. For this reason, the main focus in HERO is the DoN categorisation of anomalies and every anomalous cluster will be referred to as a (potential) attack in the rest of this paper without loss of generality,[10].

To resemble the presence of ZdAs at test-time, the goal of HERO is to categorise anomalies by their DoN even if these are OOD with respect

[8] Note that, during training, the error gradients of any end-to-end differentiable DL-pipeline *carry* the signal for the parameters of the neural network to accommodate both the representation learning and the discriminative tasks.

[9] Originally, the NAR paradigm has been applied to teach processor networks to perform multi-step algorithmic tasks like sorting and demonstrate its learning over OOD inputs at test-time [29]. To resemble a specific algorithm, each step has its own label in classical NAR. NERO instead, creates a multi-label scheme to perform hierarchical CAD of OOD classes [9] over *low-dimensional* traffic-flow statistics.

[10] In other words, every anomaly is referred as an attack *a-priori* and the benign or malign nature of anomalies is supposed to be more easily inferred *a posteriori*. Namely, when looking at the degree of novelty of anomalies.

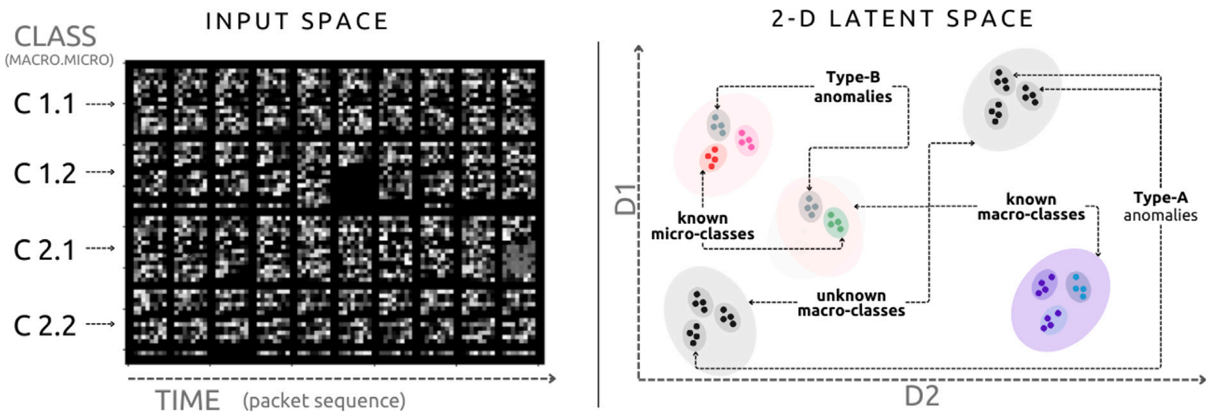


Fig. 2. Left: Low-dimensional visualisation of the raw-traffic captures as ingested by the HERO pipeline. Each traffic capture has a (known or unknown) macro-class and micro-class. The first n bytes of the first n packets of each flow are converted to gray-scale pixels to form a $n \times n$ image that represents the correspondent network observations. The sender and receiver IP addresses and ports are masked. **Right:** An ideal 2D representational space of input samples. Anomalies appertain to unknown micro-classes. Type-A anomalies (black dots) will appertain also to an unknown macro-class, while the macro-class of type-B anomalies (grey dots) is supposed to be known. (Known classes instead correspond to the coloured points in the plot). HERO performs hierarchical clustering of anomalies and classifies them as type-A or B without having seen any sample of a subset of classes during training, and without having their corresponding labels at test-time (i.e., classifies OOD samples on a zero-shot basis). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

to the training-time anomalies [10]. Overall, the ZdA-detection task will be modelled as a hierarchical and collective anomaly-detection task over out-of-distribution anomalies. Namely, apart of performing known-attack detection and anomaly detection over in-distribution samples, OOD anomalous traffic flows will be clustered into similar groups, and these clusters will extend a pre-existent and incomplete two-level taxonomy of traffic classes. By doing so, HERO will differentiate ZdAs by their DoN, i.e., by classifying them as type-A or type-B ZdAs. The ideal latent or representational space over which HERO performs such a task is represented in the right panel of Fig. 2, in which the hierarchical nature of the taxonomy – and the clustering task – is more evident.

Note that, apart from clustering and categorising anomalies, the samples of known classes will also be classified as such by HERO. Overall, the two tasks are performed by the framework, as schematised in panel **F** of Fig. 1.:

1. Hierarchical classification of known traffic.
2. Hierarchical clustering and categorisation of anomalies.

To compute a useful supervisory signal and perform gradient-based updating, standard binary and multi-class cross-entropy losses are computed both for the known-class classification and anomaly detection tasks. Also, the clustering loss, which is an adjacency or kernel regression loss based on metric-based meta-learning, can be seen as an additional regularisation loss signal based on cross-entropy. More details on this loss function are given in Appendix A.3. The loss computation process is represented by panel **E** in Fig. 1.

(b) Train–test split for resembling OOD samples:

To prove OOD ZdA detection capabilities, HERO should be evaluated on the classification of type-A and type-B ZdA samples from classes that were not seen during the training. However, there is a subtle difference between type-A and type-B ZdAs in terms of train–test-split. Namely, the macro-classes of type-Bs *must* be included in the training data, while those of type-As must not [9]. Formally, let $\mathcal{A}$, $\mathcal{B}$ C_{micro} and C_{macro} be the set of type-A ZdAs, type-B ZdAs, micro-classes, and macro-classes, respectively. Two conditions will need to be met to prove OOD ZdA detection:

- For every type-B ZdA, $\beta_i \in \mathcal{B}$, there must be at least one known micro-class $c_j \in C_{micro} \neq \beta_i$ in the training set whose macro-class is the same of β_i.
- For every type-A ZdA $\alpha_j \in \mathcal{A}$ all its micro-classes $c_k \in \alpha_j$ where $c_k \in C_{micro}$, must be kept exclusively in the test split.

The right panel of Fig. 2 shows the difference between type-A and type-B ZdAs from the perspective of a semantically aligned 2-D representational space. To build a training dataset for HERO, a multi-class two-level taxonomy of attacks and their correspondent traffic captures is needed. From such data, one can choose any desired subset of micro-classes to be masked as ZdAs, provided that the conditions above mentioned are met.

(c) Hint-labelling for ZdA detection:

Having done any train-validation-split that respects these conditions, a neural processor can be trained to identify type-A and type-B ZdAs. Following [9], such a task is decomposed into two separate classification steps or tasks, each one with its own goal: a first task classifies traffic flows as anomalous or known, while the second classifies the anomalies as type-A or type-B. Additionally, for both the train and test datasets, two different task-specific label sets or *hint-labels* are created to drive the learning of each one of the two mentioned tasks:

1. A first set of labels is related to micro-classes. In this respect, every ZdA sample is labelled as such, without specifying if these are either type-A or type-B ZdAs.
2. Macro-classes are the second label for each sample. Importantly, type-B ZdAs will be labelled with the corresponding macro-class label, while type-A ZdAs will keep the ZdA label.

Note that the first set of labels encodes a supervisory signal for optimising/evaluating the CAD task, while the second set guides the learning and evaluation of the DoN categorisation task. The central assumption backing up the effectiveness of this labelling strategy is the following: if an input sample is classified as appertaining to an *unknown* micro-class, it may be related to an unknown traffic class. Additionally, if this sample is associated with a known macro-class, it plausibly means that it is a type-B or less-novel anomaly. Instead, if it is associated with an *unknown* macro-class at the second step, it potentially represents a type-A or more-novel anomaly.[11] The next section describes specifically how to leverage the zero-shot ZdA-detection over high-dimensional raw-network traffic, which constitutes the main contribution of this paper in terms of pragmatic DL-inductive biases.

[11] The DoN categorisation in HERO is assumed to be *useful* and by no means *sufficient* for inferring the benign or malign nature of each anomalous traffic cluster. The automatic distillation of security-related semantic insights from raw traffic patterns would arguably require a much larger world-model or other forms of non-parametric knowledge and is out of the scope of HERO.

4. The HERO framework

4.1. Input space

(a) **Raw-network traffic format:**

The assumption is made of having a traffic capture module on an edge forwarder node, and the ability to perform traffic monitoring with a connection-wise granularity. In other words, each traffic capture is relative to a transport-level flow.[12] In the experiments, the byte sequences in raw-flow traffic captures were converted into greyscale images to be more efficiently processed by a neural network. Specifically, the *PcapPlusPlus* library [30] was used to split traffic into connection flows. Successively, the *heiFIP* library [31] was used to encode the first 512 bytes of the first 512 packets of each flow as bi-dimensional 512x512 images.

Following a common trend in deep learning for network analysis [32], the raw bytes of network packets are treated as numerical data, which can be directly encoded into a matrix and given in input to a deep learning model. Specifically, each byte of traffic-related information is an 8-bit value converted to an integer ranging from 0 to 255 using unsigned encoding. These integers are then converted to floats by dividing them by 255. We extract the first 512 bytes of the first 512 packets of each network flow to extract a 512x512 matrix of floats. The float numbers in these matrices are mapped to the intensities of grayscale pixels on an image saved on disk. During training, the images are loaded from the disk, converted again to numbers, and processed as bidimensional matrices by the neural modules.

In summary, each flow is represented by 512x512 matrices where each row of the matrix corresponds to one packet of such a flow. If a flow/packet has less than 512 packets/bytes, padding is made to keep the desired dimensions. Analogously, truncation is made to uniformly encode flows/packets with more than 512 packets/bytes. Sender and receiver IP addresses and ports were masked in every packet to avoid biasing the automatic feature extraction of the pipeline. Besides masking the sender and receiver addresses and ports, note the packet bytes contain the full header information and a significant quantity of payload. For example, TCP-IP packets have 40 bytes of header plus (a maximum of) 472 Payload bytes, while UDP-IP packets have 28 header bytes and 584 Payload bytes. Packets with less than 512 bytes and flows with less than 512 packets are padded to fit the input-space dimensions. More details on the nature of raw-traffic captures are available in Appendix A.2. The traffic capturing and preprocessing step is schematised in the **G** panel of Fig. 1.

(b) **On alternative pre-processing schemes:** The blueprint of HERO seeks to accentuate the "raw" characteristic of the input space. Namely, this paper advocated for a *zero-preprocessing* framework that avoids the computation of flow and packet statistics such as the mean inter-arrival times of payload lengths. This is the main reason why HERO takes raw-packet bytes of each packet on a flow and appends them as rows of a matrix, which forms the flow image. Note that the specific dimension of the input space (512x512) was chosen to be high-dimensional on-purpose,[13] while at the same time being compatible with different modest-sized open-weights pre-trained convolutional feature extractors.

However, letting apart the aim of avoiding the extraction of statistics from traffic to build compressed representations of the observation space, other preprocessing frameworks are worth to be tested in future works as input pre-processing mechanisms in HERO. Alternative schemes in pre-processing such as Flowpic [33], could enhance the resultant class separability in the encoder's input space, apart from reducing memory consumption. Note also that, if available on a specific scenario, low-latency statistics obtained by, e.g. NetFlow [34] or NFStream [20], if periodically refined and fed to the neural modules, equip HERO with near-real-time inference capabilities.

