Journal of Network and Systems Management (2026) 34:127
https://doi.org/10.1007/s10922-026-10097-8

SmartVille: A Framework for Realistic Deep Learning-Based Online Network Intrusion Detection

Jesús F. Cevallos Moreno[1] · **Alessandra Rizzardi**[1] · **Sabrina Sicari**[1] · **Alberto Coen-Porisini**[1]

Received: 19 October 2025 / Revised: 26 April 2026 / Accepted: 16 July 2026

Abstract
Deep learning has generated strong results for network intrusion detection, but much of the literature still treats the problem as static offline classification, leaving unclear how such models should be conceived for deployment. In practice, intrusion detection must learn from traffic streams, cope with previously unseen attacks, and exploit heterogeneous evidence sources without relying on bulky preprocessing or heavyweight models. This paper addresses that conceptual gap by introducing SmartVille, a framework for *formulating* and *studying* deep learning-based NID under online, open-world, and multi-modal assumptions. The main contribution is therefore not a new stand-alone detection algorithm, but a coherent research blueprint that integrates existing learning principles into a single technical vision. In particular, SmartVille advocates an end-to-end differentiable encode–process–decode organisation, in which neural encoders replace bulk feature engineering, online learning replaces static train-once evaluation, and collective anomaly detection is studied alongside supervised classification within the same framework. This perspective clarifies what SmartVille specifically solves: it provides a principled way to design, train, and benchmark adaptive NID models under realistic assumptions, while separating the theoretical contribution from its open-source implementation. Representative use cases show how the framework can be used to analyse curriculum design, input-modality composition, and architectural choices for adaptive intrusion detection research.

Keywords Network intrusion detection · Online learning · Deep learning

Extended author information available on the last page of the article

Published online: 30 July 2026

1 Introduction

Deep learning has produced a large body of work on Network Intrusion Detection (NID), yet many proposals remain difficult to translate into operational systems [1–3]. A central reason is that the problem is often treated as a static offline classification task, whereas deployed NID is inherently dynamic: traffic arrives as a stream, attack patterns evolve after deployment, useful evidence is distributed across multiple modalities, and the resulting models must remain efficient enough for constrained environments such as the IoT [4–8]. These requirements are tightly coupled, which makes isolated algorithmic improvements insufficient for studying deployable deep learning–based NID.

This motivates the need for a *framework* rather than a single detector. The goal of this study is to define and instantiate a research framework for conceiving, training, and evaluating deep learning–based NID under realistic operating assumptions. In particular, the paper asks the following research questions:

1. **RQ1:** How should deep learning–based NID be formulated when learning and inference must take place online, directly from traffic streams rather than from offline, bulk-preprocessed datasets?
2. **RQ2:** How can such systems be designed for open-world conditions, in which previously unseen attack behaviours emerge during deployment and require continual adaptation?
3. **RQ3:** How can multiple complementary signals—including raw traffic, traffic statistics, and node-level measurements—be integrated into an end-to-end differentiable pipeline without resorting to heavyweight modelling choices?
4. **RQ4:** What kind of controllable experimental environment is needed to study these questions reproducibly and benchmark different neural architectures under realistic network conditions?

To answer these questions, we introduce SmartVille, a framework for systematic research on practical deep learning–based NID. SmartVille is not presented as a single model optimised for leaderboard accuracy; rather, it is a blueprint for how the NID lifecycle can be organised around online learning, open-world adaptation, multi-modal observation, and efficient end-to-end neural processing. The **main contributions** of the paper are therefore the following:

1. A principled formulation of NID as an **online and open-world learning problem**, supported by meta-learning curriculum generation and dynamic labelling mechanisms that expose models to known, simulated-unknown, and genuinely unseen patterns during training and evaluation.
2. A **multi-modal input design** that assembles parallel online tensor streams from raw traffic, traffic statistics, and node metrics, enabling richer behavioural modelling while remaining suitable for low-resource scenarios.
3. An **end-to-end differentiable NID pipeline** based on an encode–process–decode organisation, so that different lightweight neural architectures can be integrated

and benchmarked without depending on bulky handcrafted preprocessing or non-differentiable anomaly modules.
4. An **open-source experimental implementation** of the framework, including an extensible network environment generator based on containerised GNS3 topologies with configurable attack, honeypot, monitoring, and forwarding nodes, together with a WebGUI and MLOps support for reproducible experimentation.

Overall, this work provides a research blueprint for developing and evaluating online, multi-modal, open-world, and practically deployable deep learning methods for network intrusion detection.

The remainder of this paper is structured as follows. A review of the main solutions available in literature and relevant to this work is presented in §2. The main theoretical working principles of the proposed framework are presented in §3 and its suitability for online learning is discussed. An examination of the multi-modal feature space and the online input-space shaping mechanism is instead presented in §4. The modular, end-to-end NID pipeline for benchmarking diverse neural architectures is described in §5. The open-source implementation of SmartVille is exposed in §6 alongside the documentation of example use cases. Finally §8 concludes the paper and outlines future research directions.

2 Related Works

This work intercepts three main axes of research: (1) testbeds for network intrusion detection, (2) training frameworks for deep learning-based NID, (3) architectural research and collective anomaly detection, as detailed herein.

2.1 Testbeds for Network Intrusion Detection

Cybersecurity testbeds remain a de facto entry point for ML researchers studying network intrusion detection [9, 10]. However, many existing platforms are either closed-source or difficult to access, which hinders reproducibility and systematic experimentation [11–13]. Within open-source initiatives, an important distinction concerns the *kind of network evidence* exposed to the learning system: several environments abstract attacks and defences into scalar states or reward variables [14–16], whereas other efforts move closer to operational settings by exposing actual traffic traces and packet-level evidence [17–21]. The position advocated by SmartVille is aligned with this second line of work: if the objective is to study deployable neural NID, then the framework should let models observe traffic as traffic, rather than only as pre-digested scalar surrogates. In this sense, the role of the testbed is not to claim a definitive implementation, but to provide a controllable proof of concept showing that heterogeneous `.pcap`-based behaviours can be incorporated incrementally as additional traces become available [22].

2.2 Training Frameworks for Deep Learning-Based NID

Recent research on learning frameworks for NID spans federated learning [23–25], adversarial training [26, 27], continual learning [28, 29], generative methods [30, 31], explainability [32, 33], lightweight architectures [34], reinforcement learning [35, 36], ensemble methods [37–39], and scenario-specific adaptations [40–43].

To ground these learning paradigms in reality, recent studies have increasingly emphasized practical end-to-end implementations. For example, Naseh et al. [8] and Fusco et al. [44] deploy federated and few-shot learning frameworks directly on physical IoT hardware, profiling strict energy, memory, and communication constraints at the edge. Similarly, established end-to-end NID frameworks like AB-TRAP [45] have formalized the entire lifecycle from dataset synthesis to the kernel-space realization of machine learning models. While these valuable works focus heavily on the physical/spatial constraints of hardware and rely on periodic offline retraining, SmartVille tackles a complementary operational challenge: the continuous temporal dynamics of the data streams.

Within this landscape, the main standpoint of SmartVille is that *online learning should be treated as the default formulation*, not as an occasional update phase appended to an otherwise offline pipeline [46]. In other words, the detector should begin making inferences from the moment traffic is observed, keep learning while deployed, and avoid reliance on bulky preprocessing, bulk collection, or periodic stop-and-retrain cycles. This view is especially coherent under open-world conditions, where the set of relevant traffic patterns is not fixed in advance and novel behaviours may appear after deployment [5, 47–49]. The contribution of the framework is therefore to organise NID around concurrent streaming inference and adaptation, rather than around a train-once workflow later amended with periodic continual-learning steps.

This shift in perspective becomes particularly relevant when evaluating the time complexity and computational overhead of neural NID. For instance, recent studies analyzing the IoT23 dataset [50] report training and evaluation times ranging from 1.5 to over 70 h, costs that are primarily driven by the overhead of offline bulk preprocessing and static multi-epoch training. Similarly, comprehensive experimental reviews of neural approaches [51] provide valuable time-complexity analyses, but these are typically grounded in the offline processing of pre-extracted, high-level statistical datasets (e.g., CIC-IDS) rather than raw network streams. In such offline paradigms, the system must ingest and process massive static datasets before becoming operational. By contrast, the SmartVille framework proposes bypassing bulk feature engineering and static multi-epoch training altogether. Instead, it advocates for a streaming, multi-modal tensor assembly coupled with end-to-end neural encoders, allowing models to begin learning and generating inferences continuously upon deployment. This approach shifts the evaluation of time complexity away from offline data-preparation phases and onto the live execution of the neural pipeline itself. Furthermore, while other advanced methods have successfully employed Deep Reinforcement Learning (DRL) to handle unlabeled data and reduce training complexity in offline, pre-extracted feature environments [52], the SmartVille blueprint suggests that such optimization strategies can be studied as complementary mechanisms.

Evaluating how DRL-based class-balancing or semi-supervised methods behave on raw, streaming traffic rather than static datasets represents exactly the kind of open-world research direction that the framework is designed to model and support.

2.3 Architectural Research for Collective Anomaly Detection

Architectural inductive biases in deep learning-based NID have received growing attention. A prominent example is the alignment between network data and geometric deep learning, which is well suited to the relational structure of traffic-derived observations [53, 54]. At the same time, NID is often approached as a time-series classification task, where recurrent models such as LSTMs and GRUs capture temporal dependencies [55, 56], while CNNs remain attractive in high-dimensional spaces because of parameter sharing and efficient local pattern extraction [1, 57]. The architectural position of SmartVille is not that one processor should dominate all others, but that the framework should make these alternatives comparable within a common end-to-end differentiable organisation. Concretely, SmartVille advocates an encode–process–decode decomposition [58, 58, 59], which separates representation learning, relational/temporal processing, and decision decoding so that different geometric or sequential processors can be swapped with limited redesign. This modularisation is also motivated by learning dynamics: distributing responsibilities across neural components can reduce the burden placed on a single monolithic model and make experimentation more systematic. Complementarily, SmartVille treats metric-based meta-learning [60] as a principled mechanism for modelling unknowns. By exposing the learner to known classes, simulated unknowns, and held-out unknown behaviours, the framework supports collective anomaly detection in a way that is explicitly tied to open-world deployment, where genuinely unseen attacks are expected to emerge after the system is in use.

3 A Unified Framework for Network Intrusion Detection: Supervised Classification and Collective Anomaly Detection

The paper proposes a structured perspective on machine learning-based NID by formalising it through two core tasks: **supervised classification** and **Collective Anomaly Detection (C-AD)** [61], which are detailed in the following sub-sections. The purpose of this dual-task formulation is not to advocate a specific detector, but to characterise the two complementary inference regimes that arise in open-world NID: assigning inputs to already-internalised attack categories, and organising previously unseen traffic into coherent relational structures.

3.1 Supervised Classification

Supervised classification constitutes the dominant paradigm in NID research due to its conceptual simplicity and direct applicability. However, existing approaches typically rely on offline, bulk-processing frameworks, which hinder timely adaptation to evolving traffic distributions in real-world networks. In contrast, our approach

embeds supervised classification within an **online learning** paradigm [46], leveraging **Prototypical Learning (PL)** [62] to enhance adaptability and data efficiency.

3.1.1 Prototypical Learning (PL)

Prototypical Learning (PL) is a metric-based meta-learning framework in which input samples are classified on the basis of their distances to learned class prototypes in a representation space. Formally, given a support set $\mathcal{S} = \{(x_i, y_i)\}$, the prototype for class c is computed as:

$$\mathrm{p}_c = \frac{1}{|\mathcal{S}_c|} \sum_{(x_i, y_i) \in \mathcal{S}_c} f_\theta(x_i),$$

where f_θ denotes the embedding function and $\mathcal{S}_c$ the subset of support samples labelled c. A query sample x_q is assigned to the class whose prototype minimises a divergence measure $\mathcal{D}$ between the embedding of the query and p_c :

$$\hat{y}_q = \arg\min_c \ \mathcal{D}\big(f_\theta(x_q), \mathrm{p}_c\big) \tag{1}$$

This paradigm embodies the *relational bottleneck* inductive bias–focusing on inter-sample relations, rather than absolute feature values, thus resulting in superior data efficiency and robustness to distribution shifts [63]. Consequently, PL has become a cornerstone of Few-Shot Learning [64], making it particularly suitable for adaptive NID systems [5].

3.1.2 Dynamic Threat Intelligence

A key advantage of PL over conventional classifiers lies in its inherent support for **dynamic class expansion**: new attack types can be integrated at inference time by simply computing and adding their prototypes without retraining the entire model [5, 48]. This capability enables continuous enrichment of the system's *cyber-threat intelligence*, allowing real-time incorporation of emerging threats identified through dynamic traffic labelling (see §4.2).

Despite its practical significance, this dynamic aspect is often overlooked in current NID literature. SmartVille explicitly models and evaluates this dimension, emphasising the need for deployable systems capable of evolving alongside the threat landscape.

