This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

TIGER: an open-source cyber-Threat Intelligence Game Environment for Reinforcement learning

Jesús F. Cevallos-Moreno, Alessandra Rizzardi *Member, IEEE*, Sabrina Sicari *Senior Member, IEEE,* Alberto Coen-Porisini

***Abstract*—Open-source testbeds for intrusion detection and mitigation enable benchmarking the efficacy of machine-learning-based cyber-defensive systems under increasingly realistic, heterogeneous network scenarios. In this context, the open-world nature of network intrusion detection requires defences to use continual learning strategies to adapt pattern-matching to new attack classes. The cost of periodically fine-tuning pre-trained detectors is not only computational but also encompasses the broader Cyber Threat Intelligence (CTI) life-cycle, which involves collecting, analyzing, and processing raw data into actionable insights. For ML-driven defensive systems, such actionable CTI ultimately takes the form of curated, labelled traffic traces of novel attacks. However, the concurrent optimisation of these intelligence-gathering costs and defence effectiveness has received little attention from the research community. In this respect, this work presents TIGER, an open-source Threat Intelligence Game Environment for Reinforcement learning-based agents to be trained and evaluated toward the optimisation of the costs-benefit trade-off associated with realistic ML-driven cyber-defence life-cycles. TIGER uses realistic network simulation software to model an active-learning game in which an agent learns to timely purchase CTI—abstracted in our environment as labelled samples of Zero-day attacks— to retrain its intrusion detection machinery on new attack patterns, while considering a constrained resource availability scenario.**

***Index Terms*—Automated Cyber-Threat Intelligence, Deep Reinforcement Learning, Continual Learning, Network Intrusion Detection.**

I. Introduction

Network Intrusion Detection Systems (NIDS) based on Machine Learning (ML) may be useful in real-world scenarios only if equipped with a continuous learning strategy [1], where models are periodically trained on new data [2]. Labelled datasets that provide novel knowledge beyond the current system's expertise [3] may be useful to this end. In this respect the recent growth in the availability of Cyber-Threat Intelligence (CTI) feeds potentially facilitates enrich, correlate, and validate attack-related network traffic [4].

In the broader cybersecurity industry, Cyber Threat Intelligence (CTI) is defined as the continuous process of collecting and analysing data to gain insights regarding threat actors' motives, targets, and attack methodologies [5]. It encompasses a wide range of activities, from monitoring dark web forums to detonating malware in sandboxes. However, from the perspective of an automated, Machine Learning-driven NIDS, the actionable output of this complex CTI life-cycle is fundamentally data-centric: for an ML model to adapt to a new threat, the insights generated by human analysts and threat intelligence feeds must be translated into ground-truth labels and traffic traces.

Using new training data, any NIDS based on Supervised Learning can learn the distribution of novel attack patterns and defend effectively in real scenarios [6]. However, reaching a mature specification of new attack patterns may entail costs in effort, time, and budget for the data engineering team behind the NIDS [7]. While the majority of continual learning techniques assume that ML-based detection modules can be periodically fine-tuned with new labelled data representing novel attacks, such data might be available only at the cost of expensive collection, analysis, CTI-extraction, and labelling processes. Therefore, in this work, the concept of "acquiring CTI" is operationalised as market-based class discovery in the context of active learning [8], [9]. More specifically, the CTI acquisition process is abstracted into the costly availability of partial/corrupt/delayed labelled traffic traces for training.

In any case, the ML-based cyber-defensive cycle entails not only detecting attacks but also enriching the inference machinery's learning curriculum with new types of labelled data. As explained above, given the real cost of these data, the dynamic landscape of real-world cyber-defensive scenarios requires a fair balance between acquiring supervisory signals for training and relying on pre-trained models for traffic management. To the best of the author's knowledge, the automatic optimisation of such a balance is underexplored in the RL environments currently available for intrusion detection and mitigation, and this paper aims to be among the first steps in this direction.

In the context of ML-based cyber-defensive systems, finding the optimal control policy that balances CTI-acquisition costs and benefits of defensive actions to maximise the long-run profit is a complex problem involving the estimation of second-order gradients, namely, to quantify how actions over the curriculum influence detection effectiveness. In this respect, Reinforcement Learning (RL) [10] can be useful, in that it is well-known to optimise potentially complex reward signals through interactions with the surrounding environment.