(c) **Synthetic data specification:** As mentioned in Section 2, performing zero-shot ZdA detection over raw-network traffic traces is challenging for an end-to-end DL pipeline. For this reason, HERO uses the NAR paradigm to decouple the representation learning and discriminative learning goals associated with such a task. Note that the former can be associated with automatic feature extraction over high-dimensional and noisy traffic bytes. Instead, as explained in Section 3, the discriminative task, in this case, coincides with the hierarchical collective anomaly detection of any set of input observations. To train the processor network to perform such a task, a set of low-dimensional synthetic data is generated, as schematised in the **A** panel of Fig. 1.

More specifically, pseudorandom real-valued vectors are generated from isotropic multivariate Gaussians located in random regions of the unitary 15-dimensional hypercube. Twelve macro-classes were generated for the experiments in this paper, each one containing five micro-classes. These data are then split into train and test samples, and specialised labels are generated using the procedure described in Section 3. More details on the synthetic-data generation procedure are available in Appendix A.2.

(d) **The attack semantic-graph:** HERO maps the information carried by real or synthetic traffic-flow *observations* on a heterogeneous *semantic graph* denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{macro}, \mathcal{E}^{micro})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$ is the set of vertices and the generic set $\mathcal{E} = \{e_{i,j}\}, \ \forall i, j \in \mathcal{V}$ is the set of macro/micro edges formed as follows. Each node or vertex in $\mathcal{V}$ corresponds to a unique input datum or sample, i.e., to a single transport-level flow. Two nodes $v_i, v_j \in \mathcal{V}$ will be connected by a macro edge $e^{macro}_{i,j} \in \mathcal{E}^{macro}$ if they share the same macro-label. In other words, two flows will be connected by a macro-link in the semantic graph if they appertain to the same macro-class of attack. Analogously, a micro-edge $e^{micro}_{i,j} \in \mathcal{E}^{micro}$ will exist between v_i, v_j if they are both related to the same attack micro-class.

Finally, The macro/micro *neighbourhood* of v is denoted with $\mathcal{N}(v)$ and corresponds to the set of nodes that are adjacent to v through the edges of the graph. Note that, in terms of *graph representation learning*, the *feature vector* of a node $v_i \in \mathcal{V}$ will be a high-dimensional tensor of raw-traffic bytes and will be denoted as $\mathbf{x}_i$ in the following. In synthesis, HERO will receive in input an incomplete or partial graph where nodes correspond to traffic flows and links correspond to same-class associations. Notice that only a subset of nodes will be labelled and, thus, connected between them through the edges of the graph. The goal of HERO is to perform *adjacency-regression* of this graph, i.e., to infer the classes of the unlabelled nodes. More specifically, to state if an unlabelled flow corresponds to a known class, or, eventually, a type-A, or type-B zero-day attack.

4.2. Neural models

This paragraph explains the architectural inductive biases that form the HERO pipeline. Specifically, the encoder, processor and decoder networks with its main learning goals. Additional technical details such as regularisation techniques are instead available in Appendix A.

(a) **Processor network:**

Any clustering task, as the one in Section 3, can be seen as a permutation-invariant set-to-set function that assigns a cluster to each input sample. In this respect, note that any DL-based clustering algorithm could be coupled from a graph modelisation of the input space like the one in Section 4.1. In this paper, the clustering algorithm used is based on Graph Neural Networks (GNN) [35], as these prove to be efficient neural approximators of permutation-invariant functions [36].

GNNs operate on a graph-format input batch through parametric combinations arising from the topology of the input graph [35]. Specifically, suppose $\mathbf{z}_u \in \mathbb{R}^m$ is the latent representation of a node v_u. In that

[12] A transport-level flow is the stream of packets associated with a 5-tuple made of a pair of IP addresses, a pair of transport-layer ports, and one transport-layer protocol.

[13] Stress tests on higher resolutions are left as future works

case, a GNN will map it to a representation $h_u \in \mathbb{R}^m$, as a function of the latent representation vectors of the neighbours of u. In its simplest form [37], one can express such an operation as follows:

$$h_u = \sigma\Big(\sum_{v\in\mathcal{N}_u} \mathbf{W}\mathbf{z}_v\Big) \tag{1}$$

where σ is a non-linearity, $\mathbf{W} \in \mathbb{R}^{m,m}$ is a learnable parameter matrix, and $\mathcal{N}_u$ is either the macro or micro neighbourhood of u, depending on the current focus on the current representation learning task.

Moreover, set-to-set functions can be approximated by attention mechanisms, and the Graph Attention Networks (GAT) [38] can be used to implement an attention mechanism between node sets. Specifically, GAT sophisticates the message passing in (1) with a neural attention mechanism that assigns different weights to the neighbours of v_u, as a function of their feature vectors:

$$h_u = \sum_{v\in\mathcal{N}_u} \mathrm{softmax}_{\mathcal{N}_v}(\sigma(\boldsymbol{\alpha}^T \cdot [\mathbf{W}\mathbf{z}_v || \mathbf{W}\mathbf{z}_u])) \cdot \mathbf{W}\mathbf{z}_v \tag{2}$$

where σ is typically implemented as a *Leaky Rectified Linear Unit* [39], and $\boldsymbol{\alpha}^T \in \mathbb{R}^{2m}$ is a learnable parameter vector.

The processor network in HERO is implemented using a more recent variant of the GAT architecture, namely, the GAT-v2 [40], where the multiplication with a smaller learnable attention layer $\boldsymbol{\alpha}^T \in \mathbb{R}^m$ is performed after the application of the non-linearity, permitting to decouple the domain of $\boldsymbol{\alpha}$ and $\mathbf{W}$, which enables the attention scores to vary as a function of $\mathbf{z}_u$:

$$h_u = \sum_{v\in\mathcal{N}_u} \mathrm{softmax}_{\mathcal{N}_v}(\boldsymbol{\alpha}^T \sigma(\mathbf{W} \cdot [\mathbf{z}_v + \mathbf{z}_u])) \cdot \mathbf{W}\mathbf{z}_v \tag{3}$$

More specifically, the processor network in HERO uses two stacked GAT-v2 modules to perform hierarchical clustering: The first module operated on the micro-neighbourhoods defined by $\mathcal{E}^{micro}$ classifies inputs between those related to known and unknown micro-classes, while the second module takes only the samples deemed as unknown by the previous module to discriminate between type-B or type-A zero-day attacks (the second processor layer will use the macro-neighbourhoods defined by $\mathcal{E}^{macro}$). At each forward pass, the processor modules use the available set of correspondent support labels (micro-classes or macro-classes) to initialise the neighbourhoods of elements in the graph. Importantly, ZdA observations are initially set as connected to all the other nodes. In summary, the processor module is represented by panel **C** in Fig. 1 and is made of two stacked layers, each one of these implementing (3). The goal of the first training phase is to predict the real connections between nodes in the graph using the attention weights in $\boldsymbol{\alpha}$.

(b) **Encoder network:**

The observed network traffic corresponds to the inputs of the encoder network, which is represented by panel **B** in Fig. 1 and performs the mapping from the input-space to the latent-space that the processor network operates over. Such a mapping can be denoted as:

$$\mathrm{Enc}^{\text{traffic}} : \mathbf{x}_i \rightarrow \mathbf{z}_i \in \mathbb{R}^m, \forall \mathbf{x}_i \in \mathbb{R}^d$$

Where $\mathbf{x}_i \in \mathbb{R}^d$ is any traffic observation from the input space, and $z_i \in \mathbb{R}^m$ corresponds to its latent-space encoding. Instead, d and m are the dimensions of the input and latent-spaces, respectively.

(c) **The NAR training framework and the different encoder architectures:** Following the NAR training framework in [17], HERO performs a two-stage training procedure for the whole neural pipeline: in the first phase, the processor network is to perform out-of-distribution hierarchical-CAD over synthetic data. For this task, an auxiliary encoder and decoder network are used to accommodate the shape of synthetic data. Successively, the processor's weights are frozen, and the main encoder and decoder networks are attached to the frozen processor to perform representation learning over real inputs.

The *auxiliary* encoder used for processor training was implemented as a multi-layer perceptron in our experiments. More details on these auxiliary encoders are provided in Appendix C.2. After the processor is trained, convolutional image encoders such as *ResNet-18* [41] or *MobileNetV3-small* [42] are used to transform raw-traffic captures to the processor's input space. Refer to Appendix B.2 for more detail on the encoder architectures.

(d) **Decoder:**

The decoder network transforms the processor's outputs to a representation space over which the error signal can be computed. As explained in Section 2, HERO uses metric-based meta-learning to decouple the hierarchical clustering from the specific distributions of training classes.

More specifically, the HERO decoders are Prototypical Networks [26], which operate as follows: At each training step, a set of encoded *support* inputs $\mathbf{z}_1^s, \mathbf{z}_2^s, \ldots, \mathbf{z}_{|\mathcal{B}_S|}^s$ and *query* inputs $\mathbf{z}_1^q, \mathbf{z}_2^q, \ldots, \mathbf{z}_{|\mathcal{B}_Q|}^q$ from a given batch $\mathcal{B} = \{\mathcal{B}_S \cup \mathcal{B}_Q\}$ is given to the decoder network. The decoder computes class-wise centroids or *prototypes* using the support samples:

$$\mathbf{c}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{z}_j^s, \ \forall \mathbf{c}_i \ \in \mathscr{C}, \ \text{ s.t. } \ \mathbf{z}_j^s \in \mathcal{B}_S, \ \ \forall j \in \{i, \ldots, N_i\} \tag{4}$$

where N_i is the number of support samples in class i and $\mathscr{C}$ is the set of classes included in $\mathcal{B}$. Successively, a similarity score $\mathbf{s}_{ij}$ is computed between each query input $\mathbf{z}_j^q \in \mathcal{B}_Q$ and the prototype $\mathbf{c}_i \ \in \mathscr{C}$. These scores are then sent through a learnable linear layer with a Sigmoid non-linearity to output a novelty score $v_{i,j}$:

$$v_{i,j} = \sigma(\mathbf{D}^T \cdot \mathbf{s}_{ij} + \mathbf{b}) \tag{5}$$

where $\mathbf{D} \in \mathbb{R}^{|\mathscr{C}|}$ and $\mathbf{b} \in \mathbb{R}$ are learnable parameters. Two decoder networks are implemented in HERO, one for each processor layer. The first decoder will use $v_{i,j}$ to assess whether $\mathbf{z}_j^q$ is potentially related to known types of traffic and, in a positive case, associate it to the respective micro-class. Analogously, the second decoder will be used to classify unknown traffic as type-A or type-B anomalies. In the case of type-B's, the respective macro-class will be indicated. Decoders are indicated in panel **D** of Fig. 1. Instead, the loss function computation is indicated in panel **E** in the same figure. The specific loss functions used for each task (namely, hierarchical clustering, hierarchical anomaly detection, and multi-class classification of known traffic) are specified in Appendix A.

5. Experimental case study

This Section describes the experiments made to evidence the convenience of the HERO framework for zero-shot ZdA detection over raw-network traffic. Specifically, the data, validation metrics, results and correspondent discussion is presented. For additional technical details in the experimental setting, refer to Appendix A. Hyper-parameter optimisation and ablations are instead respectively reported in Appendices B and C.

5.1. Raw network data specification

The raw IoT traffic captures used in the experiments are extracted from the attack traces offered in the BoT-IoT dataset in [43] and in the Industrial IoT dataset (Edge-IIoTset) in [44]. These datasets were designed to enhance research and development in the area of network security, particularly focusing on botnet detection within Internet of Things (IoT) and Industrial-IoT environments. The attacks in these datasets were organised into a two-level taxonomy where the macro-classes included generic denial-of-service and distributed-denial-of-service attacks, generic malware, scan, injection attacks, among others. Micro-classes were more specific, differentiating between protocols and specific malware or attack types.