3.2 Collective Anomaly Detection (C-AD)

While Anomaly Detection (AD) is widely studied in an unsupervised setting, most approaches fail to account for two critical characteristics of real-world network anomalies: **heterogeneity** and **concurrency**. More in detail, anomalous traffic often comprises multiple distinct patterns occurring simultaneously, rather than a single homogeneous deviation. For this reason, we focus on *collective* anomaly detection:

once the anomaly set is understood as internally structured, the theoretical object of interest is no longer a scalar anomaly score for each sample, but the latent partition induced by relations among anomalous samples.

3.2.1 Anomaly Heterogeneity and Concurrency

Anomalous data, though Out-Of-Distribution (OOD) with respect to known benign or malicious patterns, may themselves form coherent clusters corresponding to novel attack types [65]. This observation motivates C-AD as the appropriate abstraction: the learning problem is to preserve cluster-level regularities among OOD samples, so that novelty is represented not merely as deviation from the known, but as emergent structure within the unknown. By grouping such anomalies collectively (i.e., performing **C-AD**) SmartVille enables downstream analysis where field experts can interpret the geometric relationships between anomaly clusters and known prototypes to infer intent or origin.

Accordingly, the goal shifts from binary anomaly scoring to structured discovery, which is better aligned with open-world intrusion analysis because it distinguishes between "being novel" and "belonging to the same novel mechanism" [66].

3.2.2 Kernel Regression Formulation

Clustering can be formalised as a **set-to-set function** that maps an input set of data points to a partition, defined implicitly by a relational structure. This structure is captured by a **kernel (or adjacency) matrix** $\mathrm{K} \in \{0,1\}^{n \times n}$, where $K_{ij} = 1$ if samples x_i and x_j belong to the same cluster. This representation is central to our formulation because it makes the target of C-AD explicit at the level of pairwise relations, rather than tying the problem to any non-differentiable clustering routine or to any fixed number of anomaly classes.

The task of inferring K from data constitutes **kernel regression**, which is a natural fit for modern deep architectures such as Transformers and Graph Neural Networks (GNNs), that inherently model pairwise interactions. Specifically, self-attention mechanisms compute similarity scores across all pairs in a set, effectively approximating a soft kernel matrix. In SmartVille, C-AD is rendered differentiable by minimising the discrepancy between predicted and ground-truth kernels via a suitable loss function (e.g., binary cross-entropy or contrastive loss).

3.2.3 C-AD as Supervised Meta-learning

To reconcile the unsupervised nature of anomaly detection with the need for supervisory signals in kernel regression, we adopt the **meta-learning** setup defined in the ASAP blueprint [60], which combines prototypical learning and dynamic labelling to automatically create or synthesise anomaly prototypes. The reason for adopting this construction is methodological: if C-AD is to be learned end-to-end, then the training procedure must expose the model to unknown-pattern structure during optimisation while still reserving genuinely unseen structure for evaluation. More specifically, the set of traffic patterns, denoted by $\mathscr{P}$, is partitioned as follows:

$$\mathscr{P} := \mathscr{P}_K^{train} \cup \mathscr{P}_U^{train} \cup \mathscr{P}_U^{test} \tag{2}$$

where each element of the partition corresponds to a disjoint subset of patterns and has a specific role in the training and evaluation phase of the NID module. The labels "known" and "unknown" here denote the epistemic status of traffic patterns with respect to the learner, not an ontological property of the traffic itself: "known" patterns are those available for direct class-supervised adaptation, whereas "unknown" patterns are withheld from such direct supervision and are only accessible through their latent relational organisation.:

- **Train-known**: $\mathscr{P}_K^{\text{train}}$, with $|\mathscr{P}_K^{\text{train}}| \geq 0$, provides labelled data for standard supervised classification.
- **Train-unknown**: $\mathscr{P}_U^{\text{train}}$, with $|\mathscr{P}_U^{\text{train}}| > 1$, contains unlabelled patterns whose implicit labels define the ground-truth kernel during training.
- **Test-unknown**: $\mathscr{P}_U^{\text{test}}$, with $|\mathscr{P}_U^{\text{test}}| > 1$, consists of entirely held-out OOD patterns used to evaluate the generalization of clustering performance.

This partitioning transforms C-AD into a **supervised meta-learning task**, where the model learns not to recognize specific patterns, but rather the underlying principles of **inter-cluster separability** and **intra-cluster cohesion**. In this sense, $\mathscr{P}_U^{\text{train}}$ and $\mathscr{P}_U^{\text{test}}$ play distinct theoretical roles: the former provides relational supervision over unknownness as a *training signal*, while the latter tests whether the learned relational inductive bias transfers to unknownness as a *generalisation regime*. The resulting meta-distribution focus encourages inductive biases that generalise to previously unseen anomaly types.

We stress that the hidden labels used to construct ground-truth kernels for $\mathscr{P}_U^{\text{train}}$ are primarily a research-time supervision device, introduced to train differentiable C-AD models under controlled open-world conditions. In operational deployments, such supervision would only be approximated retrospectively through analyst investigation, incident correlation, sandboxing, or threat-intelligence enrichment, and is therefore weaker and delayed compared with the oracle labels available during meta-training.

A central tenet of SmartVille is that this formulation more faithfully captures the open-world epistemology of NID: some traffic can be assigned by direct recognition, some can only be organised relationally, and the latter must be evaluated under distributional novelty rather than class repetition.

Together, supervised classification and C-AD use online Input-Space shaping and meta-training curricula variation to form a cohesive framework for developing and evaluating adaptive NID systems grounded in deep learning principles.

Fig. 1 offers a simple schematic representation of the main degrees of freedom envisioned by SmartVille for training and evaluating machine learning-based NIDs. Its purpose is not to prescribe a single experimental protocol, but to clarify that the framework supports multiple tasks with different levels of difficulty, ranging from supervised classification to OOD anomaly detection, and further to OOD collective anomaly detection—that is, the relational clustering of OOD samples—as the most challenging setting. Within this online-learning perspective, SmartVille advo-

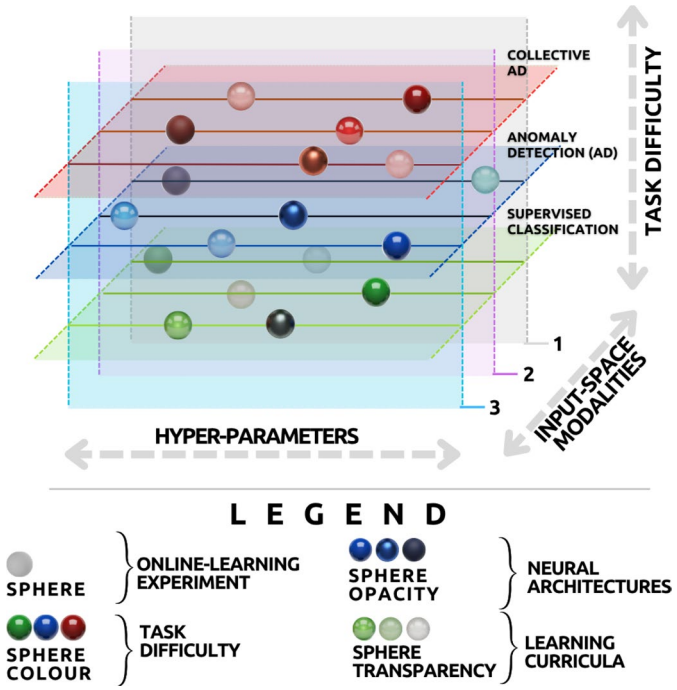

Fig. 1 The main components of the SmartVille framework

cates systematic variation of the input modalities, hyperparameter settings, training curricula, and neural architectures, so that these design dimensions can be studied jointly while preserving a consistent stream-oriented learning regime. The following section provides detailed information on how data streams need to be processed for this purpose.

4 Input-Space Shaping

Online learning reflects the *open-world* paradigm in multi-class classification, wherein the set of patterns to be classified is inherently incomplete and continuously evolving. This setting introduces two primary forms of incompleteness in the learning curriculum: (1) not all observed patterns during training are explicitly labelled, and (2) previously unseen patterns (whether labelled or unlabelled) may emerge at any stage of the training process. In other words, novel data instances corresponding to entirely new traffic patterns may appear after extended periods of model deployment and training.

Such features necessitate specialised mechanisms for structuring ML experiments in online NID. Specifically, the experimental timeline must support on-demand triggering of traffic patterns and dynamic labelling of incoming data. This is not an implementation preference, but a methodological requirement: if the objective is to study continual adaptation under distribution shift, then the learner must be exposed

to traffic streams whose composition can change during training and evaluation. A controllable network environment is therefore needed not because it is the only possible engineering solution, but because it provides the minimum experimental structure required to relate theoretical assumptions about open-world learning to observable traffic dynamics.

4.1 Tensor Assembly as Stream Processing

A suitable research testbed for deep learning-based NID requires a state-space encoding mechanism that delivers time-synchronised representations of the current network state to the inference engine. SmartVille therefore adopts a stream-oriented encoding view that avoids bulk preprocessing and preserves end-to-end trainability. The rationale is theoretical before practical: if learning and inference are assumed to occur online, then the input representation should itself be assembled online from modalities that can be observed contemporaneously. Under this perspective, three feature spaces emerge as foundational rather than arbitrary choices, with room for future extension through additional domains:

1. **Traffic Statistics**: this feature space is retained because it provides a compact behavioural summary over short horizons, making it well suited to online adaptation when storing or reprocessing raw traces would be undesirable. Such statistics typically pertain to individual flows at the transport layer (i.e., layer 4) of the ISO/OSI model and include metrics such as packet size, inter-arrival times, flow duration, and protocol-specific attributes.
2. **Raw Traffic Data**: this feature space is included because flow statistics alone can suppress discriminative structure that is only visible in packet contents or short byte-level patterns. The framework does not assume that every deployment must use raw bytes, but argues that a realistic research setting should make this modality available whenever inspection policies permit it. Traditional deep packet inspection methods are often impractical at network edges due to computational overhead and privacy constraints. For this reason, SmartVille treats selective sampling—rather than exhaustive payload processing—as the relevant design principle, and uses SDN in the implementation as one concrete mechanism for exposing such samples to the learning pipeline [67].
3. **Network Node-Related Features**: this feature space is included because some attacks are expressed not only through network exchanges, but also through their side effects on end hosts. CPU utilisation, memory consumption, or interface load can therefore act as complementary evidence when network-only observations are ambiguous. These features should be monitored from participating network nodes whenever feasible and encoded as a parallel input stream, not to inflate the system with extra telemetry, but to test the research hypothesis that multi-modal evidence improves discrimination under open-world conditions.

Additional feature domains may be integrated into SmartVille to enhance the comprehensiveness of the input space. Crucially, however, input construction must treat data points as time-ordered sequences of features, analogous to trimmed realisations

of underlying stochastic processes. This temporal structuring is a direct consequence of the problem formulation: if attacks unfold as evolving behaviours rather than isolated snapshots, then the learner must receive temporally coherent observations. Sequence assembly is therefore justified not as a generic architectural preference, but as a representation choice aligned with the dynamics of online NID.

4.2 Dynamic Labelling

Cyber threat intelligence is inherently time-evolving, since new attack vectors, benign behaviours, and protocol anomalies emerge continuously. To reflect this dynamism, NID training environments must incorporate a dynamic labelling strategy that allows for the retrospective and prospective annotation of traffic patterns as knowledge evolves. The motivation is again methodological: once open-world deployment is assumed, labels can no longer be treated as a fixed table decided before training begins.

In SmartVille, dynamic labelling comprises two interrelated operations: (1) the reassignment of traffic patterns across the three subsets defined by the ASAP partition (introduced in Sect. 3.2.3), namely known patterns ($\mathscr{P}_K$), unknown patterns designated for training ($\mathscr{P}_U^{\text{train}}$), and unknown patterns reserved for testing ($\mathscr{P}_U^{\text{test}}$); (2) the routing of incoming data samples to training or evaluation buffers based on their current pattern labels and partition status. This mechanism is introduced to operationalise the distinction between known, simulated-unknown, and genuinely unseen behaviours, which would otherwise remain only a conceptual partition.

Formally, at a given time instant t, an input batch $\mathscr{B}_t$ containing $|\mathscr{B}_t|$ data points is processed not only for inference, but also through a concurrent dispatching mechanism. Specifically:

- Data points belonging to patterns in $\mathscr{P}_U^{\text{test}}$ are directed to test buffers.
- Data points associated with patterns in $\mathscr{P}_K \cup \mathscr{P}_U^{\text{train}}$ are routed to either training or test buffers according to a user-defined binomial distribution, enabling controlled evaluation and continual learning.

The buffering infrastructure serves multiple purposes: it facilitates periodic model retraining and performance evaluation, and it supports the formation of *support batches* used in prototypical learning, where inference relies on comparisons between query inputs and representative prototypes derived from labelled support sets, as explained in §3.1.1.