RL agents are widely deployed in industrial contexts to assist decision-making and optimise the cost-benefit balance [11]. Crucially, optimising a task through RL presupposes access to an environment equipped with a well-designed reward signal that serves as a proxy for the goal being pursued.

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

JFCM: jf.cevallosmoreno@uninsubria.it, ORCID: 0000-0003-2752-4616

AR: alessandra.rizzardi@uninsubria.it, ORCID: 0000-0003-4765-5365

SS: (Corresponding author) sabrina.sicari@uninsubria.it, ORCID: 0000-0002-6824-8075

ACP: alberto.coenporisini@uninsubria.it, ORCID: 0000-0002-3788-8926

Manuscript received May 19, 2025; revised XX XX, 2025.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.

For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

This paper aims to contribute to this precise problem by providing an open-source, realistic network environment with a customisable reward signal that can drive the automatic balancing between reliance on unsupervised detection and mitigation, and continuous learning in ML-based intrusion detection.

Main contribution: This work presents TIGER, an open-source **T**hreat **I**ntelligence **G**ame **E**nvironment to train and evaluate **R**einforcement learning agents toward the optimisation of the cost-benefit trade-off associated with the life-cycle of attack intrusion and mitigation systems based on Machine Learning. Specifically, TIGER is the first environment to model the cost of enriching the learning curriculum on an active class-discovery basis, and balancing it with the potential benefits of attack mitigation based on pre-trained traffic classifiers. By doing so, TIGER aims to foster research on active learning for proactive cyber-defensive policies in resource-constrained scenarios[1].

Outline: This paper is organised as follows. Section II gives an overview of related works and their relation with TIGER. Instead, Section III offers a detailed exposition of the TIGER environment, including its design goals and internal mathematical framework. Section IV then documents a use-case study of three Deep-learning-based RL agents trained using TIGER, and finally, Section V offers some conclusions and points to future related research directions. Table I contains a list of the abbreviations used in this paper.

Table I
ABBREVIATIONS USED IN THIS WORK.

Abbreviation	Meaning
CTI	Cyber-Threat Intelligence
DDQN	Double Deep Q-Network
DQN	Deep Q-Network
DM	Decision Module
DRL	Deep Reinforcement Learning
FSL	Few-Shot Learning
IM	Inference Module
IoC	Indicators of Compromise
LLM	Large Language-Model
MDP	Markov Decision Process
NIDS	Network Intrusion Detection System
ML	Machine Learning
OOD	Out-of-distribution
POMDP	Partially-Observable Markov Decision Process
RB-IB	Relational-Bottleneck Inductive-Bias
RL	Reinforcement Learning
SDN	Software-Defined Networking
SIEM	Security Information and Event Management
SOC	Security Operations Center
SOTA	State-of-the-art
ZdA	Zero-day attacks

II. RELATED WORKS

Five main research areas intersect the current work: Automated cyber threat intelligence, ML-based network intrusion detection, RL-based intrusion mitigation, continual learning, and active learning strategies.

a) ***Automated cyber threat intelligence:*** The work in [12] leverages predictable grammatical relations between Indicators of Compromise (IoCs) derived from various public sources and their contextual terms within sentences to automate IoC extraction from unstructured text. Similarly, Zhou et al. [13] propose a neural-based sequence labelling model to extract IoCs from cybersecurity reports without requiring expert knowledge. A common weakness of this type of automatic extraction of IoCs based on named-entity recognition [14], [15] is the strong dependency of the model's performance on the quality and comprehensiveness of the training dataset.

More recent approaches instead rely on knowledge graphs to discover semantic relations among IoCs, automating a process that traditionally requires significant manual effort [16], [17]. Limitations of these approaches are related to handling complex sentence structures and grammatical patterns in CTI records [18], [19]. A recent trend instead uses Large Language Models (LLMs) for automatic CTI extraction [20].