To test the type-A and type-B ZdA detection, a two-level taxonomy of attacks was necessary which permitted to create a train–test split as specified in Section 3. Joining data from the above-mentioned data sources, a comprehensive dataset was made, which included a total of 7 macro-classes of attacks and 17 micro-classes. More specifically:

- 10 different micro-classes of attacks were extracted from the BoT-IoT dataset. These attacks were grouped to form 4 attack macro-classes: Denial-of-Service(DoS), Distributed-Denial-of-Service (DDoS), Scanning, and Data Theft attacks.
- 7 macro-classes of attacks were imported from the Edge-IIoTset and were grouped under 3 macro classes: Injection based-attacks, Man-in-the-middle (MITM), and other Malwares.

As specified in Section 4.1, an observation or data-point in the dataset corresponds to a (part of) a 5-tuple flow (a sequence of packets having a unique transport protocol, and the same sender and receiver ports and IP addresses). In this context, the training split of the dataset used in this paper had less than 17000 samples, i.e., no more than 100 samples per micro-class, while the evaluation dataset consisted of 30000 samples. More details on the specific micro-attack classes, the corresponding data volumes, and associated meta-labels are available in Appendix A.2.

Protocols: The mentioned datasets incorporate network traffic protocols like TCP, UDP, ARP, IPv6-ICMP, ICMP, IGMP, and RARP. The experimentation with more modern lightweight transport protocols such as QUIC is left among future works. The application-layer payloads tend instead to concentrate on HTTP and MQTT in the data sources used in this paper.

Please note that other transport layer protocols that use larger packet lengths (e.g., SCTP) or that enable the identification of a larger session length in terms of packets (e.g., QUIC and MPTCP) would be less compatible with a 512x512 fixed encoding of a flow. To this respect, future work directions could explore how to adapt the network to a variable input length. In the case of QUIC, for example, the initial packet (including crypto handshake, version negotiation and other header information) must be at least 1200 bytes [45]. In cases where systematic truncation of packets compromises the classification accuracy, Adaptive tensor shaping techniques, such as ragged tensors [46], in combination with multi-scale recurrent modules [47], could be used to build neural pipelines that manage inputs with heterogeneous sizes.

5.2. Evaluation metrics

For the multi-class classification of currently-known attacks, the **accuracy** measure defined in Eq. (6) is used, where the number of *total predictions* corresponds to the known attacks in each batch/epoch:

$$\text{Acc} = \frac{\text{Correct predictions}}{\text{Total predictions}} \tag{6}$$

For ZdA detection purposes, the **balanced accuracy** is used as defined in Eq. (7):

$$\hat{\text{Acc}} = \frac{\text{TNP} + \text{TPP}}{2} \tag{7}$$

Where TNP is the *true negative proportion* and is defined as the ratio of predicted negatives and the total number of negatives in the batch/episode:

$$\text{TNP} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{8}$$

Conversely, TPP is the *true positive proportion*, also called *recall*, *sensitivity*, in literature, and is defined analogously:

$$\text{TPP} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{9}$$

The reason for using (7) is that ZdAs – the *positive* samples in the open-set classification – represent an unbalanced class. Consequently, using (6) would result in a biased comparison.[14]

During training, the Area Under the Curve (AUC) of both the Receiver Operating Characteristic (ROC) curve and the Precision–Recall (PR) curve were tracked. These metrics provide insight into the model's performance by measuring the trade-off between sensitivity and specificity (via the ROC curve) and precision versus recall (via the PR curve). Tracking these metrics for each of the four tasks, – namely, micro and macro anomaly detection during training and testing – offers a detailed perspective on the width and effectiveness of the learned decision boundaries across training.

Table 2
Running evaluation-time metrics (max values).

Evaluation Metric	NERO	NERO_FT	ASAP	HERO
Type-B ZdA Bal. Acc.	0.500	0.951	0.950	**0.991**
Type-A ZdA Bal. Acc.	0.511	0.620	0.795	**0.932**
Micro-class Acc.	0.999	0.999	**1.000**	**1.000**
Macro-class Acc.	0.993	0.980	0.991	**0.996**
Type-B ZdA ROC-AUC	0.500	0.941	0.937	**0.990**
Type-B ZdA PR-AUC	0.762	0.970	0.949	**0.992**
Type-A ZdA ROC-AUC	0.759	0.866	0.931	**0.973**
Type-A ZdA PR-AUC	0.607	0.759	0.827	**0.918**
Type-B ZdA Acc.	0.524	0.934	0.944	**0.990**
Type-A ZdA Acc.	0.732	0.732	0.793	**0.940**

Note: All metrics are reported on the test set. ROC-AUC: Area Under the Receiver Operating Characteristic Curve; PR-AUC: Area Under the Precision–Recall Curve. Bal. Acc. refers to balanced accuracy as defined in (7), while Acc. refers to unbalanced accuracy as defined (7).

5.3. Baselines

The HERO framework is compared with three baselines. The first baseline is an adaptation of the training regime in the NERO framework [9], i.e., the whole encoder-processor-decoder pipeline is trained end-to-end with the real data. A further variation of this baseline consists in a two-step training where the whole pipeline is first trained using the same synthetic data generated in HERO, and then re-trained with real data using the new encoder and decoders. This baseline is referred to as NERO-FT from now on. The third and last baseline is an adaptation of the ASAP framework in [10] to the hierarchical regime of the problem modelled in this paper (the ASAP framework normally performs CAD over a single-level taxonomy of attacks). More specifically, the decoders of the ASAP framework are inherited in the architecture of HERO, and the two-phase training regime is adopted in this baseline.[15]

5.4. Results and discussion

The main results of the experiments are available in Fig. 3 and Table 2. Considering the train–test split described in Section 3, a periodic evaluation procedure was modelled to test HERO over OOD samples. More specifically, after each training epoch, an evaluation round was taken out where the data included OOD samples from type-A and type-B ZdAs. The multi-class classification accuracy as defined in (6) for micro and macro classes and the balanced binary classification accuracy in (7) for type-A and type-B ZdAs were tracked during the evaluation rounds and plotted in Fig. 3. Please note that these plots use a 20-step running average for ease of visualisation. All runs used an *early stopping* policy with a 40-epoch *patience*[16] where the stopping criterion corresponded to task of the most challenging task, i.e., the type-A ZdA detection accuracy metric.

[14] In cases of high unbalance, e.g. 90%/10%, A deterministic negative predictor could achieve better performance than any other baseline.

[15] Note that ASAP uses multi-modal inputs to classify attacks, and this relies on simpler inductive biases. Refer to [10] for more information about the neural architectures of the ASAP framework. In the experiments carried out, the ASAP baseline differs from HERO only for the decoder, which is a fixed (non-learnable) function. More specifically, the decoder uses a fixed threshold over the prototypical logits vector to discriminate anomalies from samples of known classes.

[16] https://en.wikipedia.org/wiki/Early_stopping

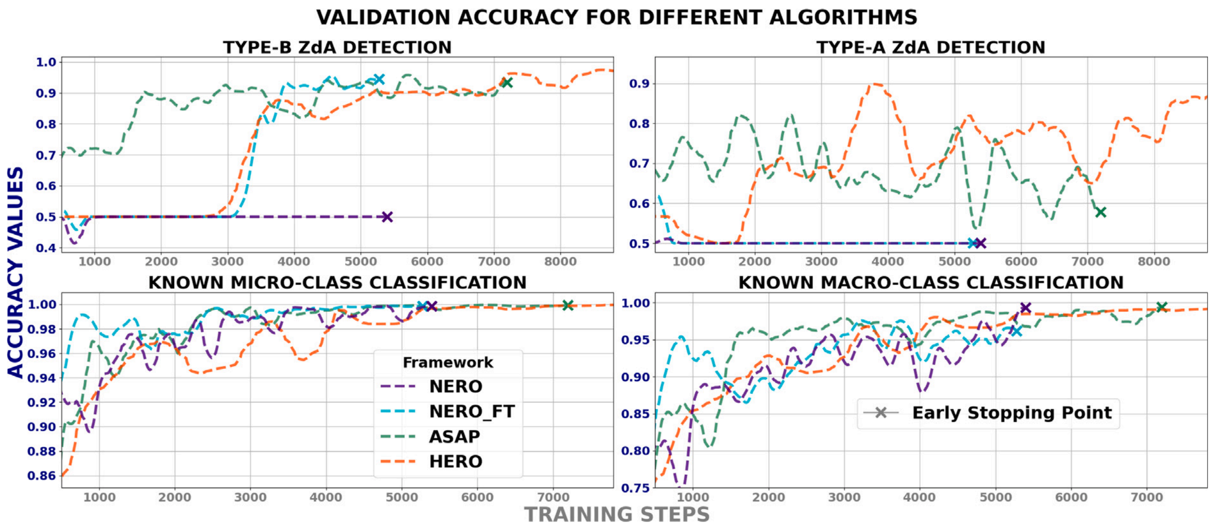


Fig. 3. Performance metrics in the experiments for HERO and various baselines. The first row presents the test-time balanced accuracy of clustering out-of-distribution attacks (first row) as type-a (right) or type-b (left) anomalies. The second row presents the multi-class classification accuracies for samples of known micro (left) and macro (right) attacks. These metrics correspond to online evaluation rounds, which took place after every training epoch. HERO is the unique algorithm that converges to acceptable values under the four tasks. Accuracies over training data are omitted because almost every baseline converges to acceptable performance over these data.

The upper line plots in Fig. 3 correspond to the type-B (left) and type-A (right) OOD ZdA detection task, where balanced accuracy as defined in (7) were registered during training for the whole evaluation dataset. Note that OOD ZdA detection is the most challenging task for the pipeline: in this task, samples of new classes of ZdAs are given in input, and the absolute position of the respective representations might be at any region of the latent space. In the worst case, the latent representations of OOD ZdAs could be even closer to representations of known attacks than what train-time ZdAs are.[17] In this respect, further experimentation varying the curricula in real data and the noise injection schemes over synthetic data might be necessary to precisely define the necessary conditions, in terms of input space noise, for the overall convergence of HERO. More details on these settings are available at Appendix A.2.

The lower plots in Fig. 3 refer instead to multi-class classification evaluation accuracy as defined in (6) for both micro (left) and macro (right) classes. These plots show that every baseline converges to acceptable multi-class classification accuracies for test data regarding known micro and macro classes: accuracies over training data correspond to *in-distribution* multi-class classification. Accuracy metrics over training data are omitted in the plots of Fig. 3 because almost every baseline converges to acceptable values for every task. Again, all the metrics reported correspond to the periodic evaluation of the baselines over the test-set or hold-out samples. The maximum values obtained across all the evaluation rounds are extrapolated and reported in Table 2. Note that such a table reports also the unbalanced binary accuracies for type-A and type-B ZdA detection as defined in (6).

The NERO and NERO-FT baselines strive to converge to acceptable ZdA detection accuracy for both type-B and type-A attacks. This difficulty in converging to ZdA detection may be a manifestation of overfitting the training data distributions. In other words, without using the two-phase training regime of the NAR paradigm, the pipelines might strive to learn a balanced combination of representation learning and hierarchical-CAD. These results confirm that differentiating the responsibility of each neural module helps the convergence of the whole pipeline on a high-dimensional and noisy input-data regime as that of raw-traffic traces.