Furthermore, the dynamic labelling process necessitates buffers being capable of seamless reorganisation in response to changes in pattern partitioning. This adaptability ensures that the model remains aligned with the latest threat intelligence and maintains robustness against concept drift.

5 Neural Pipeline

This work adopts the **Encode–Process–Decode** (EPD) paradigm [68] as the foundational architecture for deep learning–based NID. The EPD framework modularises the neural pipeline into three distinct, composable components–encoder, processor, and decoder–connected in series. This design enforces a *single-responsibility principle*, enabling systematic experimentation with alternative neural architectures, while maintaining a consistent interface and end-to-end differentiability. The overall pipeline, illustrated in Fig. 2, supports both supervised classification and C-AD within a unified and adaptive framework.

5.1 Encoder Networks

The encoder transforms raw, multi-modal input data into a distributed latent representation, where semantic similarity corresponds to geometric proximity in the embedding space. The reason for introducing an encoder is not merely to add a standard deep learning block, but to replace bulky handcrafted preprocessing with a trainable representation map. In open-world NID this is particularly important because separability cannot be assumed a priori: the representation must remain plastic enough to absorb new evidence while preserving a geometry in which known classes, emerging classes, and anomalous structures can be related coherently.

Encoders typically implement point-wise transformations (e.g., fully connected or convolutional layers), but may incorporate recurrent or Temporal Convolutional Networks (TCNs) to model sequential dependencies in time-series features, such as flow statistics or system resource usage. The framework leaves this choice open because the theoretical commitment is to the role of the encoder, not to one specific family of networks. Sparse activations, regularisation, and normalisation are included as optional mechanisms to encourage compact and stable representations

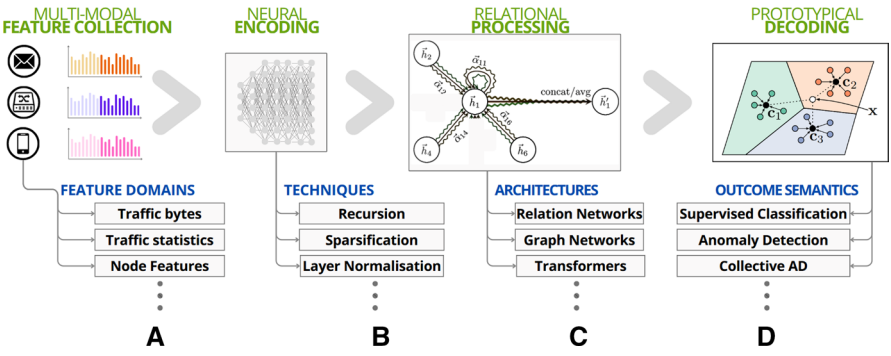

Fig. 2 The encode–process–decode (EPD) neural pipeline in SmartVille. The input data (**A**) is transformed by the encoder (**B**) into a structured latent representation. The processor (**C**) operates over this representation to capture relational patterns, performing tasks such as kernel regression for collective anomaly detection or prototype-based reasoning. Finally, the decoder (**D**) maps the processed representations to the output space, producing class association scores or anomaly cluster assignments. This modular design enables plug-and-play experimentation with diverse neural architectures while preserving end-to-end differentiability

under streaming updates. The encoder output serves as input to the processor module and is depicted in block (B) of Fig. 2.

5.2 Processor Networks

The processor module performs structured reasoning over the encoded representations, aligning internal computations with abstract algorithmic principles such as clustering, relational inference, or prototype formation [69]. Its inclusion is justified by the nature of the tasks introduced in Sect. 3: both prototypical classification and C-AD depend not only on point-wise descriptions, but on relations among samples. A processor is therefore needed whenever the hypothesis class must express interactions at the level of sets, neighbourhoods, or pairwise affinities.

In SmartVille, the processor is tasked with kernel regression as defined in §3.2.2, namely, for an online input batch $\mathscr{B}$ consisting of n data-points, it predicts a binary adjacency matrix $\mathrm{K} \in \{0,1\}^{n \times n}$ that encodes whether pairs of input samples belong to the same cluster. This choice is deliberate: adjacency prediction turns clustering into a differentiable supervised objective, avoiding reliance on non-differentiable clustering procedures that would interrupt gradient-based online adaptation. Under this formulation, architectures such as Transformers, Graph Neural Networks (GNNs), and Relation Networks become natural processor candidates because they directly encode pairwise interactions through attention or message passing [68, 70].

The processor can be trained using supervision derived from the ASAP meta-learning setup (see §3.2.3), where ground-truth kernels are constructed from hidden labels in $\mathscr{P}_U^{\text{train}}$. Intermediate supervision signals can further enhance algorithmic alignment and convergence stability. Processor architectures may vary from permutation-invariant set functions to graph-structured processors, depending on the desired inductive bias. This module is represented by block (C) in Fig. 2.

5.3 Decoder Networks

The decoder maps the processed representations to the final discriminative output space, tailoring the model's predictions to the specific task at hand. Its role is to preserve a clean separation between generic representation/relational computation and task-specific decision rules. In supervised classification, the decoder computes class association scores by measuring the distance between query embeddings and learned class prototypes, as defined in prototypical learning (§3.1.1). Distances are then transformed into probabilities through a Softmax function or compared against a learnable threshold to distinguish between known classes and anomalous OOD inputs.

For collective anomaly detection, the decoder bypasses class-level scoring and instead outputs the predicted kernel matrix directly, which is then evaluated against the ground-truth associations. Two supervised-clustering performance metrics are used for this purpose: the Adjusted Rand Index (ARI) and the Normalised Mutual Information (NMI), as shown in §7. These metrics are preferred because they evaluate structural agreement between partitions rather than superficial label identity, which is consistent with the objective of discovering coherent anomaly groupings.

Block (D) of Fig. 2 shows how the decoder finalises the inference process while allowing task-specific adaptation without modifying the core encoder or processor.

5.4 Design Philosophy

The EPD structure is introduced as a research abstraction rather than as a claim that any fixed three-block pipeline is universally optimal. Its justification is that the three required functions in this problem setting are conceptually distinct: online NID needs a mechanism to encode heterogeneous streams, a mechanism to reason over their relations, and a mechanism to map those relations to task-specific outputs. Making these roles explicit helps isolate which inductive biases are responsible for which behaviours, thereby enabling more rigorous study of architectural choices. In this sense, modularity is valuable not because it showcases an implementation, but because it supports theory-driven comparison while preserving end-to-end differentiability across the whole pipeline.

6 Implementing SmartVille

This section presents the open-source implementation of SmartVille as a proof-of-concept instantiation of the framework. The goal is not to elevate one software stack as definitive, but to show that the methodological assumptions developed so far can be operationalised in a reproducible research environment. Accordingly, the implementation should be read as one concrete realisation of the framework's requirements—online learning, multi-modal observation, and continual incorporation of new threat intelligence—rather than as the primary contribution of the paper. Figure 3 illustrates the technological stack used for this instantiation and the main components involved in the NID research lifecycle.

6.1 GNS3: Realistic Network Emulation

As discussed in §4, effective training and evaluation of online NID systems require environments that closely mirror real-world network dynamics. Discrete event simulators such as NS3 often abstract away low-level system behaviours, limiting their fidelity for intrusion detection research. In contrast, SmartVille leverages GNS3 [71], an open-source network emulation platform that supports virtualization of real operating systems, networking devices, and hardware configurations. This enables high-fidelity emulation of network topologies with accurate timing, protocol behaviour, and system-level interactions. Note that GNS3 has also been chosen in recent literature [72–74].

SmartVille integrates with the GNS3 API to automate the deployment and configuration of user-defined network topologies. Such an integration includes attack nodes, honeypots, monitoring switches, and victim systems, all containerised and interconnected via virtual links. The framework reads a declarative configuration file (in the YAML format) to instantiate the desired topology, pre-load traffic patterns,

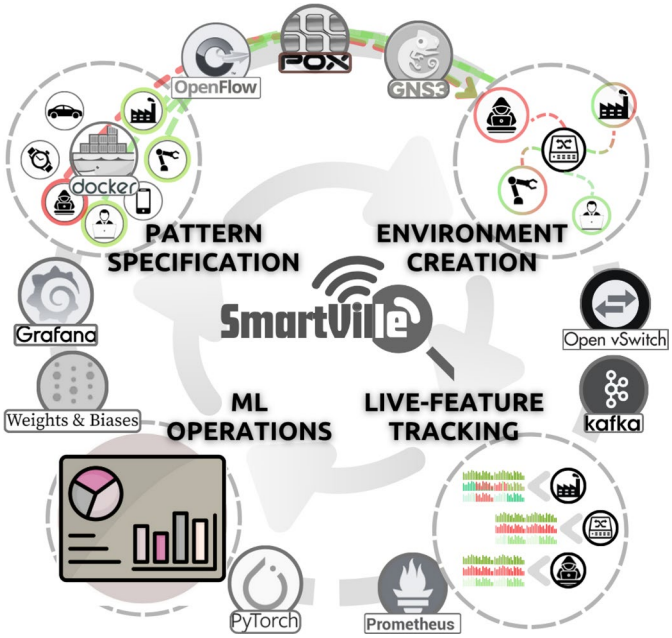

Fig. 3 Architectural overview of the SmartVille implementation. Network emulation is based on GNS3, with seamless monitoring enabled by *OpenFlow*, *Open vSwitch*, and *POX*. Node-level feature collection is performed using message-queue middleware such as *Kafka* and *Prometheus*. The deep learning backend uses *PyTorch*, while ML-Ops are streamlined via integration with *Weights&Biases* and *Grafana*

and initiate monitoring agents. This automation ensures reproducibility and scalability across experiments.

While GNS3 provides high-fidelity protocol, forwarding, and system-level emulation, it is important to acknowledge its boundaries regarding strict IoT hardware representation. Specifically, the framework abstracts the physical wireless channels and does not natively emulate the extreme memory or computational bounds of bare-metal microcontrollers. The scaling limits of SmartVille are bound by the host server's CPU and RAM capacity. For deep hardware-level validation–such as profiling thermal dynamics, power consumption, and swap memory limits on Raspberry Pis or MCUs, as demonstrated by [8] and [44]–the models developed within SmartVille should ultimately be transitioned to physical testbeds. However, for the purpose of this framework–conceiving the joint stream-processing architecture, feeding multimodal tensors, and benchmarking the differentiable EPD neural pipeline–GNS3 provides the necessary controllable and scalable environment.

6.2 Software-Defined Networking for Online Monitoring and Labelling

To support online feature extraction and dynamic labelling—which are essential under the proposed open-world formulation—the current implementation adopts a Software-Defined Networking (SDN) architecture. SDN is used here because it

offers a programmable way to expose traffic observations and labels at runtime; it is not meant to imply that SDN is the only acceptable deployment choice. In this instantiation, *Open vSwitch* (OVS) containers serve as Layer 2 forwarding elements within the GNS3 topology, while the *POX* controller acts as an SDN controller, implementing OpenFlow [75] rules to manage traffic. These rules use IP and port matching for routing purposes and enable two key capabilities:

- **Selective packet sampling:** High-priority OpenFlow rules are dynamically installed to redirect a subset of flows to user space for deep inspection, minimising overhead while preserving access to raw payload data. The first n bytes of sampled packets, where n is a user-specified parameter, are extracted, and headers are anonymised to avoid using this information a a discriminative feature in the classification stage. An analysis of the sampling interval adopted in our proof-of-concept implementation is reported in Appendix Appendix B. Importantly, those results should be read as characterising the present implementation and traffic setting, rather than as establishing an absolute optimal sampling period for the framework in general.
- **Flow statistics collection:** The POX controller queries OVS counters every 5 s to gather per-flow statistics, including packet count, byte count, and flow duration. These are aggregated into time-series tensors for use in the NID pipeline.

By combining SDN with dynamic labelling, SmartVille ensures that each observed flow can be associated with its ground-truth traffic pattern in real time, enabling both supervised classification and C-AD during training.

6.3 On-Demand Traffic Generation

To support online learning and dynamic curriculum scheduling, the implementation includes a *Traffic Manager* module that orchestrates the replay of attack and benign traffic patterns from pre-loaded `.pcap` traces. This choice is motivated by experimental controllability: replayed traces make it possible to trigger specific behaviours at precise times, which is useful when studying curriculum composition, distribution shift, and dynamic labelling. The system is initialised with the *Aposemat IoT23* dataset [76], which includes diverse IoT-oriented malware such as *Mirai*, *Gafgyt*, *Hakai*, *Torii*, and *Hajime*, among others.[1] Each pattern is assigned to a dedicated attacker or honeypot node, allowing independent triggering.

Traffic generation is fully programmable: users can specify which patterns to activate, their start time, duration, and frequency through either the WebGUI or configuration files. This capability is crucial for simulating concept drift, zero-day attacks, and evolving threat landscapes, which represent the hallmarks of the open-world setting.