Concerning approaches based on natural language processing, TIGER abstracts the process of CTI-insight distillation into the acquisition of labelled training data [21]–[23]. This abstraction is done to focus on balancing the cost-benefit trade-off between investing in CTI enhancement and the effective use of current intelligence to defend against cyberattacks.

b) ***ML-based network intrusion detection:*** Common statistical learning techniques for NIDS are either based on anomaly detection of unlabelled traffic [24], [25], closed-set supervised classification from labelled attack-traces [26], or the hybridisation of these [27], [28]. More recently, the research community is widely adopting the *open-world* assumption, under which new attack patterns arise during deployment that differ from those used in the pre-training phase [29]. This research trend is said to be focused on generalising the detection toward zero-day exploits [30], and some of these works rely on neuro-symbolic approaches [31]–[34] while others are more explicitly modelled as meta-learning architectures [22], [35]. The simulation of the acquisition and use of the Cyber-Threat Intelligence (CTI) modelled in TIGER implies precisely such an open setting. Specifically, TIGER uses the ASAP framework in [21], which adopts a particular form of hint-labelling [23] that permits the detection and differentiation of zero-day attacks through collective anomaly detection [36], [37].

c) ***Reinforcement learning environments for cyber-security:*** Offline reinforcement learning has been widely used in NIDS to implement continual learning, adversarial training, and data rebalancing strategies [38]. A more recent paradigm is instead online-RL, which requires the creation of interactive learning environments for intrusion detection and response.

Existing online-RL environments for cybersecurity can be broadly categorised into two groups based on their fidelity and operational focus. The first set of environments relies on simplified, graph-based abstractions that map defence and attack forces to probabilistic or scalar values. For

[1]TIGER is available as an open-source project at: github.com/DISTA-IoT/tiger

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

Table II
COMPARISON OF TIGER WITH STATE-OF-THE-ART OPEN-SOURCE RL CYBER-SECURITY ENVIRONMENTS.

Environment	Agent Focus	Fidelity / Data Source	CTI Acq.	Cost Focus	Primary Goal
Elderman *et al.* [39]	Red & Blue	Abstract / Game	No	None	Abstract game-theoretic attack/defense.
CyberBattleSim [40]	Red	Abstract / Simulated	No	None	Automated lateral movement and pentesting.
Yawning Titan [41]	Blue	Abstract / Simulated	No	Mitigation	Optimal node isolation/patching under operational costs.
CyberShield [42]	Red & Blue	Abstract / Simulated	No	Mitigation	Multi-agent competitive moving target defense.
NASimEmu [43]	Red	Emulated / Real Exploits	No	None	Generalizable automated penetration testing.
PenGym [44]	Red	Emulated / Real Exploits	No	None	Bridging the reality gap in pentesting agents.
Cyberwheel [45]	Blue	Sim+Emu / SIEM Logs	No	Deception	High-fidelity defence via cyber deception and decoys.
TIGER (Ours)	**Blue**	**Simulated / Real Traces**	**Yes**	**Intelligence**	**Cost-aware CTI acquisition and intrusion detection.**

instance, in the work of Elderman *et al.* [39], a discrete-time simulation implements an abstract game where an attack succeeds if its scalar value exceeds the defence value. Similar low-dimensional, abstract representations of network topologies and vulnerabilities are utilised in Microsoft's CyberBattleSim [40], Yawning Titan [41], and CyberShield [42]. While these environments are highly scalable and useful for training agents to perform network-level mitigation (e.g., node isolation, patching) or lateral movement, they abstract away the underlying network traffic and the complexities of actual intrusion detection.

A second set of more realistic environments builds emulated or hybrid infrastructures where real network metrics, alerts, or raw traffic are collected. Frameworks like NASimEmu [43] and PenGym [44] focus predominantly on training offensive (Red) agents for automated penetration testing using actual exploits. Recently, Cyberwheel [45] introduced a high-fidelity simulation and emulation environment specifically designed to train autonomous defence agents to deploy cyber deception strategies based on SIEM alerts rather than raw traffic. A summarised view of these environments is offerered in Table II.

d) ***Continual learning****:* Continual learning, also known as lifelong learning, is a critical area in ML that focuses on the ability of models to learn from a continuous stream of data over time [1], [46]. One of the main advantages of continual learning is the adaptation of the statistical learning machinery to distribution shifts [47], [48] while avoiding catastrophic forgetting of pre-training distributions [49], [50]. Continual learning techniques have recently been used to adapt ML-based intrusion detection systems to distribution shifts characterised by evolving network conditions [6], [51]. TIGER is within this research trend, and its main novelty is the simulation of a dynamical network environment to assess the automatic balancing of costs and benefits when applying continual learning strategies.