On the other hand, among the algorithms that use a two-step training regime – namely, ASAP and HERO– only HERO converges to a high detection accuracy for both types of ZdAs: ASAP converges only for type-B ZdA detection. Note that this observation is also on the side of separating the module responsibilities. In fact, because the processor is frozen during the second training phase, when using the fixed decoders, ASAP puts more *representative pressure* in the gradients of the encoders' weights, which strive to converge to a perfect representational space as that in the right panel of Fig. 2.

However, note that the learning phases modelled in HERO impose a hard separation of module responsibility that might not necessarily be optimal in every high-dimensional scenario for other soft-separation schemes. In other words, while the NAR paradigm imposes a strict differentiation of responsibilities, the end-to-end training, as done in NERO and NERO-FT, can be seen as completely sharing the responsibilities between modules. The natural question that arises to this point is if a mid-term mixture of responsibilities, where the learning rates of different modules are diversified at each training phase, could result in better overall accuracies of the pipeline. The ablative studies in Appendix C answer such a question noting that some improvements are found in some learning rate configurations, but the gains might not be decisive in the presented use case.

Fig. 4 instead presents the evolution of the area under the curve for the Receiver Operating Characteristic (ROC) curve (left column plots) and Precision–Recall (PR) curve (right column plots). The former gives insight of the trade-off between sensitivity and specificity when detecting type-A (upper-left) and type-B (lower-left) anomalies, while the latter gives useful information about the trade-off between precision and recall for the same tasks (type-A ZdA detection corresponds to the upper-right plot, while the lower-right plot shows the evolution of this metric in the type-B ZdA detection task). It is worth stressing that every plot in Fig. 4 corresponds to the evaluation of various metrics computed periodically during the training procedure using the test-split.

Note how the AUC values for both the PR and the ROC curves reflect a more stable balance during training when detecting type-B anomalies (second row of plots) as opposed to type-A (first row). This behaviour could seem surprising, given the hierarchical nature of the task: type-A anomaly detection is a more coarse-grained task with respect to type-B detection, and thus, the latter should require a more precise definition of the classification boundary with respect to the former. However,

[17] Recall also that prototypical classification of OOD samples, i.e. Few-Shot learning, is impossible with such attacks because no support labels are given. Hence, the pipeline can just recognise that these representations belong to an unidentified cluster.

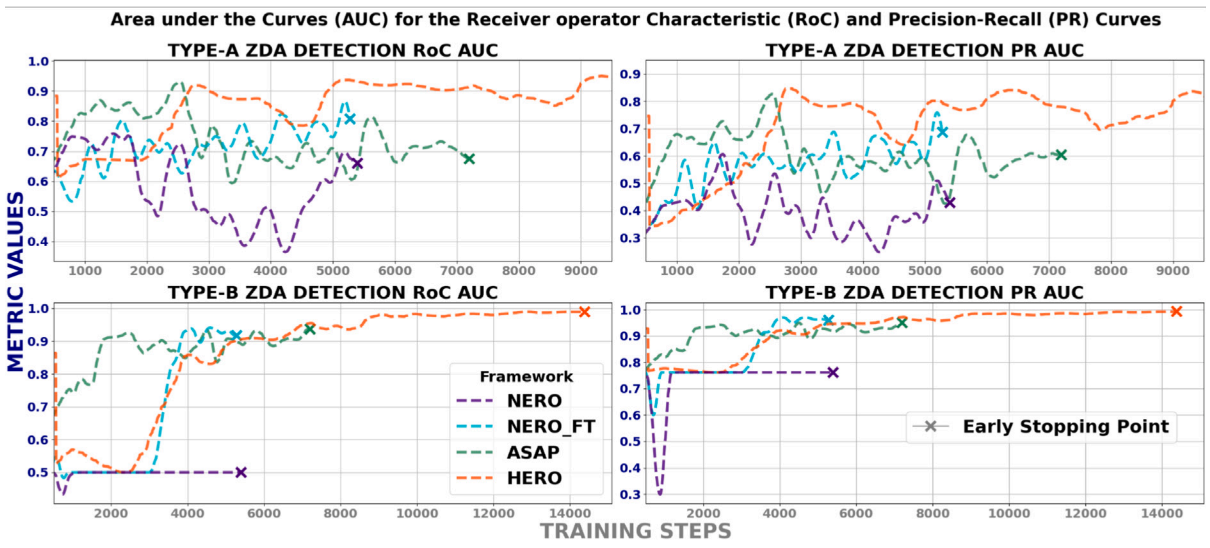


Fig. 4. Evolution of the Area Under the Curve (AUC) for the Receivers' Operator Characteristic (ROC) Curve and the Precision–Recall (PR) Curve for both the type-A and type-B ZdA detection during online evaluation rounds. Also here, early stopping was set to 40 evaluation rounds. Contrary to any other baseline, HERO converges to acceptable values for both tasks.

the stability in the lower row of plots of Fig. 4 could be associated to the **deceptive** behaviour of (most of) the baselines, where the type-B classification task converges but the type-A does not.

It is important to stress, in this respect, that while a *flat* anomaly detection task such as type-B ZdA detection is easier to be generalised to OOD anomalies, the hierarchical nature of the type-A ZdA detection task is far more challenging to generalise, because the combinatorial nature of the outcomes: micro-anomalies can be associated either with macro-anomalies or known macro-classes. Perhaps the initial instability observed in the RoC-AUC and PR-AUC plots for HERO in the type-B ZdA task (with respect to ASAP, for example) could in fact be associated to a *synchronisation* phase when correctly learning the type-B and type-A classification tasks. More details on the performance evaluation of the baselines over both the ZdA detection task are available in Appendix D. Additional metric reporting is available in the public *Wandb* workspace of the project,[18] while the code to reproduce the plots in this paper can be found in the public repository of HERO.

6. Related works

(a) **Neural Algorithmic Reasoning** Although the NAR blueprint has been recently formalised, other reasoning-oriented inductive biases have been applied in deep learning in the last three decades [48]. The seminal work in [49] formalises the notion of *relational inductive biases* in deep learning to boost the performance and generalisation capability of connectionist AI. Instead, the authors in [50] formally demonstrated that the neural message-passing framework in [51] could potentially align with learning the rationale of any dynamic-programming algorithm In this respect, in [52], several inductive biases that facilitate out-of-distribution generalisation of neural message-passing architectures are presented. In [53], instead, a generalisation for the neural message-passing operation is given, and evidence is presented about the potential alignment between this formal scheme and various forms of symmetry-aware neural architectures.

A successful and direct application of the NAR blueprint is offered in [54], where Deac et al. showed improved data efficiency concerning state-of-the-art deep reinforcement learning models in the ATARI benchmark. The authors of [55] instead taught an ANN to output candidate values for missing parameters in network configuration specifications.

The work in [56] proposed a benchmark dataset to assess the algorithmic reasoning capabilities of neural networks. A successive work in [29] identified many inductive biases that lead to the convergence of a unique GNN-based processor network that can solve many potentially different dynamic programs. Instead, many other neural algorithmic reasoning inductive biases are evidenced in [57], where the focus goes beyond dynamic programming and concentrates on the ability of GNNs to solve combinatorial problems. To the best of the authors' knowledge, the unique application of the labelling strategies akin to the NAR framework for ZdA detection was presented in [9], while HERO is the first work to propose the application of the two-step training scheme, which is a main ingredient of the full NAR blueprint.

(b) **Zero-day Attack detection:**

The authors of [8] relied on zero-shot learning to achieve zero-day attack detection. They relied on descriptions of both seen and unseen classes and a multi-modal learning paradigm in which an auto-encoder is trained to translate between the feature and semantic domains. Having tested their solution with the KDD-CUP 99 dataset [58], the authors hope to extend the benchmarking using more recent datasets.

To cope with the specific requirements of the IoT scenario, authors in [16] introduced a hierarchical model in which two anomaly-detection modules are considered. The first is a lightweight anomaly detection component that permits the benign traffic to bypass the second module, which operates open-set intrusion classification. Implementing a federated learning framework to train the attack classification module is among the future works evidenced by the authors.

In [59], an encoder–decoder paradigm is used to learn effective representations of known attack vectors into a regularised latent space. Without investigating type-A attacks, the authors in [59] used several device-specific slices of the N-BaIot [60] dataset to evaluate the convergence of the latent space to representations that permit shallow ML classifiers to differentiate between normal traffic and (type-B) unknown attack samples. As a future work, the authors of [60] intended to investigate how to extend the representational capacity of the latent space to more than one macro-category of attack samples, i.e., type-A attacks.

In [61], a two-step ZdA attack detection based on conditional and variational encodings is presented. In the first phase, a closed-set classification task is optimised in the latent space. The latent space is learned by using a conditional and variational encoding pipeline. In the second stage, the conditioned reconstructions of the learned latent variables are used to assess if an observation is an instance of

[18] https://wandb.ai/jfcevallos/HERO

an unknown attack. In the hierarchical generative paradigm adopted in [61] *extreme value theory* concepts are applied to make inferences about the potential novelty of observations, while this proposal uses the neural machinery to learn to extract the novelty degree automatically through a specialised labelling strategy.

The authors in [62] presented experiments that resembled a ZdA detection problem by excluding one class at a time from the training set of two ML-based NIDS. They used nearest-neighbour assignation to classify an out-of-distribution sample as an attack. Among the conclusions reached was the convenience of the Netflow formatted dataset [21] for augmenting the ZdA detection rate, and, unsurprisingly, the difficulty of recognising ZdAs whose influence in the distribution of the training set statistics would have been significant if those attacks had been among the training samples.

The authors in [8] designed another strategy to embed the features of the NSL-KDD [63] dataset in a more sparse dataset by training a Word2Vec [64] embedding model using a subset of 20 pre-selected features from this dataset. They used a sparse auto-encoder and the semantic auto-encoder paradigm to preserve some semantic or latent space regularities.

Prototypical decoding has also been used in recent works centred on ZdA detection [8,19,62]. Unlike these works, our proposal relies on two prototypical learning tasks mapped to identify ZdAs and the differentiation of type-A and type-B ZdAs. By doing so, a *distribution-agnostic* pipeline that hierarchically distills ZdAs aligns with the abstract reasoning behind ZdA detection as explained in .

Finally, compared to the above-referenced literature, our main contribution relies on adopting the two-stage NAR paradigm. More specifically, a synthetic pre-training task for the processor network was designed to reach data-efficient convergence of an OOD ZdA detector with high-dimensional raw-network captures. Note that, rather than referring to a distribution shift in the *cardinality* of the inputs [65], the OOD qualifier in this work refers to the distribution shift regarding the *content* of the training observations.

7. Conclusive remarks

This work focused on zero-day attack detection over high-dimensional raw-traffic traces. Using the neural algorithmic reasoning paradigm, HERO provides explicit guidelines on building a DL pipeline that follows an abstract reasoning-based supervisory signal to generalise ZdA attack detection and categorisation over unknown classes that are unseen during training. By providing a step-wise training framework for an *encode-process-decode* architecture [66], HERO reaches convergence by decoupling the representation learning task from the algorithmic learning task using a pre-trained neural algorithmically-aligned processor. At the same time, its neural machinery may potentially learn to implicitly mimic or even optimise deterministic algorithmic rules [65].