[1] Original .pcap files are publicly available at https://www.stratosphereips.org/datasets-iot23.

Additionally, node-level system metrics (e.g., CPU usage, memory consumption, and interface load) are collected via lightweight agents and streamed through *Kafka*[2] and *Prometheus*,[3] forming a parallel input stream for multi-modal feature fusion.

6.4 Dynamic Experiment Configuration via WebGUI

To facilitate accessibility and reproducibility, SmartVille provides a Flask-based WebGUI that extends the native GNS3 interface with NID-specific controls. As shown in Fig. 4, the interface allows users to configure key aspects of the experiment lifecycle:

- Selection of input modalities (e.g., flow statistics, raw traffic, node features),
- Definition of the pattern partition $\mathscr{P}_K$, $\mathscr{P}_U^{\text{train}}$, $\mathscr{P}_U^{\text{test}}$ (as in Equation (2)),
- Specification of time-series window length, sampling frequency, and feature dimensions,
- Selection and parametrisation of neural architectures in the EPD pipeline.

The WebGUI also enables real-time monitoring of network activity, traffic generation status, and system health.

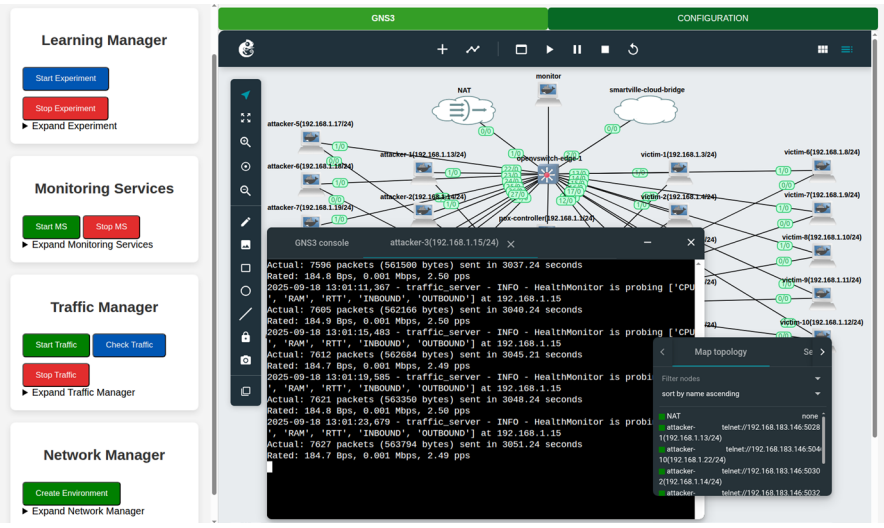

Fig. 4 The SmartVille WebGUI for configuring and managing online NID experiments. The interface supports dynamic scenario generation, monitoring setup, on-demand traffic triggering, and live training and evaluation of deep learning models

[2] https://kafka.apache.org/.

[3] https://prometheus.io/

6.5 Machine Learning Operations (MLOps)

To align with modern research practices, SmartVille integrates with *Weights&Biases* (*W&B*) for experiment tracking, interpretability, and reproducibility. Upon experiment launch, all hyperparameters, architecture choices, and dataset configurations are automatically logged. During training, performance metrics for every task are automatically measured and streamed to a centralised *W&B* dashboard. These metrics include Accuracy for supervised classification and anomaly detection, and ARI/NMI for supervised clustering or adjacency regression.

Additionally, the framework generates and uploads interpretability artifacts such as:

- UMAP visualizations of latent embeddings,
- predicted vs. ground-truth clusters,
- prototype evolution over time,
- training and evaluation confusion matrices for classification and anomaly detection.

Such features support transparent model analysis and comparative benchmarking across architectural variants. An example dashboard is publicly accessible at https://wandb.ai/jfcevallos/SMARTVILLE_2.0.

All code, configurations, and experiment logs are version-controlled and available in the public repository at https://github.com/DISTA-IoT/insubria-smartville, ensuri ng full reproducibility and extensibility.

7 Use Case Studies

This section demonstrates how the SmartVille framework enables reproducible experimentation for deep learning-based NID under realistic online and open-world conditions. Three use cases are presented, with the aim of exemplifying the following key research axes supported by the framework: (1) variation of meta-training curricula to assess generalisation in collective anomaly detection; (2) ablation studies on multi-modal input-space composition; (3) modular benchmarking of neural architectures within the EPD pipeline. All experiments were conducted on a containerised network topology comprising 20 nodes emulated in GNS3 (version 2.2.54) [71]. To maintain broad applicability, the emulated nodes communicate over a standard TCP/UDP/IP protocol stack over Ethernet, bypassing IoT-specific physical layer abstractions. Traffic is orchestrated via pre-loaded `.pcap` traces from the Aposemat IoT23 dataset [76] (dataset's content will be better detailed along with the description of the case studies). The NID system is deployed on a central monitoring switch in a star topology, enabling full visibility of inter-node communications. Complete experimental configurations, hyperparameters, and code are available in the public repository.[4]

[4] https://github.com/DISTA-IoT/insubria-smartville.

Before detailing the experiments, it is worth stressing the intended role of this section. The three use cases should be read explicitly as *baseline studies* enabled by the SmartVille framework, not as evidence that the paper proposes or endorses a single fixed neural pipeline as the final solution to NID. Their purpose is to show how the framework supports agile, theory-driven evaluation of alternative design choices under controlled online conditions: changing the meta-training curriculum, changing the input-space composition, and changing the neural modules within the same end-to-end differentiable organisation. In other words, the contribution of SmartVille lies in the research formulation and experimental methodology that make these comparisons possible, while the specific pipelines instantiated here serve as representative baselines within that broader theoretical framework.

7.1 Motivating Scenario and Threat Model

A useful way to interpret the role of SmartVille is to consider a small organisation or smart-building network in which heterogeneous devices—e.g., workstations, cameras, smart lamps, voice assistants, and service nodes—communicate through a monitored access switch. In this scenario, the network operator does not control the external adversary, but does control the local monitoring infrastructure and can continuously observe traffic and host-level side effects. The practical objective is not only to recognise attacks already present in historical data, but also to detect suspicious behaviours that were not explicitly labelled at deployment time, while the network remains operational.

Under this interpretation, the attacks of interest originate from *external or already-compromised end hosts* and manifest through the traffic exchanged with honest infrastructure. Representative examples include malware attempting command-and-control communication (e.g., *Torii* or *Okiru*), worms and botnets propagating across vulnerable devices (e.g., *Muhstik* or *Mirai*-like behaviour), horizontal scanning activity, and generic DDoS-related traffic patterns. These attacks are precisely the kind of behaviours reproduced in the use cases through the IoT23 traces described later in this section and in Appendix A.

The corresponding threat model is intentionally simple. We assume that the monitored switch, the telemetry-collection logic, and the orchestration components of the framework operate *honestly* and are not themselves compromised by the attacker. In other words, the adversary is not the switch, not the controller, and not the framework's management plane; rather, the adversary is an external actor, or malware acting through infected endpoints, whose effects become observable in packet streams, flow statistics, and node-level measurements. This trust assumption is coherent with the scope of the paper, which is to study online multi-modal intrusion detection once trustworthy observations are available, rather than to solve secure control-plane design or Byzantine monitoring.

Regarding capabilities, the attacker may generate malicious traffic, contact remote command-and-control servers, scan the local network, exploit vulnerable hosts, and change behaviour over time so that some traffic patterns seen at test time are absent from the labelled training set. This is exactly why the framework is formulated in open-world terms. By contrast, the attacker is not assumed to have arbitrary control

over the defender's labels, to tamper with the monitoring stack, or to forge perfect ground-truth annotations inside the framework. From the defender's perspective, the network topology and telemetry channels are known, but the precise future attack family, its timing, and its concrete behavioural signature may be unknown before deployment.

This motivating scenario also clarifies how an administrator would use SmartVille in practice. The monitoring infrastructure collects the three modalities discussed earlier, the deployed model processes them online, and the system can then support two complementary actions: flagging traffic associated with already-known malicious classes, and highlighting groups of traffic whose joint behaviour is novel with respect to what the model was trained to recognise. The use cases below should therefore be read as controlled studies of that operational setting: how different curricula affect adaptation to unseen behaviours, how different modalities contribute to detection, and how alternative neural modules change performance under the same threat assumptions.

This point is especially important for interpreting the open-world results. SmartVille is *not* introduced as a single proprietary detector that must itself establish universal superiority in recognising unknown attacks. Rather, it is a principled *framework* for formulating, training, and evaluating candidate NID pipelines under online, multi-modal, and open-world assumptions. Accordingly, the evidence reported in the following use cases should be read as showing how the framework can be used to benchmark different design choices and study their generalisation behaviour on held-out attack families under a common protocol. SmartVille is not a fixed algorithm. Consequently, the relevant question is not whether it demonstrates superiority to any other algorithm, but whether it provides a coherent and reproducible methodology for testing how different algorithms, curricula, modalities, and neural operators behave when genuinely unseen attacks arise after deployment.

7.2 Use Case 1: Meta-Training Curricula Baselines

The meta-learning formulation of C-AD, introduced in §3.2.3, relies on a partitioned pattern space $\mathscr{P} = \mathscr{P}_K \cup \mathscr{P}_U^{\text{train}} \cup \mathscr{P}_U^{\text{test}}$, where models learn to cluster unknown anomalies during training and generalise to held-out OOD patterns at test time. The choice of pattern assignment to each subset critically influences model generalisation, depending on both domain expertise (e.g., threat maturity) and statistical divergence between clusters.

In this use case, the impact of three distinct training curricula on C-AD performance is evaluated using the Adjusted Rand Index (ARI) as the primary metric for kernel regression accuracy. The IoT23 dataset provides 13 distinct traffic patterns, including benign home-IoT related devices such as Amazon Echo and attacks from DDoS families (*Mirai*, *Gafgyt*, *Hakai*), command-and-control (C&C) malware (*Torii*, *Okiru*), and IoT-specific worms (*Hajime*, *Muhstik*). Table 1 summarises the pattern types and extracted traffic counts reported in Appendix A.

For this use case, the patterns were partitioned as follows:

1. **Curriculum A:**

Table 1 IoT23 patterns used in our experiments. (See Appendix A for details)

Pattern	Type	Count
Hajime	Trojan/worm	50k flows
Hakai	DDoS botnet	12k flows
Gafgyt	DDoS botnet	50k flows
Mirai	DDoS botnet	24k flows
Torii	C&C malware	50k flows
Muhstik	Worm	50k flows
Okiru	C&C botnet	50k flows
H-Scan	Horizontal scan	50k flows
C&C HB	C&C heartbeat	1.5k flows
Gen-DDoS	Generic DDoS	50k flows

$$\mathscr{P}_K := \{\text{Hue, Hakai, Torii}\}$$
$$\mathscr{P}_U^{train} := \{\text{Okiru, C[NONSPACE] \& C HB, Gen-DDoS}\}$$
$$\mathscr{P}_U^{test} := \{\text{Mirai, Gafgyt, Hajime, H-Scan, Muhstik, Doorlock, Echo}\}$$

2. **Curriculum B:**

$$\mathscr{P}_K := \{\text{Hue, Hakai, Torii}\}$$
$$\mathscr{P}_U^{train} := \{\text{Mirai, Gafgyt, Hajime}\}$$
$$\mathscr{P}_U^{test} := \{\text{Okiru, C[NONSPACE] \& C HB, Gen-DDoS, H-Scan, Muhstik, Doorlock, Echo}\}$$

3. **Curriculum C:**

$$\mathscr{P}_K := \{\text{Hue, Hakai, Torii}\}$$
$$\mathscr{P}_U^{train} := \{\text{H-Scan, Muhstik, Doorlock}\}$$
$$\mathscr{P}_U^{test} := \{\text{Mirai, Gafgyt, Hajime, Echo, Okiru, C[NONSPACE] \& C HB, Gen-DDoS}\}$$

Each curriculum maintains the same known classes ($\mathscr{P}_K$) for supervised classification, while varying the distributional overlap between $\mathscr{P}_U^{\text{train}}$ and $\mathscr{P}_U^{\text{test}}$. Fig. 5 reports the results of this use case aggregated over five random seeds, capturing both central tendency and run-to-run variability for classification Accuracy, anomaly detection Accuracy, and kernel-regression ARI. The figure shows that all three curricula reach strong and comparatively stable performance on the directly supervised portions of the task, whereas the most informative separation emerges on the held-out $\mathscr{P}_U^{\text{test}}$ classes. In particular, the OOD clustering metric discriminates the curricula much more clearly than the in-distribution metrics, which remain high across settings.