The framework of Korba *et al.* [52] models an evolving curriculum when training a NIDS. The authors used experience replay [53] to avoid catastrophic forgetting while adapting to new attack patterns. TIGER also models a dynamic curriculum while focusing on finding the optimal timing for triggering re-training cycles. The *spider* framework [54] focuses instead on low-data regimes and semi-supervised learning, using the *gradient projection memory* mechanism [55] to reduce task-specific gradient overlap. While also focused on low-training-data regimes, TIGER uses Few-Shot Learning (FSL) [56] modules to improve data efficiency and maintain an end-to-end differentiable architecture.

e) ***Cost-Sensitive and Stream-Based Active Learning****:* Traditional active learning literature typically focuses on instance-level queries to maximise sample efficiency in closed-world setups [57]. However, real-world deployments frequently encounter streaming data with emerging novel classes, necessitating algorithms for *Active Class Discovery* [58] and *Open-World Active Learning* [59] on an online basis [60]. When queries incur costs [9], [61], the problem becomes a cost-sensitive optimisation task. Prior works have begun exploring the coupling of stream-based active learning with reinforcement learning policies [62] to navigate these trade-offs. TIGER serves as a bridge between these domains. It models CTI sources as an active-learning market, providing a testbed for evaluating RL controllers tasked with balancing the cost of class discovery against the risks of immediate operational utility of intrusion detection.

Motivation and Novelty of TIGER. Despite the proliferation of RL environments for cybersecurity, a critical gap remains: *none of the existing frameworks model the operational cost-benefit tradeoff of acquiring external Cyber Threat Intelligence (CTI)*. While environments like Yawning Titan or Cyberwheel incorporate the costs of taking mitigative actions (e.g., isolating a node, maintaining a decoy), they assume the defender's observation space is either strictly local, perfectly known, or cost-free once extracted. In real-world Security Operations Centres (SOCs), analysts do not solely rely on local traffic; they must dynamically decide when to query expensive external threat intelligence feeds, sandbox environments, or expert analyses to classify ambiguous traffic.

TIGER was specifically introduced to bridge this gap. Being implemented on top of the SMARTVILLE testbed [63], TIGER is grounded in real intrusions and bases its detection on raw network information. However, TIGER fundamentally differentiates itself by expanding the RL agent's action space beyond simple alert generation or network mitigation. Instead, TIGER explicitly trains agents to optimise the *acquisition and utilisation of CTI assets* under a budget constraint. It teaches the NIDS to balance the monetary or computational cost of gathering intelligence against the penalty of misclassification, providing a realistic simulation of SOC resource management that previous simulators lack.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