Future work directions have two main axes of development. The first involves experimenting HERO over online-learning and detection scenarios: although our experiments validate the effectiveness of our approach with real raw traffic captures from well-known intrusion detection benchmarks, patching the traffic into images at real-time could facilitate the usage of convolutional encoders in online-learning scenarios. A second road of research instead could augment the lightweightness of the neural modules in HERO by using quantisation [67] or logic gate networks [68].

CRediT authorship contribution statement

Jesús F. Cevallos M.: Writing – original draft, Validation, Methodology, Conceptualization. **Alessandra Rizzardi:** Writing – original draft, Validation, Supervision, Funding acquisition, Conceptualization. **Sabrina Sicari:** Writing – original draft, Supervision, Conceptualization. **Alberto Coen-Porisini:** Supervision, Funding acquisition.

Table A.3
Hyperparameter specifications.

Hyperparameter	Specification
Learning Rate (unless explicitly changed)	0.001
Support samples per class (K-SHOT)	5
Query samples per class	20
Classes per batch during processor training (N-WAY)	5
Classes per batch during enc./dec. training (N-WAY)	4
Total synthetic macro classes	12
Total synthetic micro classes	60
Total real macro classes	7
Total real micro classes	17
Dropout Rate (processor and encoders)	0.3
Hidden space dimension	128
Optimizer	Adam
Train/Evaluation split over known attacks	80%/20%
Train/Eval. tasks per episode (phase 1)	500/100
Train/Eval. Tasks per ep. (phase 2)	100/20

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the SERENA-IIoT project, which has been funded by MUR (*Ministero dell'Università e della Ricerca*) under the PRIN 2022 program (project code 2022CN4EBH), and in part by project SERICS (project code PE00000014), under the NRRP MUR program funded by the EU - NGEU.

Appendix A. Additional technical specification

A.1. Implementation details

All the code of this work is implemented in Python 3.11.9, and the neural modules are implemented using the *PyTorch* deep learning library, version 2.5.1. The convolutional encoders used are those available in the Torch-vision library, version 0.20.1. All the experiments were done using an NVIDIA A100-SXM4 GPU with 80 GB of RAM. The most important parameters and hyper-parameters used for the experiments are detailed in Table A.3. Refer to the public repository of the project for the code to reproduce all the experiments and the specification of additional hyper-parameters.

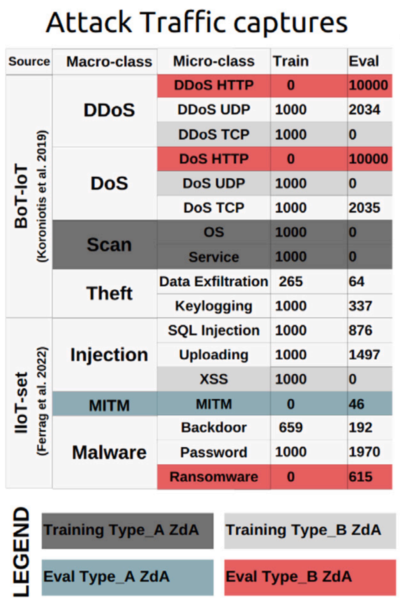
A.2. More details on data

This paragraph reports more specific information about the data used in this experiment. All the code for data generation, pre-processing, and all the pre-processed and generated data used in the experiments is available in the public repository of HERO.

(a) Synthetic data generation

To train the encoder network on a hierarchical-CAD task, pseudo-random real-valued vectors were generated from isotropic multivariate Gaussians. The centroids of macro-clusters were generated by first, then, the centroids of micro-clusters are generated by sampling random translations from the respective macro-centroids with a constrained variance. Finally, dimensional-wise Gaussian noise in injected in each sample where the noise variance is one order of magnitude bigger than that of the cluster-specific Gaussians.

Twelve macro-classes were generated for the experiments in this paper, each one containing five micro-classes. Specifically, a total of 4 macro-classes were masked as type-A ZdA. To avoid overfitting to



Attack Traffic captures

Source	Macro-class	Micro-class	Train	Eval
BoT-IoT (Koroniotis et al. 2019)	DDoS	DDoS HTTP	0	10000
		DDoS UDP	1000	2034
		DDoS TCP	1000	0
	DoS	DoS HTTP	0	10000
		DoS UDP	1000	0
		DoS TCP	1000	2035
	Scan	OS	1000	0
		Service	1000	0
	Theft	Data Exfiltration	265	64
		Keylogging	1000	337
IIoT-set (Ferrag et al. 2022)	Injection	SQL Injection	1000	876
		Uploading	1000	1497
		XSS	1000	0
	MITM	MITM	0	46
	Malware	Backdoor	659	192
		Password	1000	1970
		Ransomware	0	615

LEGEND

Training Type_A ZdA	Training Type_B ZdA
Eval Type_A ZdA	Eval Type_B ZdA

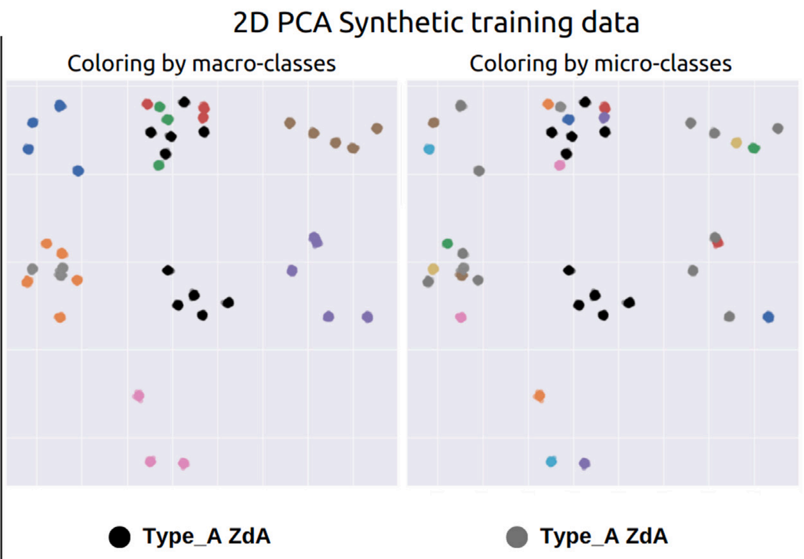


Fig. 5. Left: Train–test split made in the experiments. To prove generalisation over ZdA detection, evaluating the pipeline should involve assessing the classification of *new classes* of type-A and type-B ZdAs concerning those observed during the training. **Right:** Synthetic noisy observations are generated from random multi-dimensional Gaussian blobs. Some blobs are masked as type-A (black) and type-B (grey) ZdAs. type-B ZdAs appertain to known macro classes, while type-A ZdAs are attacks from an unknown macro-class. The known macro (left) and micro (right) classes use other colours instead. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

a unique distribution, type-A ZdAs were selected by choosing macro-clusters located in different regions of the resulting vector space. From the remaining macro-classes, twenty micro-classes were masked as type-B ZdAs, eleven were used in the training dataset, and nine in the test dataset. The dataset contains only five hundred samples for each class to resemble constrained data availability.

A two-dimensional Principal Component Analysis (PCA) decomposition of the synthetic training dataset is shown in the right panel of Fig. 5. In such a figure, type-A ZdAs in the left-side panel are black-coloured, while type-B ZdAs are indicated with grey points in the right-side panel. The known macro (left) and micro (right) classes use other colours instead.

(b) Real data

Raw network traffic associated with attack and benign traffic was obtained from the BoT-IoT [43] and Edge-IIoTset [44] datasets for the experiments in this paper as explained in Section 4.1. Detailed information on the train–test split and the per-class data volume are shown in the left panel of Fig. 5, where the meta-labels (i.e. ZdA labels for train and test) are colour-coded. Please note that, as explained in Section 3, a theoretically sound evaluation of type-A and type-B attack detection requires a subset of **classes** to be excluded completely from the training set. Specularly, it is not required that train-time **anomalies** (fake) type-A and type-B attacks used during training are included in the test set, and the evaluation split used in the experiments omitted these classes.

A note on encryption. Many existing approaches have demonstrated the effectiveness of neural networks, particularly convolutional neural networks (CNNs), in classifying encrypted traffic [69]. These methods often rely on raw byte-level features, flow statistics, or structural patterns in the encrypted data (e.g., packet sizes, timing, or header fields) to distinguish between different types of traffic [70]. Our approach follows this established paradigm by treating raw packet bytes as grayscale pixel values in a 512x512 matrix, enabling the CNN to learn from encrypted and unencrypted traffic without requiring decryption or explicit feature engineering [71].

Although not specified by the authors in [43], the flow traffic captures might contain encrypted traffic (e.g. MQTT related connections in the Edge-IIoTset or some post-exploitation communication in the Metasploit framework used in the BoT-IoT). If present, the bytes relative to encrypted payloads are represented as grey-scale pixels, as done with unencrypted data. Even if encryption removes semantic meaning, structural artifacts (e.g., packet size, timing, header fields, or entropy patterns) may persist in the byte stream and be detectable by the CNN [72]. Again, while encrypted payloads are randomised, protocol-specific headers or flow-level characteristics may still be captured in the first 512 packets, enabling classification based on non-payload features [73].

On multiple flow attacks. Please note that, from an epistemic point of view, the *instances* of some classes of attacks, e.g. distributed attacks, have a unique *causal root* that might generate multiple connections and flows. Apart from raising the need for delivering more mature threat intelligence insights, this context factor has two other side-effects: data imbalance (distributed attacks may be more represented with respect to sibyl or low-frequency ones), low representativeness (focusing on a specific class of attacks, it is better that the attacks flows used for learning are generated by more than one unique attacker).

As mentioned in Section 5.1, our paper deals with the former of the above-mentioned problems by modelling a low-data regime while inheriting the potential heterogeneity restriction resulting from the settings in [43,44]. In any case, our work focuses on inferring the *classes* of attacks by looking at individual transport-level flows. Future works could explore instead the approximation of set-to-set functions that associate (sets of) labels to sets of flows as a whole [74,75].

A.3. Kernel regularisation

Following the insights from the ASAP framework [10], HERO uses the *manifold-learning* inductive-bias [76,77] that helps the overall convergence of the pipeline. In manifold learning, the pipeline is encouraged to pull apart the hidden representations of nodes appertaining to different classes and to put the representations of same-class observations closer. Specifically, our training explicitly minimises two cross-entropy (CE) terms: first, the CE between the distributions of the baseline and predicted adjacencies, and second, the CE between the respective complementary distributions. The former CE term acts as an *attractive force* that pushes together latent space representations of the same class, while the second acts as a *repulsive force* analogously. The resultant effect is a clustering-friendly latent space, which, as shown in Section 5, helps to generalise prototypical classification out-of-distribution.