The five-seed plots confirm that curriculum design materially affects OOD collective anomaly detection. Curriculum B attains the strongest and most sustained ARI on $\mathscr{P}_U^{\text{test}}$, while curricula A and C remain more limited despite achieving strong

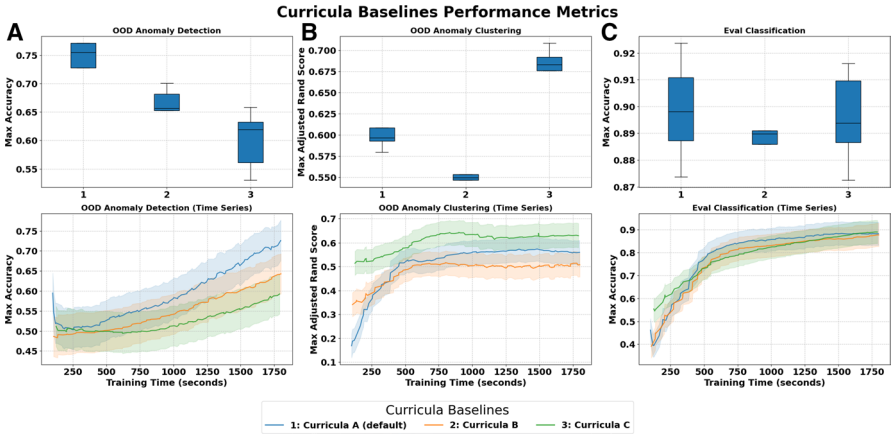

Fig. 5 Use Case 1 baseline comparison for meta-training curricula A, B, and C. The figure reports the evolution of classification and collective-anomaly-detection metrics under the three curriculum choices, with each curve summarising five runs with different random seeds. This representation highlights both average behaviour and run-to-run variability, making it possible to compare curricula not only in terms of final scores but also in terms of stability during online training and evaluation

supervised and in-training anomaly scores. This gap is important because it shows that success on the directly optimised components of the task does not automatically translate into better clustering of genuinely unseen behaviours. A plausible explanation is that exposing the model during meta-training to prototypical DDoS-like structures (here, *Mirai*, *Gafgyt*, and *Hajime*) induces a more transferable relational bias for the held-out anomaly families than the alternatives used in curricula A and C. More generally, Fig. 5 supports the claim that curriculum selection is not a secondary implementation detail, but a first-class design variable for open-world NID research within SmartVille.

7.3 Use Case 2: Input-Space Baselines

To evaluate the contribution of multi-modal features to detection performance, an ablation study on input-space composition is conducted. Three feature domains are considered:

- **Flow Statistics:** They are extracted every 5 s via OpenFlow port stats, including packet count, byte count, and flow duration; they are also aggregated into 10-timestep sliding windows.
- **Raw Traffic Bytes:** The first 64 bytes of randomly sampled packets are captured by dynamically activating high-priority OpenFlow rules for 1-second intervals. Headers are anonymised, and payloads are stacked into 10-timestep sequences.
- **Node-Level Features:** System metrics (i.e., inbound and outbound interface load) are collected every 5 s from source and destination nodes via lightweight agents, forming parallel time-series inputs.

Three configurations are evaluated: (i) flow statistics + node features, (ii) flow + raw traffic, and (iii) all three modalities. Fig. 6 reports the results over five random seeds. The aggregated plots confirm that modality choice has a measurable impact across both the classification and collective-anomaly-detection components of the framework. In particular, the full three-modality configuration provides the strongest overall behaviour, indicating that packet-level evidence and node-level measurements contribute complementary information beyond flow statistics alone. The figure also shows that the two bimodal settings are not equivalent: the combination of flow statistics with node features remains consistently competitive and, in the reported runs, stronger than the flow-plus-raw-traffic alternative on the main downstream metrics. This suggests that host-side measurements can supply stable contextual cues that remain useful even when packet samples are sparse or only partially informative.

All parameters, including window length, sampling frequency, and byte depth, are configurable via the WebGUI or YAML files, enabling rapid exploration of trade-offs between monitoring overhead and detection efficacy.

To better understand the contribution of each modality and map specific attack behaviours to their most discriminative features, Fig. 7 reports the aggregated confusion matrices across the tested modality configurations. The visual comparison clearly illustrates the complementary roles of the input spaces. Relying solely on flow features (right) struggles to cleanly isolate complex Command and Control (C&C)

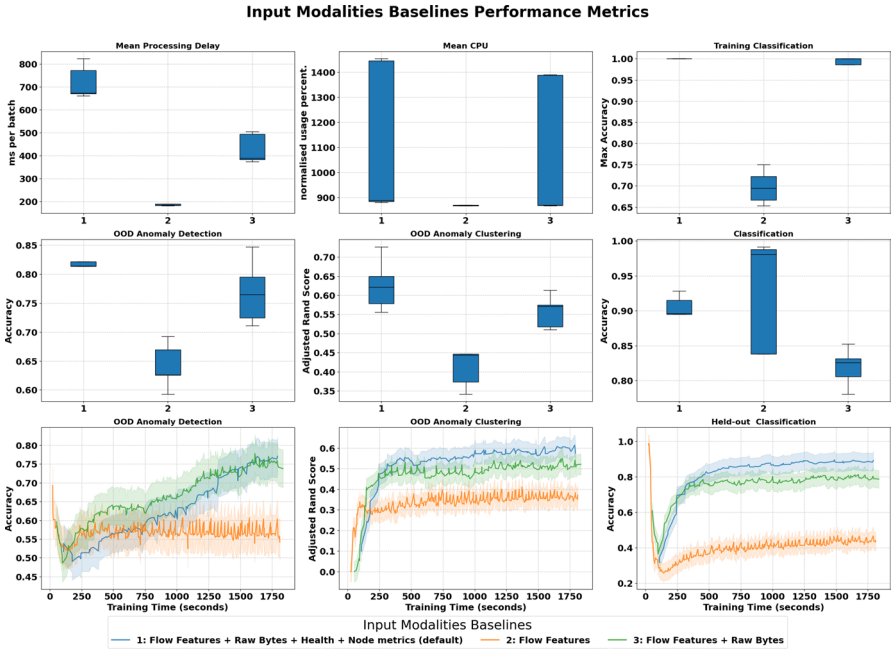

Fig. 6 Use Case 2 baseline comparison for input-modality configurations. The figure reports the evolution of classification and collective-anomaly-detection metrics for three input compositions: flow statistics plus node features, flow statistics plus raw traffic, and the full multi-modal setting. Each curve summarises five runs with different random seeds, the plot makes it possible to compare not only mean performance but also the stability of each modality choice during online training and evaluation

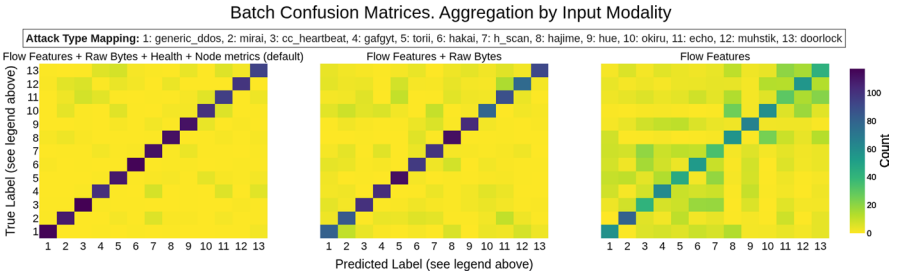

Fig. 7 Aggregated confusion matrices for varying input-space compositions over five random seeds. The comparison highlights how the full multi-modal configuration (left) yields the sharpest class separation compared to using only flow and raw bytes (middle) or flow features alone (right). Classes 1-13 correspond to the IoT23 traffic patterns detailed in the figure legend

behaviours like *Torii* (class 5) and *Okiru* (class 10), as well as benign device traffic like *Echo* (class 11) and *Doorlock* (class 13). However, it retains a baseline discriminative signal for volumetric attacks like *Mirai* (class 2) and *Generic DDoS* (class 1), where flow frequency is naturally more revealing than payload content.

The introduction of sampled raw traffic bytes (middle) drastically sharpens the diagonal, particularly for the aforementioned C&C malware and IoT botnets. This highlights the value of the raw-byte modality: because many legacy IoT attacks rely on unencrypted handshakes or highly predictable application-layer structures, the neural encoder easily captures their byte-level signatures. However, this reliance may represent a vulnerability: if an attacker were to utilize fully encrypted payloads or advanced obfuscation, selective byte sampling would yield limited discriminative value, rendering the payloads potentially indistinguishable from encrypted benign traffic (such as the TLS-based communications of the *Hue* and *Echo* devices).

This reality is precisely what justifies the full multi-modal configuration (left). By integrating node-level telemetry (e.g., CPU load, interface saturation) alongside flow statistics and raw bytes, the model achieves the strongest and most stable separation. In an operational open-world setting where payload inspection may suddenly fail due to encryption, the end-to-end differentiable nature of the pipeline allows the neural processor to organically shift its attention toward the structural flow patterns and host-side side effects. This analysis confirms that multi-modal fusion in SmartVille is not merely an arbitrary performance booster, but a structural necessity for maintaining detectability against evolving, obfuscated threats.

7.4 Use Case 3: Neural Architecture Baselines

The modular EPD pipeline in §5 supports plug-and-play experimentation with diverse neural architectures. This use case evaluates four variants of the encoder and processor modules, trained using the same dataset and split, namely curricula A as defined in §7.2, and using all the input modalities defined in §7.3. To study this design space, different relational operator baselines and loss functions for the anomaly-clustering stage are benchmarked:

- The default operator is a Similarity Network [77] that takes the L1 distance be-

tween two latent representations and outputs an association score. We refer to this operator as the Distance Kernel Regressor (DistanceKR).

- A lighter alternative applies a learned transformation to the inner product between latent representations. We refer to this operator as the Dot-Product Kernel Regressor (DotProdKR).
- The kernel-regression task requires a loss that penalises incorrect associations. Our default choice explicitly separates attractive and repulsive terms, following the spirit of manifold-learning objectives such as UMAP [78]. This formulation helps preserve both local neighbourhood structure and global separation:

$$\mathcal{L}_{\text{UMAP}} = \frac{1}{N}\sum_{i=1}^{N}[w_a \sum_{j} -B_{ij}\log(P_{ij}+\epsilon)+ w_r \sum_{j} -(1-B_{ij})\log(1-P_{ij}+\epsilon)]$$

Here, B denotes the reference kernel, P the predicted kernel, and w_a, w_r the attractive and repulsive weights, respectively. This objective is rooted in the fuzzy-set cross-entropy formulation used in topological data analysis.

- As a simpler alternative, we also consider a weighted Binary Cross-Entropy (BCE) objective. In this case, kernel regression is treated as an element-wise probabilistic alignment problem:

$$\mathcal{L}_{BCE} = -\frac{1}{N \cdot M}\sum_{i,j} W_{ij}[B_{ij}\log(P_{ij}) + (1-B_{ij})\log(1-P_{ij})]$$

In this formulation, the weight matrix W is assigned dynamically: $W_{ij} = w_a$ when $B_{ij} \approx 1$ and $W_{ij} = w_r$ when $B_{ij} \approx 0$, following standard cost-sensitive practices for similarity learning [79].

Fig. 8 contains a five-seed baseline comparison of using the different neural operators listed above. The repeated runs preserve the same qualitative message while making the variability explicit: operator choices within the EPD pipeline do affect performance, but the comparison remains stable across random initialisations. In our specific setting, the DistanceKR variants generally show faster and more consistent convergence, especially for the collective anomaly-detection metrics, whereas the DotProdKR variants appear more variable and typically settle at lower clustering quality. The comparison between the two kernel-regression losses is more nuanced: both objectives remain viable baselines, but the UMAP-inspired objective appears to provide the most robust behaviour for anomaly grouping, while the weighted BCE objective remains a simpler alternative that is still competitive in parts of the train-

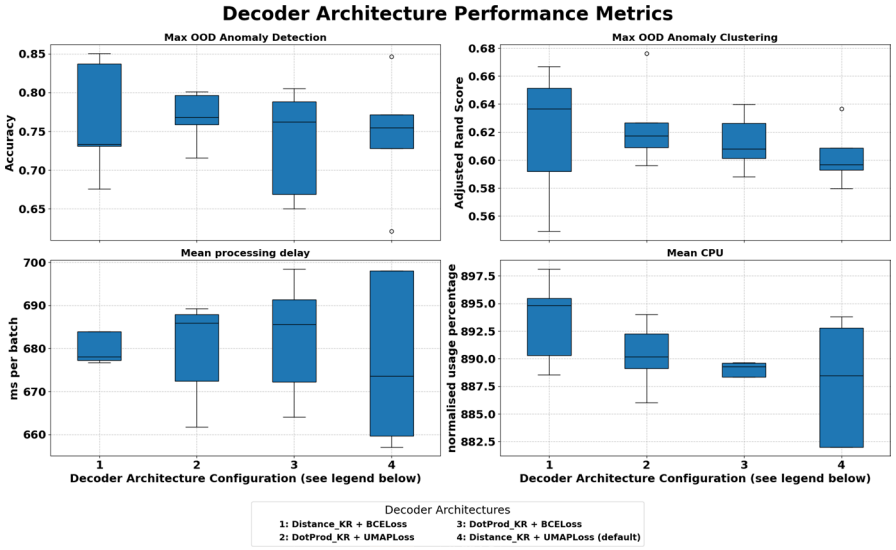

Fig. 8 Use Case 3 baseline comparison under the same data split, input modalities, and training protocol. The figure reports the evolution of classification and collective-anomaly-detection metrics for different neural operator baselines obtained by combining the Distance Kernel Regressor (DistanceKR) or the Dot-Product Kernel Regressor (DotProdKR) with either the UMAP-inspired kernel-regression loss or the weighted BCE loss. Each curve summarises five runs with different random seeds, so the plot highlights both average behaviour and run-to-run variability

ing trajectory. Overall, the point of Fig. 8 is therefore methodological: SmartVille makes it possible to benchmark relational operators and training objectives under identical online conditions, and the five-seed results show that these trends are not an artifact of a single lucky run.