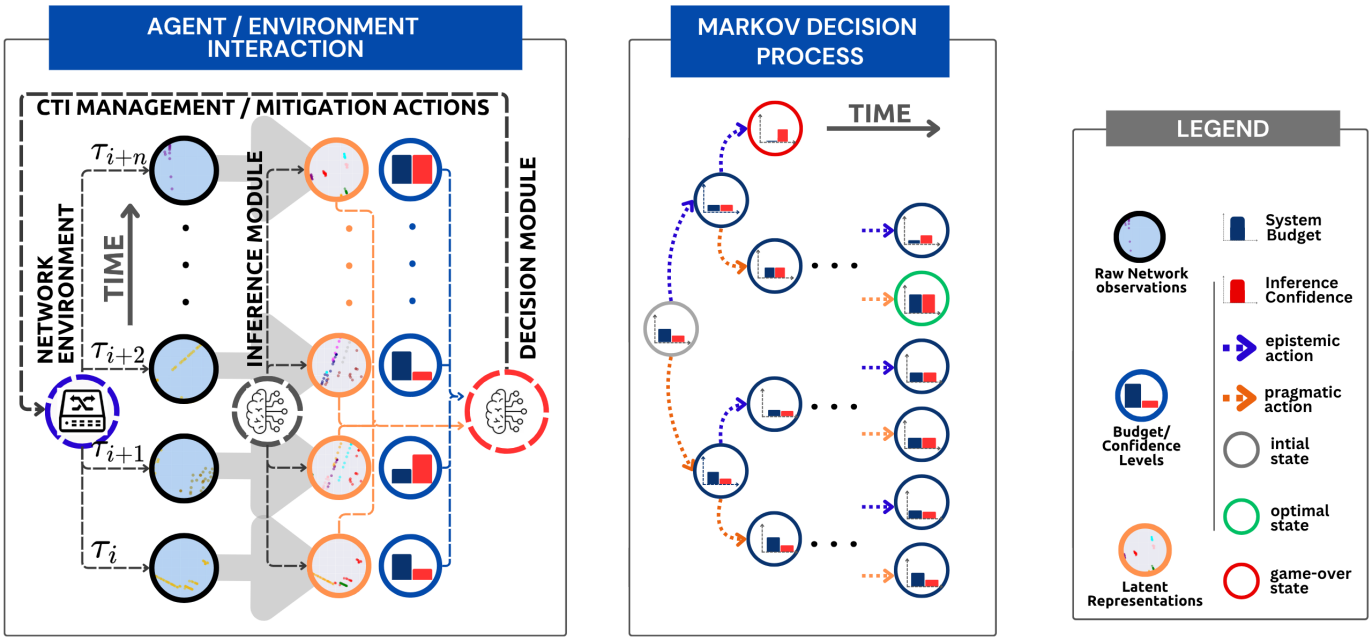


Figure 1. **Left:** The TIGER environment is a network system that receives benign and malicious traffic. A defensive agent has an *inference module* (IM) for intrusion detection and a *decision module* (DM) for attack mitigation. **Right:** In TIGER, the objective is to maximise the overall system's budget. Pragmatic actions (attack mitigation) may immediately influence the reward. The DM monitors the confidence of the IM's inferences and eventually augments it by taking epistemic actions (buying new training data).

III. The TIGER Environment

This section presents the design principles and working mechanisms of the TIGER environment. It first explains the game in which the agent and the environment interact. Such an explanation follows the left panel in Figure 1. The second part of this section illustrates how the key challenge modelled in TIGER relates to balancing exploitation and exploration throughout a sequential decision-making process, as shown in the right panel of the same figure.

A. Overview

a) ***The network simulation****:* The TIGER environment is based on SMARTVILLE [63], an open-source test-bed for online learning NIDS. SMARTVILLE offers a network simulation made of multiple attackers and honeypot nodes that resemble an Internet-of-Things (IoT) scenario on top of the GNS3 software[2]. Software-Defined Networking (SDN) is used in SMARTVILLE to collect raw traffic and flow statistics, thereby forming a rich feature space for network intrusion detection.

More specifically, the TIGER environment is a network system where malicious and benign nodes send traffic that passes through an SDN switch governed by an SDN controller, represented by the purple dashed circle in the left panel of Figure 1. To ensure a realistic setting, the network simulation is asynchronous to the event-based calls made to the agent. These calls occur when the SDN controller receives network-state snapshot messages from the forwarding switch. The black-bordered circles in the left panel of Figure 1 represent the raw network observations contained in those messages. The controller receives these messages periodically at time steps $\tau_i, \tau_{i+1}, ...\tau_{i+n}$, as shown in Figure 1, and takes actions related to attack mitigation or CTI acquisition, as explained in the following.

b) ***Modelling of the cyber-threat intelligence game****:* In the context of a cyber-defensive life-cycle, TIGER implements a sequential decision-making game in which the environment dynamics are designed to provide a training and evaluation ground for agents to balance the costs of acquiring training data with the benefits of using it to train an ML-based NIDS. To this end, the mentioned CTI game is designed with the following characteristics:

- The game has a unique player: the agent. This agent monitors a network system that generates both benign and malicious traffic and must take periodic actions to prevent cyber-attacks from succeeding. To this end, the agent is composed of an *Inference Module* (IM) and a *decision module* (DM), which are represented respectively by the grey and red dashed circles at the left panel in Figure 1. The IM is responsible for predicting network intrusion events. In contrast, the DM opts for obeying or not to such predictions -i.e., enacting or not mitigation actions- and, if eventually, buying new training data to augment the potential efficacy of the IM. In other words, at each time step, traffic flows try to enter the system, and the agent needs to decide if taking *pragmatic* actions, i.e., related to blocking or accepting such flows or an *epistemic* action, which is related to buying more data to train the IM. Pragmatic and epistemic actions are indicated by orange and purple dashed arrows in the right panel of Figure 1.
- The initial game conditions equip the agent with a pre-trained IM and a finite positive budget $\mathcal{B}$. Such initial

[2]https://gns3.com/

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

conditions are represented by the bar plot in the grey-bordered circle at the right panel in Figure 1. At each time step, each attack flow permitted to enter the system incurs a cost for the network and thus subtracts from the agent's budget. Instead, benign flows that are permitted to enter the system result in a positive contribution to $\mathcal{B}$. Finally, buying new data involves paying a price, thereby subtracting from the budget. The agent's goal is to maximise $\mathcal{B}$. The agent wins the game if it reaches $\mathcal{B}_{max}$ or loses if the budget equals $\mathcal{B}_{min}$. The winning and losing ending states for the game are represented by the green- and red-bordered circles in the right panel of Figure 1, respectively.

*c) **Neural architecture and continual learning strategy**:* At the beginning of each episode, the agent's inference module is fixed to a pre-trained state that uses a fixed set of attack-specific labelled data. As the episode progresses, epistemic actions generate (potentially noisy or partial) annotations on samples of new types of attack. Note that, even if the IM is continually trained within an episode, it is reset to the initial pre-trained state at the beginning of each episode. Otherwise, the CTI knowledge would accumulate across episodes, and the decision module would converge on using only pragmatic actions, because epistemic actions would no longer be useful. This section now explains the implementation of the continual learning strategy, which is schematised in Figure 2.

The TIGER environment aims to help train and evaluate agents (i.e., the decision module) on their ability to shape the curriculum for training intrusion detectors (i.e., the inference module). These agents need to balance the IM's training costs with its effectiveness in intrusion detection. Such a balance is not obvious for two main reasons:

1) Some zero-day attacks might appear highly anomalous to the IM. Thus, the DM might be able to block these attacks by relying on the current IM's predictions even before training it on the corresponding classes.
2) Other types of attacks may instead require the IM to be fine-tuned on labelled data to be detected. In this case, the DM should preemptively acquire such data to train the IM before these attacks impact the system.

In this respect, the neural architecture of the IM should better respect the following desiderata:

(a) Given a set of inputs, the IM should be able to perform anomaly detection and input classification concurrently.
(b) Anomalies should be eventually clustered into groups to allow the DM to block some clusters and let others pass.
(c) To seamlessly exploit CTI acquired after pre-training, the number of attacks to which inputs can be classified each time should be dynamic.

The requirements above motivate the adoption of the ASAP framework [21] for implementing the IM in the TIGER environment. Namely, while many neural intrusion detectors address either one or two of the above-listed requirements concurrently, the ASAP framework fulfils all of them without relying on symbolic priors, but rather in a fully differentiable manner.

The ASAP framework [21], which is a recurrent neural-representation pipeline exploiting the *relational-bottleneck* inductive bias [64] (RB-IB), hint-labelling strategies [23], and FSL [65] for concurrent multi-class classification, collective anomaly detection and open-set recognition. Through manifold learning and meta-training, the latent space of ASAP converges to a clustering-friendly geometric distribution of neural representations. Additionally, given a set of input representations, the ASAP network exploits the RB-IB to perform multi-class classification using a set of labelled support samples, as commonly done in FSL [65][3].

In ASAP, a fixed subset of known classes is labelled as pseudo-unknown traffic to train an adjacency-matrix prediction layer to perform collective anomaly detection [36]. By doing so, ASAP learns to generalise this behaviour over completely held-out classes at test time. From a geometrical perspective, ASAP uses metric-based meta-learning [66]. Namely, the IM predicts whether an input should be associated with any known class based on its distance to the corresponding *prototype*[4].