In terms of the conceptual graph, taking into account the availability of target labels for all the samples of the batch, a binary ground-truth adjacency matrix $\mathbf{A}^{|\mathscr{B}|\times|\mathscr{B}|}$ is computed that interconnects samples from the same class. The attention matrix of each processor layer is used instead as the adjacency predictions. The working principle of this regularisation mechanism is as follows. Given an input batch $\mathscr{B}$, the corresponding class one-hot labels $\mathbf{y}_i, \forall i \in |\mathscr{B}|$ are used to build a semantic adjacency kernel $\mathbf{A}$:

$$\mathbf{A} = \left[a_{ij}\right]_{i,j\in|\mathscr{B}|} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{|\mathscr{B}|} \end{bmatrix} \times \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_{|\mathscr{B}|} \end{bmatrix}^T$$

On the other hand, pair-wise similarities α_{ij} between the latent representations $\mathbf{z}_i, \mathbf{z}_j \forall i, j \in |\mathscr{B}|$ are computed. The similarity distributions are then used to minimise the divergence between the predicted and ground truth similarity and distance distributions using cross-entropy:

$$\mathscr{L}_{AR} = -\frac{1}{|\mathscr{B}|}\sum_{i=1}^{|\mathscr{B}|}\left(\boldsymbol{a}_i^T \cdot \log(\boldsymbol{\alpha}_i) + (1-\boldsymbol{a}_i)^T \cdot \log(1-\boldsymbol{\alpha}_i)\right)$$
$$\text{where } \boldsymbol{a}_i = \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ \vdots \\ a_{i|\mathscr{B}|} \end{bmatrix} \text{ and } \boldsymbol{\alpha}_i = \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \alpha_{i3} \\ \vdots \\ \alpha_{i|\mathscr{B}|} \end{bmatrix} \tag{A.1}$$

By minimising (A.1), the alignment between the geometric relations of samples in the semantic and latent-spaces is maximised. In other words, the model is biased towards learning the distribution of similarities between the environment observations, which is denoted as $\mathscr{D}$. Notice that $\mathscr{D}$ describes the regularities that permit to cluster elements in a way which is consistent with the semantic labels at hand. Note also that the existence and the learnability of $\mathscr{D}$ is a fundamental requirement to generalise collective anomaly detection over OOD observations.

A.4. Detaching micro and macro gradients

By restricting the closed-set gradient computation on query samples related to known classes, the encoder and processor are prevented from *pushing* the latent representations of query samples that correspond to unknown attacks to known class signatures, which would be incorrect. In addition, the gradients related to open-set loss are detached from the rest of the pipeline to avoid overfitting the latent representations to the ZdAs present in the training data. Another crucial aspect of meta-training is the random sampling of different classes and different support and query samples within each batch. Having a limited number of classes and a limited number of samples for each class, the training batches are sampled with replacement each time from the available data.

A.5. Additional hyper-parameter detail

When pre-training the processor network with synthetic data, the training episodes were made of 500 batches or tasks, while the evaluation episodes contained 50 batches. At each batch, a 5-way 5-shot classification is performed: 5 micro-classes are sampled from different macro-classes, one of which is a type-B ZdA and another being a type-A ZdA. For each sampled class, five support samples and 50 query samples are given, and this pipeline classifies the query samples both in the open and closed set tasks of each phase. The processor training reaches an accuracy of 100% on all four tasks after 50 epochs. Obviously, training on synthetic data is much faster than on high-dimensional images. For fine-tuning the pipeline using real traffic captures, the unique variations consisted of reducing the training tasks to 100, the evaluation tasks to 20, and the N-way of episodic learning to N = 4. For every 100 evaluation steps.

Appendix B. Hyper-parameter optimisation

The main axes of hyper-parameter tuning are reported in this Section. Refer to our public *Wandb* project dashboard[19] for more metric reporting over more than 50 experiments.

B.1. Kernel regularisation

Note that, to reach OOD clustering, the effectiveness of the metric-based meta-learning inductive bias described in Section 2 relies on a precise hypothesis: the distribution of the inter-cluster densities and intra-cluster separations generalises over all the classes (including the OOD ones) and is learnable. Under an ideal context, such hypothesis proofs are completely valid, and thus, OOD clustering is learnable with perfect accuracy. However, in a real scenario, such conditions may not be completely met. In other words, there could be a divergence between the distribution of cluster separations and densities when considering all the classes and the same distribution over the restricted set of train-time classes.

This divergence implies that a potential downside of the kernel regularisation loss described in Appendix A.3 may be the overfitting of the distribution of inter-cluster densities and intra-cluster separations of the training-time classes, which may hinder the accuracy of test time clustering. In this respect, multiple experiments were carried on multiplying the regularisation loss in (A.1) by a scaling factor ranging from 0 to 1. The best results were encountered using a scaling factor of 0.1. The main results for our hyper-parameter tuning of the kernel regularisation are plotted in the upper-left panel of Fig. 6.

B.2. Alternative neural architectures

(a) Encoders

After training the processor network and making it converge to OOD hierarchical-CAD, the processor is frozen an it has to be used as a deterministic block for the encoder and decoder networks to learn to de-noise and separate raw-network traffic traces into separate clusters on a latent space whose geometry is the same the processor ingested during its own training phase.

Notice this is an automatic feature engineering or representation learning task that the encoder is trained to perform over raw-traffic traces. Raw traffic is made of sequences of packets and can thus be well-processed by recurrent neural networks, set-to-set encoders like transformers, or dilated-enough convolutional modules. However, aiming to reduce the computational burden of the overall pipeline, convolutional networks with skip connections were used in this paper. Specifically, some of the more lightweight pre-trained modules available in the Torch Vision library were used for training the encoder, specifically some architectures of the *SqueezeNet* [78], *MobileNetV3* [42] and ResNet [41] were tested, having found the better results for the latter models.

(b) Decoders

Various decoding architectures were tested for the anomaly detection task in both levels of the hierarchical-CAD task. More specifically, having a set of logits or association scores regarding the known classes as specified in Section 4.2, the anomaly-detection task consists of

[19] https://wandb.ai/jfcevallos/HERO

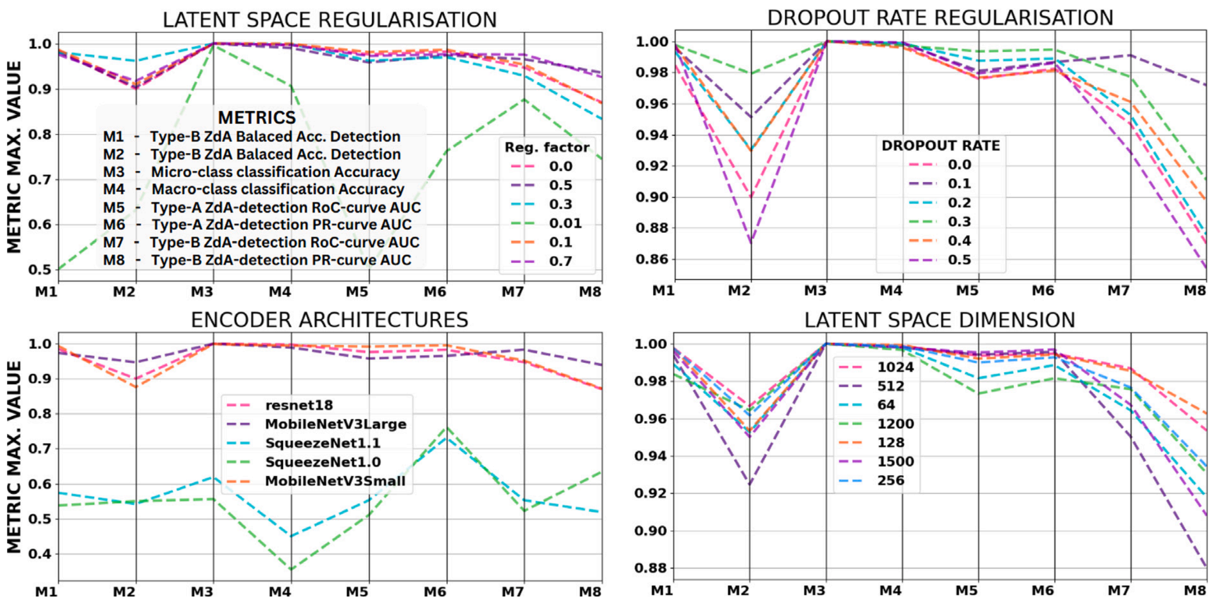


Fig. 6. Parallel coordinates plots for hyper-parameter tuning and various performance metrics. Upper-left: scaling factor of the latent space regularisation loss function in (A.1). Upper-right: dropout rate at after the last layer of the encoder. Lower-left: Encoder architectures. Lower-right: dimensionality of the latent space over which the processor network operates.

classifying a set of logits as *valid* or *invalid*, i.e., telling if there is a closer-enough association score among the set of logits.

Notice that the specific threshold of significance for any set of logits may depend on the extension and overall sparsity of the processor's latent space. For this reason, multiple architectures with increasing complexity were tested in the decoder modules. However, the best two alternatives were the *MinMaxDecoder* used in the ASAP framework [10] and the final architecture described in Section 4.2. The latter performed better than the latter, and the reason for this superiority might be the fact that the former is a static function with a fixed threshold while the second has a learnable transformation. Notice that a fixed threshold works in dynamical domains because it shifts the anomaly criteria learning to the rest of the pipeline.

B.3. Dropout

Several experiments were made to assess if the dropout mechanism could enhance the robustness of the encoder network in the second phase of training. specifically, a dropout layer that randomly masks a fraction p of neurons in the semi-last layer of the encoder was added with p ranging from 0 to 0.5 in the experiments. No significative benefit was noticed from the usage of such a dropout mechanism besides a slightly faster convergence of the AUC for the RoC and PR curves of the macro-level classification task during training. The main results of tuning the dropout ratio are summarised in the upper-right panel of Fig. 6. The injection of different levels of dropout mechanisms in the processor network during the first learning phase is left as a future work instead (the processor had a fixed dropout rate of $p = 0.1$ in all the experiments carried out).

B.4. Latent-space dimensionality

The dimension of the latent space or representation space over which the processor network operates was also subjected to tuning, as shown in the lower right panel of Fig. 6. Note that this dimension corresponds to the output-space dimension of the encoder and the input-space dimension of the decoder. For this reason, these tunings required both training phases to be executed in series. No decisive improvements were found by augmenting the dimension of the latent space above 128. In this respect, the evolution of 2D PCA decompositions of the latent spaces is available in the public *Wandb* dashboard of the HERO project.

Appendix C. Ablations

C.1. Re-training the processor

The canonical NAR framework works with two training phases: the first one trains the processor on a discriminative task, and the second trains the encoder and decoders on the representation learning and output-space accommodation or *unembedding* task. Usually, the weights of the processor network are frozen during the second stage of training. However, in cases where the discriminative task alignment follows a fixed algorithm that could be optimised by a *soft-alignment* or algorithmic relaxation induced by the high dimensionality and fuzziness of the processor's latent space.

For this reason, different experiments were made in which the processor's weights were kept as learnable during the second stage of training. The learning rates of the processor were generally one or two orders of magnitude minor than those of the encoder and decoder. This hybrid learning stage can be seen as a fine-tuning of an algorithmically aligned processor that can better accommodate the specific use case at hand. In our experiment, there was not a decisive increment of performance in terms of accuracy when using this fine-tuning, however, as can be seen in the right panel of Fig. 7, a slightly better AUCs for both the RoC and PR curves in ZdA detection tasks was obtained with a hybrid fine-tuning of the processor where it had a learning rate three orders of magnitude lower than that of the encoders and decoders.

C.2. Diversification of learning rates

Note that perfectly decoupling the representation-learning and discriminative tasks might not always correspond to optimal performance on a high-dimensional and noisy input space as the one targeted by HERO. In this respect, similarly to what has been done in the previous paragraph, one can ask if a bespoke learning-rate differentiation during the first training phase could result in the overall enhancement of the pipeline's performance at test time.