These findings underscore the importance of aligning architectural choices with the desired inductive biases, in particular concerning relational invariance and geometric separability, in open-world NID. The SmartVille framework enables such comparisons in a standardised, end-to-end differentiable setting, facilitating evidence-based design of neural intrusion detectors.

8 Conclusions and Future Work

SmartVille presented a unified, end-to-end differentiable framework for deep learning–based NID that addresses the key challenges of online learning, open-world dynamics, multi-modality, and deployability. Rather than proposing a single detector to be judged only by point accuracy, SmartVille is as a way of *conceiving*, studying, and benchmarking the NID lifecycle under realistic operating assumptions. Its design is grounded in two core principles: the **relational bottleneck** as an architectural inductive bias, and a prototypical meta-training paradigm for generalisation under distribution shift.

The relational bottleneck shifts inference from absolute feature matching to geometric relationships in latent space, thus enabling data-efficient, robust detection through prototypical learning and kernel regression. This supports both the classification of known attacks and the collective anomaly detection (C-AD) of heterogeneous, concurrent threats. Meanwhile, the ASAP curriculum formalises C-AD as a meta-learning task, training models to recognise clustering principles rather than fixed patterns, thereby improving generalisation to unseen anomalies.

Beyond algorithmic design, SmartVille integrates realistic system-level components: dynamic labelling, on-demand traffic generation via GNS3, multi-modal feature extraction, and SDN-based monitoring. These enable reproducible emulation of evolving network environments and, more importantly, provide a concrete proof of concept for the proposed machine-learning framework. In this sense, the open-source implementation, including a WebGUI and MLOps integration, is meant to support standardised benchmarking and deployment-oriented research, rather than to claim a definitive or universally optimal engineering solution.

Future work will extend the framework in several directions. First, the current on-demand traffic generation, based on replay of `.pcap` traces, can be enhanced with active execution of malicious software within emulated nodes, or with physical testbed deployments, to improve realism and capture runtime behavioural dynamics that replay alone may miss. Second, support for distributed and federated learning will enable evaluation of collaborative detection across network segments under privacy constraints. Third, integration of adversarial training mechanisms will allow robustness assessment against evasion attacks, while future studies should also broaden the threat model to include compromised monitoring components and telemetry tampering. Fourth, a more operational evaluation of the framework will explicitly incorporate metrics such as Mean Time To Detect (MTTD), which is particularly relevant in the online setting considered in this work. Finally, we plan to incorporate lightweight model compression techniques, such as pruning, quantisation, and knowledge distillation, to further optimise the framework for edge deployment. Such extensions will strengthen SmartVille as a platform for next-generation, operationally grounded NID research.

Appendix A: On the Traffic Patterns Included in Our Open-Source Implementation of SmartVille

The Aposemat IoT23 dataset [18] is preloaded in SmartVille to reproduce realistic IoT attack and honeypot patterns. More specifically, the following attacks were extracted from the original *.Pcap* files that are publicly available.[5] Unless explicitly stated otherwise, these flows were extracted using the attacker origin IP address as the filtering criterion. Recall that source and destination IP addresses and transport-layer ports are online masked before feeding the neural modules with raw packet bytes.

[5] https://www.stratosphereips.org/datasets-iot23.

1. **Hajime:** This Trojan malware searches to exploit Linux-related vulnerabilities. 5e4 of such flows were extracted from the dataset's capture 9.1.
2. **Hakai:** This is a distributed denial-of-service (DDoS) botnet, a specialization of the Mirai malware. (Extracted from dataset's capture 8.1.) 1.2e4 flows were extracted.
3. **Gafgyt:** This is another more general DDoS botnet. (Dataset's capture 60.1.) 5e4 flows extracted.
4. **Mirai:** This is an open-source DDoS attack specially used over IoT devices. Capture: 34.1. Flows: 2.4e4.
5. **Torii:** A Command and Conquer (C&C) and Information Gathering malware. Capture: 20.1. Flows extracted: 5e4.
6. **Muhstik:** A worm based on the Mirai Botnet. Among others, it targets IoT devices. Commonly used to mine cryptocurrency and perform DDoS attacks. Capture: 3.1. Flows extracted: 5e4.
7. **Okiru:** Another C&C Botnet that targets ARC processors, commonly used in wearables, and medical IoT, among others. Capture 7.1. Flows extracted: 5e4. The criteria for extracting these attack flows included the target source and destination transport ports.
8. **Horizontal Scan:** 5e4 traffic flows related to generic Horizontal Scan (HScan) were extracted from the dataset's capture 1.1.
9. **C&C HeartBeat:** Generic heartbeat-related server-side flows were also extracted in the context of the C&C traffic. Capture 7.1. Flows extracted: 0.15e4.
10. **Generic DDoS:** Also, a set of 5e4 generic DDoS-related flows was extracted in the context of the C&C traffic from capture 7.1.

The Honeypot devices used by the authors of the IoT23 captures were used in the experiments to emulate honeypot IoT devices used as attack victims. More specifically, these honeypots were the following:

1. **Somfy door lock device:** All the flows contained in the first three captures of folder Honeypot7.1 were extracted. These flows are related to a smart door lock device. In the implemented topology, two nodes reproduced these flows.
2. **Philips HUE smart LED lamp:** These flows were extracted from the folder Honeypot4.1, and reproduced by a unique node.
3. **Amazon Echo home intelligent personal assistant:** These flows were extracted from the folder Honeypot5.1. One node reproduced these flows.

The interested reader is referred to [18] for more details on the dataset. All the flow extraction code used for the experiments is available in the form of a GitHub Gist.[6]

[6] https://gist.github.com/QwertyJacob/e33e68f230d5ecec8b6f2524ab166ebf.

Appendix B: Sampling Interval Analyses

Figure 9 reports a micro-benchmark to clarify the sampling-period choice adopted in the current proof-of-concept implementation of SmartVille. The figure should be read as an overhead utility analysis of one particular implementation choice, rather than as a claim that the framework intrinsically depends on synchronous SDN sampling at a fixed interval. A 15-second period has been adopted in all our experiments as it provides a satisfactory operating point between predictive utility and monitoring cost, while remaining compatible with the online, stream-oriented setting targeted by the paper.

The first column reports *evaluation* metrics, not training metrics. This distinction is especially important for the collective anomaly detection setting: the corre-

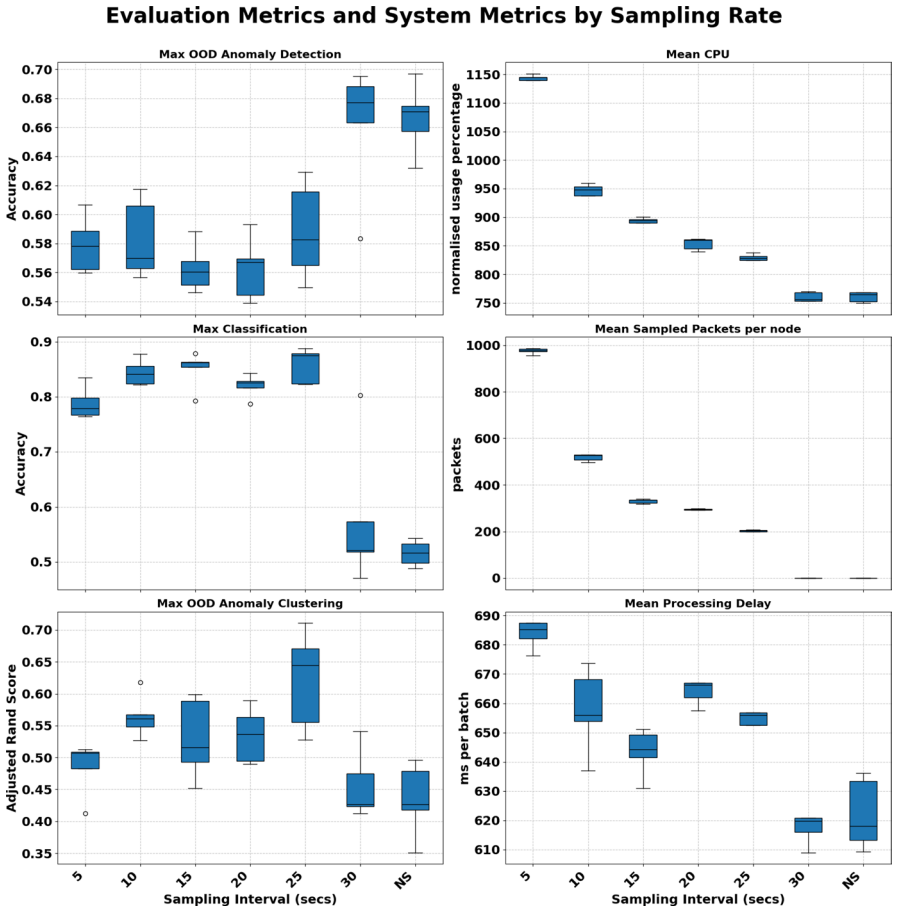

Fig. 9 Statistical analyses for different sampling intervals in our prototypical implementation of the proposed SmartVille framework. 'NS' abbreviates 'no_sampling', 'OOD' abbreviates 'Out-of-Distribution'

sponding scores are computed on held-out *classes*, i.e., genuinely out-of-distribution (OOD) traffic patterns, rather than only on held-out samples from already-observed classes. From this perspective, the plots indicate that moving from the previous 5-second configuration to a 15-second configuration does not materially degrade the framework's ability to generalise to unseen traffic patterns, while reducing the monitoring frequency.

The second column characterises the cost of the sampling policy itself. The number of sampled packets exhibits limited variance across the non-degenerate configurations, showing that the controller does not produce unstable traffic volumes when the probing interval is increased moderately. By contrast, the 30-second case often yields zero sampled packets under the current traffic patterns. This behaviour is not evidence of a general failure of the framework, but a consequence of transport-layer communication behaviour of nodes and forwarding-rule management in the switch: when the probing interval becomes too sparse relative to the lifetime of the active flows, no packets remain available for the sampling rule at the observation instant. This confirms that excessively long intervals can make the raw-traffic modality uninformative in this implementation.

Finally, the processing-delay plot should not be read as evidence that the framework would necessarily slow down an operational network. In the present prototype, inference is invoked through a blocking call, which intentionally exposes the end-to-end latency of this proof-of-concept stack for measurement purposes. The framework contribution, however, is the research formulation of online multi-modal NID, not the claim that inference must be executed synchronously in real deployments. In practical deployments, asynchronous or decoupled inference pipelines could be adopted without changing the theoretical role of the sampled observations. Thus, the relevant conclusion from Fig. 9 is narrower: within the current prototype, 15-second sampling implies a satisfactory tradeoff between control-plane activity, informativeness from packet observations, and stable OOD evaluation behaviour.

Scaling Limits of the Neural Pipeline

While the topological scaling limits of the GNS3-based network emulation are primarily bound by the host server's CPU and RAM (as discussed in §6-A), another critical dimension of scalability in online NID concerns the neural architecture itself. Specifically, when inference and training occur simultaneously over data streams without the benefit of offline multi-epoch pretraining, the capacity and depth of the neural modules become strict constraints. To characterize these limits, we conducted micro-benchmarks on the Encode-Process-Decode (EPD) pipeline, evaluating the system's behavior when scaling the hidden dimensionality and the depth of the temporal encoder.