TIGER builds a continual learning strategy over ASAP. Specifically, during any game or episode, the agent stores incoming traffic observations for known classes in class-specific buffers. The re-training procedure samples from these buffers at every time step to perform batch gradient descent. Each time an epistemic action is taken, the set of known classes is augmented with a new class, and the corresponding input samples will begin to be labelled and stored in a new dedicated buffer. From that moment on, the buffer of the new class will be among the sampled ones for re-training purposes until the end of the episode.

*d) **Relation with active learning**:* From the perspective of active learning, our work corresponds to the open-world setting [59] as opposed to classical pool-based approaches. Using an online learning testbed implies that our setting is also a stream-active learning setting [60], with the important difference that the (potentially corrupt or incomplete) oracle labels streams forever (until the end of an episode) after being queried. The epistemic actions in TIGER coincide with cluster-based queries similar to those in [67], but our oracle reveals the label of the majority class in each cluster, rather than the class of some chosen cluster participant. Additionally, rather than relying on predefined heuristics or statistical models for novelty detection [57], our modelling seeks to optimise the query criterion via reinforcement learning [62].

B. Threat model

The adversarial setting under which TIGER agents operate explicitly is now provided, structuring it into the assets at stake, the adversary's capabilities, the trust placed on the CTI feed, and the surface through which the agent's policy could be attacked.

[3]The ASAP framework relies on a prototypical classifier [66], and on a relational neural network to implement kernel regression. Within this blueprint, any neural architecture can be used in ASAP. Appendix A-A contains a comparative experiment using multiple neural architectures for the kernel regression step in ASAP.

[4]In metric-based meta-learning, the prototype of a class is the vector that corresponds to the component-wise average of the latent representations of the support samples of such a class [66].