To this end, several configurations of the first phase were made, and the learning rates of the encoder and decoder networks were reduced. Note that such a procedure can be conceived in two different forms: first, reducing the learning rate in the encoder could be seen as a noise

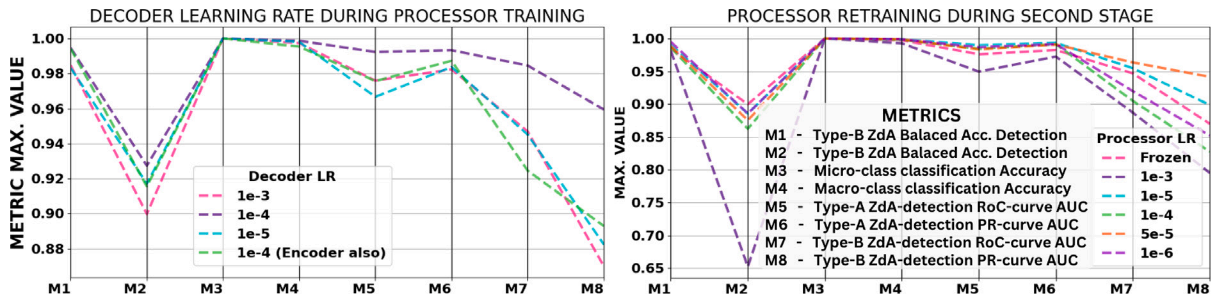
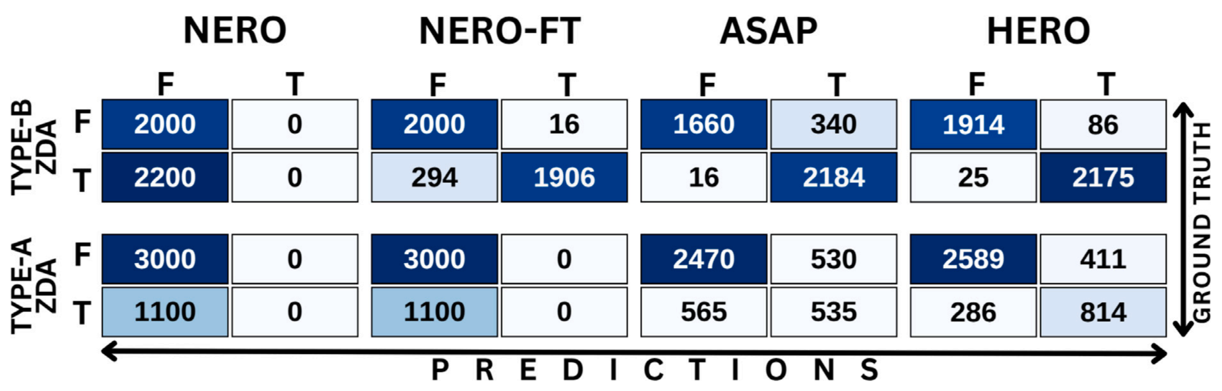


Fig. 7. Parallel coordinates plots for ablation studies of the two-stage training phase in HERO. To evaluate if a hard decoupling of the gradients is the most effective way of training the pipeline, several experiments were made in which the learning rates of the decoder and/or encoder were one or two orders of magnitude smaller than the learning rate of the processor during the first phase (left), and were the processor continued to be trained in the second phase with small learning rates (right). Both plots track important evaluation metrics during evaluation rounds at the second training phase.



		NERO F	NERO T	NERO-FT F	NERO-FT T	ASAP F	ASAP T	HERO F	HERO T
TYPE-B ZDA	F	2000	0	2000	16	1660	340	1914	86
TYPE-B ZDA	T	2200	0	294	1906	16	2184	25	2175
TYPE-A ZDA	F	3000	0	3000	0	2470	530	2589	411
TYPE-A ZDA	T	1100	0	1100	0	565	535	286	814

PREDICTIONS
GROUND TRUTH

Fig. 8. Confusion matrices for the last evaluation round performed during the training experiments described in Section 5.4. The matrices correspond to the binary type-B (upper-row) and type-A (lower-row) ZdA classification tasks. HERO shows superiority with respect to the rest of the baselines in both tasks. .

injection technique over the synthetic data that regularises eventual overfitting in the processor. Secondly, reducing the learning rates in the encoder and decoders intuitively prevents these modules learn to perform the discriminative task.

Reducing the learning rate in the encoder and decoders could result in incrementing the robustness of the processor network but also in a degradation of the overall goal of NAR: The processor could overfit the synthetic data distributions and not converge. In our experiments, no decisive gains were obtained by introducing differential learning rates in the first training phase, although some increments in the AUC of the precision–recall curve were obtained using a learning rate of 1e-4 for the decoder module in this phase. When using different learning rates for the decoder with respect to the processor's learning rate of 1e-3, maximum values for different metrics obtained in the second training phase are visible in the left panel of Fig. 7.

Appendix D. Confusion matrices

The results reported in Section 5.4 show the evolution of functional performance metrics on a series of periodic evaluation rounds consisting on 20 episodic tasks. Each task qualified the inferences of the various baseline models over 210 samples (205 for the Type-A ZdA classification). These episodic tasks were formed with held-out samples from known classes and totally unknown or out-of-distribution classes. The samples from known classes exploited the Few-Shot classification scheme explained in Section 4.2, while those from unknown classes where classified on a zero-shot basis, i.e., without using labels. More specifically:

(1) In the case of type-B ZdA detection, samples from 2 *in-distribution* or known micro-classes were combined with samples from 2 distinct OOD or unknown micro-classes. For each known micro-class, 5-shot classification of 50 query was performed, for a total of $50 \times 2 = 100$ query samples of known micro-classes. Instead, the samples from unknown micro-classes had no support set, and 55 samples were classified on a zero-shot basis (i.e., without support set). Thus, in each episodic task, a total of $55 \times 2 = 110$ samples were from unknown micro-classes, i.e., from type-B ZdAs. These were the *positives* for the corresponding task.
(2) Regarding the type-A ZdA detection task, a similar approach was used, but only 1 macro-class was actually out-of-distribution while the other 3 macro-classes were in the training set. Again, OOD samples used no support-set. By doing so, the zero-shot classification abilities of HERO were correctly tested. Instead, the in-distribution samples were classified using 5-shot prototypical classification. Consequently, each episodic type-A ZdA detection task had $55 \times 1 = 55$ positive samples and $50 \times 3 = 150$ *negative* samples. (Note that the samples from one of the type-B classes in the previous task are now equipped with a 5-shot support set which is not subject to evaluation any more).

Note that, by restricting the number of type-A ZdA classes to 1, the hierarchical distinction between type-B and type-A ZdA samples was correctly evaluated: otherwise, the second processor layer could simply learn to replicate the inferences of the first processor layer, without differentiating the degree of novelty among different anomalies.

Multiplying the episodic samples for the number of evaluation episodes or tasks, namely, 20, one can realise that each periodic evaluation procedure was composed as follows:

(1) For the type-B ZdA detection task, 4200 samples were classified, 2200 of which were actual type-B ZdAs or positives while the rest 2000 were related to known micro-classes (negatives). (Positive samples were $2 \times 55 \times 20 = 2200$, while, by excluding the 5-shot support set, negative samples were $2 \times 50 \times 20 = 2000$).
(2) For the type-A ZdA detection task, 4100 samples were classified, 1100 of which were positives because only 1 macro-class among those totally excluded from the training set was fed to the pipelines at each episodic task. Positive samples were $1 \times 55 \times 20 = 1100$, while, excluding the 5-shot support set, negative samples were $3 \times 50 \times 20 = 3000$.

Fig. 8 contains the cumulative confusion matrices for the last iteration of type-B (upper row of plots) and type-A (lower row) evaluation procedures for each one of the baselines runs in Section 5.4. The confusion matrices that correspond to HERO indicate better performance with respect to other baselines. It is easy to see how NERO and NERO-FT get stuck on a deterministic behaviour, where (unbalanced) accuracy is augmented by deeming all input samples as known. Only ASAP escapes from such a deterministic behaviour. Still, HERO shows superior (balanced) accuracy with respect to ASAP in both tasks. Again, recall that the evaluation procedures are periodically run during training to assess if the baseline is making progress or if the training should be stopped because the pipeline has converged or got stacked on deceptive learning path. The confusion matrices of previous evaluation procedures are available on the public Wandb board of the project alongside additional insights, e.g. layer-based gradients, dimensional PCA reduction of the latent representations, etc.

Data availability

No data was used for the research described in the article.