Figure 10 reports the performance and computational overhead when scaling the hidden dimension size of the neural modules across four configurations: 100, 200 (the default used in previous use cases), 400, and 600. The computational cost scales predictably, with the mean processing delay per batch growing super-linearly from roughly 300 ms (dim 100) to over 3500 ms (dim 600). However, the most signifi-

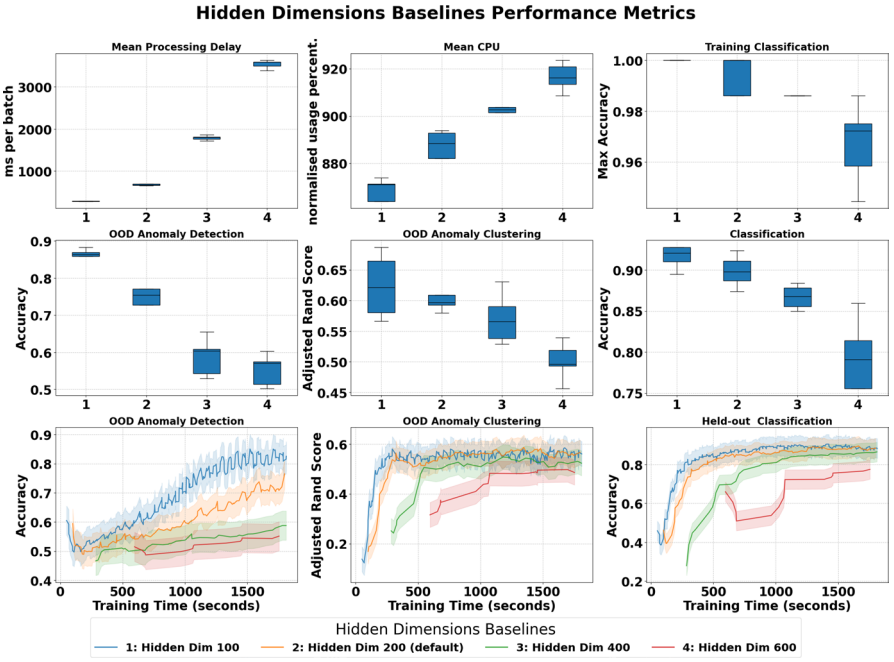

Fig. 10 Baseline performance metrics when scaling the hidden dimensions of the neural modules (100, 200, 400, and 600). The plots demonstrate that while computational delay and CPU usage increase significantly, the predictive performance across classification and OOD clustering tasks degrades with larger capacities due to online-learning instability

cant finding lies in the downstream learning metrics: increasing the hidden capacity strictly degrades performance across all tasks. Both the Out-Of-Distribution (OOD) anomaly clustering (ARI) and the hold-out classification accuracy drop significantly and exhibit higher variance as the dimension increases. This behavior highlights a fundamental property of online learning: without massive offline datasets and multiple epochs to stabilize gradients, highly parameterized models quickly overfit to the immediate streaming distribution, losing the plastic and generalizable representations required for open-world NID. Consequently, lightweight representations are not merely a computational necessity for edge devices, but an empirical requirement for stable online adaptation.

A similar trend is observed when scaling the depth of the recurrent temporal encoder. Figure 11 illustrates the framework's behavior when the sequence processing module is deepened from 1 (default) up to 4 recurrent layers. The processing delay increases linearly with depth, adding approximately 100-150 ms per batch per layer. More critically, deeper recurrent architectures struggle severely in the streaming setup. The hold-out classification accuracy drops from over 0.90 (with 1 layer) to nearly 0.70 (with 4 layers), and the anomaly clustering ARI shows a corresponding decline. Deep recurrent networks are notoriously difficult to train in single-pass, online settings due to vanishing gradients and the lack of randomized, bulk-shuffled

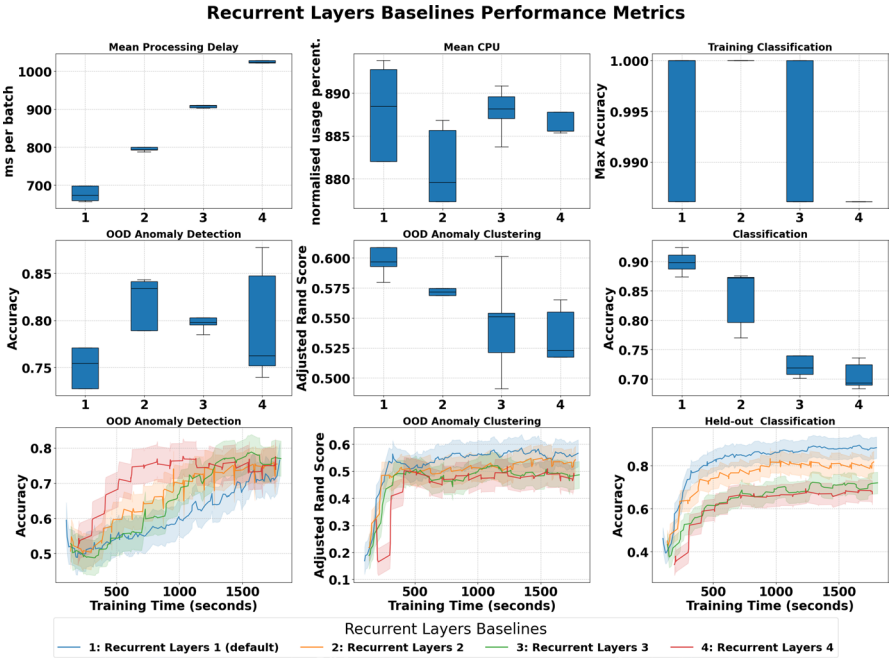

Fig. 11 Baseline performance metrics when scaling the depth of the recurrent temporal encoder (1, 2, 3, and 4 layers). The results indicate that deeper recurrent networks incur higher processing delays and fail to converge effectively under single-pass online training, yielding poorer classification and anomaly clustering scores compared to shallower variants

batches. The shallowest configuration converges much faster and maintains a more robust relational geometry for anomaly discovery.

In summary, these scaling benchmarks validate a core architectural philosophy of SmartVille: in realistic online and open-world NID deployments, larger models do not necessarily equate to better performance. Shallow, lightweight neural modules could adapt better to online-learning in open-world scenarios as they might provide a better balance between processing latency and the regularization needed to prevent catastrophic overfitting during continuous stream-based training.

Author Contributions All authors worked on the proposed solution, wrote and reviewed the manuscript.

Funding Open access funding provided by Università degli Studi dell'Insubria within the CRUI-CARE Agreement.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Rafique, S.H., Abdallah, A., Musa, N.S., Murugan, T.: Machine learning and deep learning techniques for internet of things network anomaly detection–current research trends. *Sensors*, 24(6):1968, 2024. Publisher: MDPI
2. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 20 (2019)
3. Guo, Y.: A review of machine learning-based zero-day attack detection: Challenges and future directions. Comput. Commun. **198**, 175–185 (2023)
4. Miani, R.S., Bernardo, G.D., Cassales, G.W., Senger, H., Faria, E R de.: A Survey of Data Stream-Based Intrusion Detection Systems. *IEEE Access*, 13:72953–72983, (2025) Publisher: IEEE
5. Di Monda, D., Montieri, A., Persico, V., Voria, P., De Ieso, M., Pescape', A..: Few-Shot Class-Incremental Learning for Network Intrusion Detection Systems. Accepted: 2024-11-06T12:15:46Z (2024)
6. Xu, C., Zhan, Y., Wang, Z., Yang, J.: Multimodal fusion based few-shot network intrusion detection system. Sci. Rep., 15(1):21986 (2025) Publisher: Nature Publishing Group UK London
7. Wang, Z., Shi, T., He, J., Cai, M., Zhang, J., Song, D.: CyberGym: Evaluating AI Agents' Cybersecurity Capabilities with Real-World Vulnerabilities at Scale, (2025). arXiv:2506.02548 cs
8. Naseh, D., Abdollahpour, M., Tarchi, D.: Real-world implementation and performance analysis of distributed learning frameworks for 6g iot applications. Information **15**(4), 190 (2024)
9. Chou, D., Jiang, M.: A survey on data-driven network intrusion detection. ACM Computing Surveys (CSUR) **54**(9), 1–36 (2021)
10. Mirkovic, J., Pusey, P.: User Experiences on Network Testbeds. In *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*, CSET '21, pages 72–82, New York, NY, USA, September 2021. Association for Computing Machinery
11. Goldschmidt, P., Chudá, D.: Network Intrusion Datasets: A Survey, Limitations, and Recommendations. Computers & Security **156**, 104510 (2025). arXiv:2502.06688 [cs]
12. Pospisil, O., Blazek, P., Kuchar, K., Fujdiak, R., Misurec, J.: Application Perspective on Cybersecurity Testbed for Industrial Control Systems. Sensors (Basel, Switzerland) **21**(23), 8119 (2021)
13. Judvaitis, J., Abolins, V., Elkenawy, A., Balass, R., Selavo, L., Ozols, K.: Testbed Facilities for IoT and Wireless Sensor Networks: A Systematic Review. J. Sens. Actuator Netw. **12**(3), 48 (2023). (**Publisher: Multidisciplinary Digital Publishing Institute**)
14. Elderman, R., Pater, L.J.J., Thie, A.S., Drugan, M.M., Wiering, M.A.: Adversarial reinforcement learning in a cyber security simulation. In 9th International Conference on Agents and Artificial Intelligence (ICAART 2017), pages 559–566. SciTePress Digital Library, (2017)
15. Microsoft Defender Research Team. Cyberbattlesim. https://github.com/microsoft/cyberbattlesim, 2021. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei
16. Pinto, D., Amorim, I., Maia, E., Praça, I.: A review on intrusion detection datasets: tools, processes, and features. Comput. Netw. **262**, 111177 (2025)
17. Panigrahi, R. Cicids2017, 2025
18. Garcia, S., Parmisano, A., Erquiaga, M.J.: Iot-23: A labeled dataset with malicious and benign iot network traffic (version 1.0.0), 2020
19. Nguyen H.P.T., Chen, Z., Hasegawa, K., Fukushima, K., Beuran, R.: Pengym: Pentesting training framework for reinforcement learning agents. In ICISSP, pages 498–509, (2024)

20. Janisch, J., Pevnỳ, T., Lisỳ, V.: Nasimemu: Network attack simulator & emulator for training agents generalizing to novel scenarios. In: European Symposium on Research in Computer Security, pages 589–608. Springer, (2023)
21. Hammar, K., Stadler, R.: Intrusion prevention through optimal stopping. IEEE Trans. Netw. Serv. Manage. **19**(3), 2333–2348 (2022)
22. Poisson, M., Carnier, R., Fukuda, K.: GothX: a generator of customizable, legitimate and malicious IoT network traffic. CSET @ USENIX Security Symposium, (2024). ARXIV_ID: 2407.07456 S2ID: 71a750e9babc8082053ffea9cd3b561c46c32685
23. Abou El Houda, Z., Moudoud, H., Brik, B., Adi, Ml.: A Privacy-Preserving Framework for Efficient Network Intrusion Detection in Consumer Network Using Quantum Federated Learning. IEEE Trans. Consum. Electron., (2024). S2ID: 6dceec61c248ff208e0c875e99985fe747289527
24. Rizzardi, A., Raffaele, D.C., Cevallos, M., Jessus, F., Simona, D.V., Vittorio, O., Sabrina, S., Domenico, C., Alberto, C.-P.: Open-FARI: An Open-source testbed for Federated Anomaly detection in the Railway IIoT. In *International Wireless Communications & Mobile Computing Conference (IWCMC 2025)*
25. Zhou, Q., Shi, C.: A Network Intrusion Detection Method for Various Information Systems Based on Federated and Deep Learning. *International Journal on Semantic Web and Information Systems (IJSWIS)*, (2024). S2ID: eca70ac709b9fa0564d562c8b407b7fe06626a64
26. Makridis, G., Theodoropoulos, S., Dardanis, D., Makridis, I., Separdani, M.M., Fatouros, G., Kyriazis, D., Koulouris, P.: XAI enhancing cyber defence against adversarial attacks in industrial applications. In 2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS), pages 1–8. IEEE, (2022)
27. Lopes Antunes, D., Llopis Sanchez, S.: The Age of fighting machines: the use of cyber deception for Adversarial Artificial Intelligence in Cyber Defence. In Proceedings of the 18th International Conference on Availability, Reliability and Security, pages 1–6, Benevento Italy, August (2023). ACM
28. Mao, J., Wei, Z., Li, B., Zhang, R., Song, L.: Toward Ever-Evolution Network Threats: A Hierarchical Federated Class-Incremental Learning Approach for Network Intrusion Detection in IIoT. IEEE Internet of Things Journal, (2024). S2ID: eb94951c3fcf26be000676f71598a99cd0794ac9
29. Kumar Amalapuram, S., Tadwai, A., Vinta, R., Channappayya, S.S., Reddy Tamma, B.: Continual learning for anomaly based network intrusion detection. In 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), pages 497–505. IEEE, (2022)
30. Ding, H., Sun, Y., Huang, N., Cui, X.: TMG-GAN: Generative Adversarial Networks-Based Imbalanced Learning for Network Intrusion Detection. *IEEE Transactions on Information Forensics and Security*, pages 1–1, January (2023). MAG ID: 4388505134 S2ID: 6953a2df9f5ffe2d3c699ca518d8716b65964ff3
31. Ganguli, D., Hernandez, D., Lovitt, L., DasSarma, N., Henighan, T., Jones, A., Joseph, N., Kernion, J., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Elhage, N., Showk, S.E., Fort, S., Hatfield-Dodds, Z., Johnston, S., Kravec, S., Nanda, N., Ndousse, K., Olsson, C., Amodei, D., Amodei, D., Brown, T., Kaplan, J., McCandlish, S., Olah, C., Clark, J.: Predictability and Surprise in Large Generative Models. In 2022 ACM Conference on Fairness Accountability and Transparency, pages 1747–1764, (2022). arXiv:2202.07785 [cs]
32. Arreche, O., Guntur,T., Abdallah, M.: XAI-IDS: Toward Proposing an Explainable Artificial Intelligence Framework for Enhancing Network Intrusion Detection Systems. Applied Sciences, (2024). S2ID: 6c9e74de76d15f7d3ad18eeaa31ebcad31da5751
33. Kumar, A., Thing, V.: Evaluating The Explainability of State-of-the-Art Deep Learning-based Network Intrusion Detection Systems. (2024). ARXIV_ID: 2408.14040 S2ID: be70fe228b50f075e7d4e85b56f4fa5d5e6c9d1d
34. Bouayad, A., Alami, H., Idrissi, M.J., Berrada, I.: Lightweight Federated Learning for Efficient Network Intrusion Detection. IEEE Access, (2024). S2ID: 2bb4c88e0f0461e7ccf27dc0614c56f29c71d6f8
35. Cevallos-Moreno, J.F., Rizzardi, A., Sicari, S., Coen-Porisini, A.: Deep reinforcement learning for intrusion detection in internet of things: Best practices, lessons learnt, and open challenges. Comput. Netw. **236**, 110016 (2023)
36. He, M., Wang, X., Wei, P., Yang, L., Teng, Y., Lyu, R.: Reinforcement Learning Meets Network Intrusion Detection: A Transferable and Adaptable Framework for Anomaly Behavior Identification. IEEE Transactions on Network and Service Management, (2024). S2ID: c278728159ae482a3d2c30c75e8147432b39ed48

37. Sayem, I.M., Sayed, M., Saha, S., Haque, A.: ENIDS: A Deep Learning-Based Ensemble Framework for Network Intrusion Detection Systems. IEEE Transactions on Network and Service Management, (2024). S2ID: 1112ac045088215252dc99b1d629ca33b0a5114b
38. Arreche, O., Bibers, I., Abdallah, M.: A Two-Level Ensemble Learning Framework for Enhancing Network Intrusion Detection Systems. IEEE Access, (2024). S2ID: fb7aaf6a1fbb78e0cb8b95f8cebada871930ecc9
39. Alotaibi, F., Maffeis, S.: Mateen: Adaptive Ensemble Learning for Network Anomaly Detection. International Symposium on Recent Advances in Intrusion Detection, (2024). S2ID: 0a584a51cf00d719f019ff936f5ba34ae728318b
40. Salek, M.S., Biswas, P.K., Pollard, J., Hales, J., Shen, Z., Dixit, V., Chowdhury, M., Khan, S.M., Wang, Y.: A Novel Hybrid Quantum-Classical Framework for an In-Vehicle Controller Area Network Intrusion Detection. IEEE Access, 11:96081–96092, (2023). MAG ID: 4385756467 S2ID: a3bcdd7be0a844f37556cf191ab11b5524e5a202
41. Hadi, H.J., Cao, Y., Li, S., Hu, Y., Wang, J., Wang, S.: Real-Time Collaborative Intrusion Detection System in UAV Networks Using Deep Learning. IEEE Internet of Things Journal, (2024). S2ID: d8f8308290aa179e328e53b7eeecee7155966a1d
42. Gyamfi, E., Gyamfi, E., Jurcut, A.D., Jurcut, A.D.: Novel Online Network Intrusion Detection System for Industrial IoT based on OI-SVDD and AS-ELM. IEEE Internet of Things Journal, pages 1–1, (2022). MAG ID: 4285138763 S2ID: c7239f29211e04c048cf71ee71d80ee7d36bd49e
43. Tang, B., Lu, Y., Li, Q., Bai, Y., Yu, J., Xu, Y.: A Diffusion Model Based on Network Intrusion Detection Method for Industrial Cyber-Physical Systems. Italian National Conference on Sensors, 23(3):1141–1141. (2023). MAG ID: 4317423310 S2ID: 5dd9dc9351810cea33175afc73d979c3b51b87c8
44. Fusco, P., Palmieri, F., Ficco, M.: Combining epsilon-greedy reinforcement learning based gradient sparsification and siamese neural networks for few-shot federated tinyml intrusion detection in iot. *Internet of Things*, page 101820, (2025)
45. Bertoli, G.D.C., Alves Pereira Júnior, L., Saotome, O., Dos Santos, A.L.., Alves Neto Verri, F., Augusto Cavalheiro Marcondes, C., Barbieri, S., Rodrigues, M.S., Parente De Oliveira, J.M.: An end-to-end framework for machine learning-based network intrusion detection system. IEEE access, 9:106790–106805, (2021)
46. Cai, Z., Sener, O., Koltun, V.: Online continual learning with natural distribution shifts: An empirical study with visual data. In Proceedings of the IEEE/CVF international conference on computer vision, pages 8281–8290, (2021)
47. Amalapuram, S.K., Kumar, S., Tamma, B.R., Channappayya, S.: SOUL: A Semi-supervised Open-world contInUal Learning method for Network Intrusion Detection, December (2024). arXiv:2412.00911 [cs]
48. Boukela, L., Zhang, G., Yacoub, M., Bouzefrane, S.: A near-autonomous and incremental intrusion detection system through active learning of known and unknown attacks. In 2021 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), pages 374–379, June (2021). arXiv:2310.17430 [cs]
49. Uddin, M.A.., Aryal, S.: Mohamed Reda Bouadjenek, Muna Al-Hawawreh, and Md Alamin Talukder. A Dual-Tier Adaptive One-Class Classification IDS for Emerging Cyberthreats, (2024). arXiv:2403.13010 cs
50. Jahangir, M.T., Wakeel, M., Asif, H., Ateeq, A.: Systematic approach to analyze the avast iot-23 challenge dataset for malware detection using machine learning. In 2023 18th International Conference on Emerging Technologies (ICET), pages 234–239. IEEE, (2023)
51. Di Mauro, M., Galatro, G., Liotta, A.: Experimental review of neural-based approaches for network intrusion management. IEEE Trans. Netw. Serv. Manage. **17**(4), 2480–2495 (2020)
52. Dong, S., Xia, Y., Peng, T.: Network abnormal traffic detection model based on semi-supervised deep reinforcement learning. IEEE Trans. Netw. Serv. Manage. **18**(4), 4197–4212 (2021)
53. Fieblinger, R., Alam, Md T., Rastogi, N.: Actionable cyber threat intelligence using knowledge graphs and large language models. In 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pages 100–111. IEEE, (2024)
54. Survey, A.: Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, and Anis Zouaoui. Graph Neural Networks for Intrusion Detection. IEEE Access **11**, 49114–49139 (2023)
55. Sivamohan, S., Sridhar, S.N.: An optimized model for network intrusion detection systems in industry 4.0 using XAI based Bi-LSTM framework. Neural computing & applications (Print), 35(15):11459–11475, March (2023). MAG ID: 4323864106 S2ID: 95f45f14463c4c8e94ec5ce728180cf33577d0ee

56. Elahi, Md.A., Ahammed Songram, R., Uz Zaman, Md.S.: Network-Shield: Exploring the Efficacy of GRU Model in Intrusion Detection Using CIC-IDS 2018 Dataset. In *Proceedings of the 3rd International Conference on Computing Advancements*, pages 1058–1065, Dhaka Bangladesh, October (2024). ACM
57. Mohammadpour, L.: Teck Chaw Ling, Chee Sun Liew, and Alihossein Aryanfar. A survey of CNN-based network intrusion detection. Applied Sciences **12**(16), 8162 (2022). (**Number:**)
58. Cevallos-Moreno, J.F., Rizzardi, A., Sicari, S., Coen-Porisini, A.: Nero: Neural algorithmic reasoning for zero-day attack detection in the iot: A hybrid approach. Computers & Security **142**, 103898 (2024)
59. Jesus, F., Cevallos, M., Rizzardi, A., Sicari, S., Coen-Porisini, A.: HERO: from High-dimensional network traffic to zERO-Day attack detection. Computer Networks, pages Accepted, In press, (2025)
60. Cevallos-Moreno, J.F., Rizzardi, A., Sicari, S., Coen-Porisini, A.: Asap: Automatic synthesis of attack prototypes, an online-learning, end-to-end approach. *Computer Networks*, page 110828, (2024)
61. Mohiuddin Ahmed and Al-Sakib Khan Pathan: Deep learning for collective anomaly detection. Int. J. Comput. Sci. Eng. **21**(1), 137–145 (2020)
62. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems, vol. 30, (2017)
63. Webb, T.W., Frankland, S.M., Altabaa, A., Segert, S., Krishnamurthy, K., Campbell, D., Russin, J., Giallanza, T., O'Reilly, R., Lafferty, J., et al.: The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*, (2024)
64. Laenen, S., Bertinetto, L.: On episodes, prototypical networks, and few-shot learning. Adv. Neural. Inf. Process. Syst. **34**, 24581–24592 (2021)
65. Farrukh, Y.A., Wali, S., Khan, I., Bastian, N.D.: Detecting unknown attacks in iot environments: An open set classifier for enhanced network intrusion detection. In MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM), pages 121–126. IEEE, (2023)
66. Wu, Y., Zhang, L., Yang, L., Yang, F., Ma, L., Lu, Z., Jiang, W.: Intrusion Detection for Internet of Things: An Anchor Graph Clustering Approach. *IEEE Transactions on Information Forensics and Security*, (2025). Publisher: IEEE
67. Alqahtani, A.H.: An incremental hybrid adaptive network-based IDS in Software Defined Networks to detect stealth attacks, (2024). arXiv:2404.01109 cs
68. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks, October (2018). arXiv:1806.01261 [cs]
69. Veličković, P., Blundell, C.: Neural algorithmic reasoning. Patterns, 2(7), (2021)
70. Numeroso, D.: Reasoning Algorithmically in Graph Neural Networks, February (2024). arXiv:2402.13744 [cs]
71. GNS3 Project. GNS3: The software that empowers network professionals. https://www.gns3.com, (2024)
72. Ridwan, M.A., Asyikin Mohamed Radzi, N., Hanis Mohd Azmi, K., Abdullah, F., Munirah Wan Ahmad, W.S.H.: A new machine learning-based hybrid intrusion detection system and intelligent routing algorithm for mpls network. International Journal of Advanced Computer Science and Applications, 14(4), (2023)
73. Azmi, A.A., Mohd Hussin, Z., Mohammad, S.: Exploring iot security in iot devices. In Advances in Technology Transfer Through IoT and IT Solutions, pages 103–113. Springer, (2023)
74. Kummerow, A., Henneke, M., Bachmann, P., Krackruegge, S., Laessig, J., Nicolai, S.: Cyber-security platform for the transparent cyber-attack detection in energy supply infrastructures. In ETG Congress 2023, 1(2) pages 1–7. VDE, (2023)
75. OpenFlow: Enabling innovation in campus networks. https://www.opennetworking.org/sdn-resources/onf-specifications/openflow/. Open Networking Foundation
76. Garcia, S., Parmisano, A., Erquiaga, M.J.: Iot-23: A labeled dataset with malicious and benign iot network traffic, (2020)
77. Koch, G., Zemel, R., Salakhutdinov, R., et al.: Siamese neural networks for one-shot image recognition. In ICML deep learning workshop **2**, 1–30 (2015). (**Lille**)
78. Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I.W.H., Ng, L.G., Ginhoux, F., Newell, E.W.: Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 37(1):38–44, 2019

79. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, (2006)

Jesús F. Cevallos M. received a Ph.D. in Computer Science Engineering from Sapienza University (Rome) in 2022. He now holds a post-doc 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 Associate 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.

Authors and Affiliations

Jesús F. Cevallos Moreno[1] · Alessandra Rizzardi[1] · Sabrina Sicari[1] · Alberto Coen-Porisini[1]

✉ Sabrina Sicari
sabrina.sicari@uninsubria.it

Jesús F. Cevallos Moreno
jf.cevallosmoreno@uninsubria.it

Alessandra Rizzardi
alessandra.rizzardi@uninsubria.it

Alberto Coen-Porisini
alberto.coenporisini@uninsubria.it

[1] Dipartimento di Scienze Teoriche ed Applicate, Università degli Studi dell'Insubria, Via O.Rossi 9, 21100 Varese, Italy