References

[1] R. De la Torre Vico, R. Magán-Carrión, R.A. Rodríguez-Gómez, Exploring the use of LLMs to understand network traces, in: International Conference on EUropean Transnational Education, Springer, 2024, pp. 122–131.
[2] Y. Ginige, T. Dahanayaka, S. Seneviratne, TrafficGPT: An LLM approach for open-set encrypted traffic classification, in: Proceedings of the Asian Internet Engineering Conference 2024, 2024, pp. 26–35.
[3] R. Ahmad, I. Alsmadi, W. Alhamdani, L. Tawalbeh, Zero-day attack detection: a systematic literature review, Artif. Intell. Rev. (2023) 1–79.
[4] Y. Guo, A review of machine learning-based zero-day attack detection: Challenges and future directions, Comput. Commun. 198 (2023) 175–185.
[5] M. Huisman, J.N. Van Rijn, A. Plaat, A survey of deep meta-learning, Artif. Intell. Rev. 54 (6) (2021) 4483–4541.
[6] D. Wang, Y. Cheng, M. Yu, X. Guo, T. Zhang, A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning, Neurocomputing 349 (2019) 202–211.
[7] D. Wang, M. Zhang, Y. Xu, W. Lu, J. Yang, T. Zhang, Metric-based meta-learning model for few-shot fault diagnosis under multiple limited data conditions, Mech. Syst. Signal Process. 155 (2021) 107510.
[8] Z. Zhang, Q. Liu, S. Qiu, S. Zhou, C. Zhang, Unknown attack detection based on zero-shot learning, IEEE Access 8 (2020) 193981–193991.
[9] J.F. Cevallos M., A. Rizzardi, S. Sicari, A. Coen-Porisini, NERO: Neural algorithmic reasoning for zeRO-day attack detection in the IoT: A hybrid approach, Comput. Secur. 142 (2024) 103898.
[10] J.F. Cevallos M., A. Rizzardi, S. Sicari, A. Coen Porisini, ASAP: Automatic synthesis of attack prototypes, an online-learning, end-to-end approach, Comput. Netw. (2024) 110828.
[11] F. Pourpanah, M. Abdar, Y. Luo, X. Zhou, R. Wang, C.P. Lim, X.-Z. Wang, Q.J. Wu, A review of generalized zero-shot learning methods, IEEE Trans. Pattern Anal. Mach. Intell. 45 (4) (2022) 4051–4070.
[12] M.A. Alsoufi, S. Razak, M.M. Siraj, I. Nafea, F.A. Ghaleb, F. Saeed, M. Nasser, Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review, Appl. Sci. 11 (18) (2021) 8383.
[13] O.H. Abdulganiyu, T. Ait Tchakoucht, Y.K. Saheed, A systematic literature review for network intrusion detection system (IDS), Int. J. Inf. Secur. 22 (5) (2023) 1125–1162.
[14] C. Wang, H. Zhou, Z. Hao, S. Hu, J. Li, X. Zhang, B. Jiang, X. Chen, Network traffic analysis over clustering-based collective anomaly detection, Comput. Netw. 205 (2022) 108760.
[15] M. Al-Zewairi, S. Almajali, M. Ayyash, Unknown security attack detection using shallow and deep ANN classifiers, Electronics 9 (12) (2020) 2006.
[16] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapé, A hierarchical hybrid intrusion detection approach in IoT scenarios, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–7.
[17] P. Veličković, C. Blundell, Neural algorithmic reasoning, Patterns 2 (7) (2021) 100273.
[18] Z.K. Maseer, Q.K. Kadhim, B. Al-Bander, R. Yusof, A. Saif, Meta-analysis and systematic review for anomaly network intrusion detection systems: Detection methods, dataset, validation methodology, and challenges, IET Networks 13 (5–6) (2024) 339–376.
[19] T.T. thein, Y. Shiraishi, M. Morii, Few-shot learning-based malicious IoT traffic detection with prototypical graph neural networks, Ieice Transactions Inf. Syst. 106 (9) (2023) 1480–1489.
[20] Z. Aouini, A. Pekar, NFStream: A flexible network data analysis framework, Comput. Netw. 204 (2022) 108719.
[21] M. Sarhan, S. Layeghy, N. Moustafa, M. Portmann, Netflow datasets for machine learning-based network intrusion detection systems, in: Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10, Springer, 2021, pp. 117–135.
[22] J.F. Cevallos M., A. Rizzardi, S. Sicari, A.C. Porisini, Deep reinforcement learning for intrusion detection in Internet of Things: Best practices, lessons learnt, and open challenges, Comput. Netw. 3 (3) (2023) 110016.
[23] J. Liu, Z. Shen, Y. He, X. Zhang, R. Xu, H. Yu, P. Cui, Towards out-of-distribution generalization: A survey, 2021, arXiv preprint arXiv:2108.13624.
[24] J. Yang, K. Zhou, Y. Li, Z. Liu, Generalized out-of-distribution detection: A survey, Int. J. Comput. Vis. (2024) 1–28.
[25] M. Kaya, H.Ş. Bilge, Deep metric learning: A survey, Symmetry 11 (9) (2019) 1066.
[26] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[27] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. (Csur) 53 (3) (2020) 1–34.
[28] P. Veličković, M. Bošnjak, T. Kipf, A. Lerchner, R. Hadsell, R. Pascanu, C. Blundell, Reasoning-modulated representations, in: Learning on Graphs Conference, PMLR, 2022, pp. 50–51.
[29] B. Ibarz, V. Kurin, G. Papamakarios, K. Nikiforou, M. Bennani, R. Csordás, A.J. Dudzik, M. Bošnjak, A. Vitvitskyi, Y. Rubanova, et al., A generalist neural algorithmic learner, in: Learning on Graphs Conference, PMLR, 2022, pp. 1–2.
[30] seladb, PcapPlusPlus, 2023, '(Accessed 18 November 2023)', https://pcapplusplus.github.io/.
[31] S. Machmeier, V. Heuveline, heiFIP: A network traffic image converter, 2023, http://dx.doi.org/10.5281/zenodo.8348868.
[32] J. Krupski, W. Graniszewski, M. Iwanowski, Data transformation schemes for CNN-based network traffic analysis: A survey, Electronics 10 (16) (2021) 2042, http://dx.doi.org/10.3390/electronics10162042, Number: 16 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. https://www.mdpi.com/2079-9292/10/16/2042.
[33] T. Shapira, Y. Shavitt, FlowPic: A generic representation for encrypted traffic classification and applications identification, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1218–1232.
[34] Cisco Systems, Cisco IOS NetFlow Version 9 Flow-Record Format - White Paper, 2011, Online https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.pdf.
[35] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[36] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Networks Learn. Syst. 32 (1) (2020) 4–24.
[37] W.L. Hamilton, Graph representation learning, Synth. Lect. Artifical Intell. Mach. Learn. 14 (3) (2020) 1–159.
[38] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al., Graph attention networks, Stat 1050 (20) (2017) 10–48550.
[39] B. Xu, Empirical evaluation of rectified activations in convolutional network, 2015, arXiv preprint arXiv:1505.00853.
[40] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, 2021, arXiv preprint arXiv:2105.14491.
[41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[42] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1314–1324.
[43] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future Gener. Comput. Syst. 100 (2019) 779–796.
[44] M.A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and iIoT applications for centralized and federated learning, IEEE Access 10 (2022) 40281–40306.

[45] J. Iyengar, M. Thomson, QUIC: A UDP-Based Multiplexed and Secure Transport, (RFC 9000) Internet Engineering Task Force, 2021, http://dx.doi.org/10.17487/RFC9000, Request for Comments RFC 9000, Num Pages: 151 [Online]. Available: https://datatracker.ietf.org/doc/rfc9000.

[46] P. Fegade, T. Chen, P.B. Gibbons, T.C. Mowry, The CoRa tensor compiler: Compilation for ragged tensors with minimal padding, 2022, arXiv:2110.10221.

[47] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, S. Hochreiter, xLSTM: Extended long short-term memory, 2024, arXiv:2405.04517.

[48] H. Li, J. Song, M. Xue, H. Zhang, J. Ye, L. Cheng, M. Song, A survey of neural trees, 2022, arXiv preprint arXiv:2209.03415.

[49] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.

[50] A.J. Dudzik, P. Veličković, Graph neural networks are dynamic programmers, Adv. Neural Inf. Process. Syst. 35 (2022) 20635–20647.

[51] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.

[52] P. Veličković, R. Ying, M. Padovano, R. Hadsell, C. Blundell, Neural execution of graph algorithms, 2019, arXiv preprint arXiv:1910.10593.

[53] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021, arXiv preprint arXiv:2104.13478.

[54] A. Deac, P. Veličković, O. Milinković, P.-L. Bacon, J. Tang, M. Nikolić, Xlvin: executed latent value iteration nets, 2020, arXiv preprint arXiv:2010.13146.

[55] L. Beurer-Kellner, M. Vechev, L. Vanbever, P. Veličković, Learning to configure computer networks with neural algorithmic reasoning, 2022, arXiv preprint arXiv:2211.01980.

[56] P. Veličković, A.P. Badia, D. Budden, R. Pascanu, A. Banino, M. Dashevskiy, R. Hadsell, C. Blundell, The CLRS algorithmic reasoning benchmark, in: International Conference on Machine Learning, PMLR, 2022, pp. 22084–22102.

[57] Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, P. Velickovic, Combinatorial optimization and reasoning with graph neural networks, J. Mach. Learn. Res. 24 (2023) 130–131.

[58] C. Elkan, Results of the KDD'99 classifier learning, Acm Sigkdd Explor. Newsl. 1 (2) (2000) 63–64.

[59] L. Vu, Q.U. Nguyen, D.N. Nguyen, D.T. Hoang, E. Dutkiewicz, et al., Learning latent representation for iot anomaly detection, IEEE Trans. Cybern. 52 (5) (2020) 3769–3782.

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

[61] J. Yang, X. Chen, S. Chen, X. Jiang, X. Tan, Conditional variational auto-encoder and extreme value theory aided two-stage learning approach for intelligent fine-grained known/unknown intrusion detection, IEEE Trans. Inf. Forensics Secur. 16 (2021) 3538–3553.

[62] M. Sarhan, S. Layeghy, M. Gallagher, M. Portmann, From zero-shot machine learning to zero-day attack detection, Int. J. Inf. Secur. (2023) 1–13.

[63] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ieee, 2009, pp. 1–6.

[64] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.

[65] Y. Li, F. Gimeno, P. Kohli, O. Vinyals, Strong generalization and efficiency in neural programs, 2020, arXiv preprint arXiv:2007.03629.

[66] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.

[67] K. Paupamah, S. James, R. Klein, Quantisation and pruning for neural network compression and regularisation, in: 2020 International SAUPEC/RobMech/PRASA Conference, IEEE, 2020, pp. 1–6.

[68] F. Petersen, C. Borgelt, H. Kuehne, O. Deussen, Deep differentiable logic gate networks, Adv. Neural Inf. Process. Syst. 35 (2022) 2006–2018.

[69] F. Hendaoui, A. Ferchichi, L. Trabelsi, R. Meddeb, R. Ahmed, M.K. Khelifi, Advances in deep learning intrusion detection over encrypted data with privacy preservation: a systematic review, Clust. Comput. (2024) 1–42.

[70] Y. He, W. Li, Image-based encrypted traffic classification with convolution neural networks, in: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), IEEE, 2020, pp. 271–278, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9172859/.

[71] Z. Diao, G. Xie, X. Wang, R. Ren, X. Meng, G. Zhang, K. Xie, M. Qiao, EC-GCN: A encrypted traffic classification framework based on multi-scale graph convolution networks, Comput. Netw. 224 (2023) 109614, Publisher: Elsevier [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623000592.

[72] Z. Zou, J. Ge, H. Zheng, Y. Wu, C. Han, Z. Yao, Encrypted traffic classification with a convolutional long short-term memory neural network, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2018, pp. 329–334, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8622812/.

[73] X. Jia, M. Zhang, Encrypted packet inspection based on oblivious transfer, Secur. Commun. Networks 2022 (2022).

[74] J.F. Cevallos M., A. Rizzardi, S. Sicari, A. Coen-Porisini, TIGER: An Open-Source Cyber-Threat Intelligence Game Environment for Reinforcement Learning, Tech. Report. Under Review, [Online]. Available: https://github.com/DISTA-IoT/smartville.

[75] F. Zhang, Y. Qu, Y. Xu, S. Wang, Graph embedding-based approach for detecting group shilling attacks in collaborative recommender systems, Knowl.-Based Syst. 199 (2020) 105984.

[76] J. Wang, Z. Zhang, H. Zha, Adaptive manifold learning, in: L. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems, vol. 17, MIT Press, 2004, [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2004/file/eb0ecdb070a1a0ac46de0cd733d39cf3-Paper.pdf.

[77] N. Lei, D. An, Y. Guo, K. Su, S. Liu, Z. Luo, S.-T. Yau, X. Gu, A geometric understanding of deep learning, Engineering 6 (3) (2020) 361–374.

[78] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size, 2016, ArXiv E-Prints.

Jesús F. Cevallos-Moreno received a Ph.D. in Computer Science Engineering from Sapienza University (Rome) in 2022. He now covers a postdoc researcher position at the University of Insubria (Varese). His main research interests are industrial/ethical applications of Deep Learning with a special focus on Natural Language Processing and Neural Algorithmic Reasoning.

Alessandra Rizzardi is an Assistant Professor at University of Insubria (Varese), where she received BS/MS degree in Computer Science 110/110 cum laude in 2011 and 2013, respectively. In 2016 she got Ph.D. in Computer Science and Computational Mathematics at the same university, under the guidance of Prof. Sabrina Sicari. Her research activity is on WSN and IoT security issues. She is member of ETT, ITL, and Sensors editorial board. She is IEEE member.

Sabrina Sicari is a Full Professor at University of Insubria (Varese). She received the M.Sc. degree in Electronic Engineering, 110/110 cum laude, from the University of Catania, in 2002, where in 2006, she got Ph.D. in Computer and Telecommunications Engineering, followed by Prof. Aurelio La Corte. She is a COMNET, IEEE IoT, ETT, and ITL editorial board member. Her research concerns security, privacy, and trust in WSN, WMSN, IoT, and distributed systems. She is an IEEE senior member.

Alberto Coen-Porisini received a Dr. Eng. degree and Ph.D. in Computer Engineering from Politecnico di Milano in 1987 and 1992. He has been a Full Professor of Software Engineering at Universitá degli Studi dell'Insubria since 2001, Dean of the School of Science from 2006 and Dean from 2012 to 2018. His research regards specification/design of real-time systems, privacy models, and WSN.