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

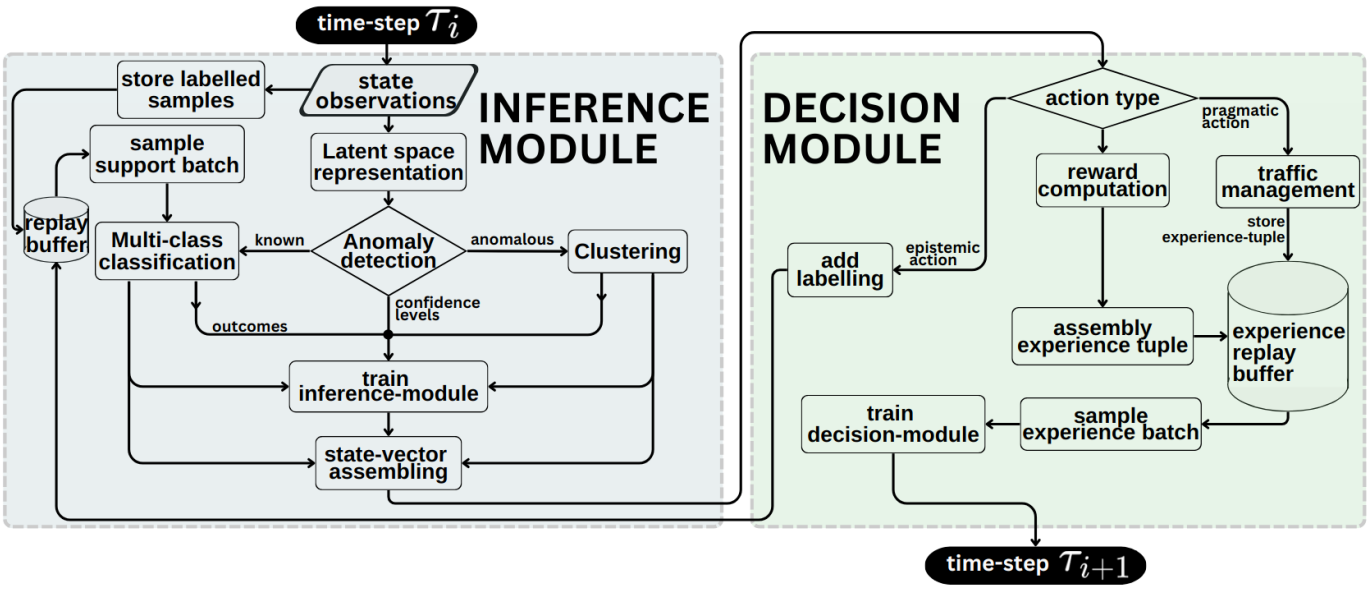


Figure 2. The TIGER agents have an inference module (left-panel) and a decision module (right-panel). The former produces predictions related to multi-class classification and collective anomaly detection and is continuously trained using replay buffers. The latter instead evaluates those predictions and learns to trigger defensive traffic management actions or acquire new attack-specific replay buffers for re-training the inference module.

*a) **Assets and defender**:* The defender is the single agent of Section III-A, comprising the inference module (IM) and the decision module (DM). Two assets are at stake: the *operational budget* $\mathcal{B}$, which aggregates the cost of admitting malicious flows, the opportunity cost of blocking benign flows, and the price paid for CTI; and the *detection capability* of the IM. The defender seeks to preserve both, keeping $\mathcal{B}$ away from the losing threshold $\mathcal{B}_{min}$, under a stream of both known and previously unseen attacks.

*b) **Adversary capabilities**:* The adversary is modelled as a set of malicious nodes that inject attack traffic into the network (Section III-A). Its capability is to introduce, at scheduled times, flows belonging either to attack classes already known to the IM at pre-training time, or to novel post-deployment attack classes (zero-days) that the IM has never been trained on — the latter being exactly the classes for which the DM may acquire CTI.

The adversary is *passive and non-adaptive*: it neither observes the defender's internal state nor reacts to its block, accept, or acquire decisions, so the traffic-generating process is exogenous and stationary with respect to the agent's policy (an assumption stated and justified in Section III-C). An *adaptive* adversary — one that mutates its traffic to evade the current detector, mimics benign flows, or backs off when blocked — is left as future work.

*c) **CTI-feed trustworthiness**:* By default, the CTI provider is treated as a trusted oracle: upon payment it returns clean, correctly labelled, complete, and immediately usable traffic traces for the acquired class. Because this is an idealisation, TIGER optionally activates a CTI-delivery-fidelity mechanism as presented in Section III-C, under which a purchased feed may contain *mislabelled* samples — a configurable fraction of the acquired class's labels flipped to a wrong class, which directly captures a *malicious or unreliable provider that injects poisoned labels* — cover only part of the class's behaviour, or arrive after a delay.

The severity of the first two impairment channels can be coupled to the hidden quality of the targeted cluster, so that indiscriminate purchasing exposes the defender to noisier intelligence. Both a trustworthy feed and a low-fidelity or partially poisoned one therefore lie within the environment's expressivity, whereas the IM's pre-training corpus is assumed clean and trusted.

*d) **Targeting the agent's policy and budget**:* A traffic-side, budget-draining adversary — one that deliberately crafts or floods traffic to exhaust the budget or to bait costly purchases — is not modelled, as it would require the adaptive capability excluded above. Both adaptive evasion and budget-draining attacks are regarded as important directions for future work (Section V).

However, through the fidelity mechanism introduced above, an unreliable provider can render an epistemic action wasteful by delivering corrupt intelligence that poisons the IM. The operational budget is the natural attack surface in this case: inducing wrong decisions — admitting malicious flows, or provoking unnecessary blocks of benign flows — drives $\mathcal{B}$ toward $\mathcal{B}_{min}$. The DM must learn to avoid such a waste through selective purchasing.

C. The CTI game's Markov Decision Process

Notice that different action sequences can lead to different rewarding paths. In the optimal-control and RL literature, the optimisation of sequential decision-making problems is commonly pursued through a modelling based on the Markov Decision Process (MDP) formalism [10], [68]. An MDP defines the set of possible *states* in which an agent can occur to be at any instant, a set of *actions* the agent can take at every state to interact with the environment, a set of *state-transition* dynamics, and a *reward* mechanism that describes how the environment reacts to the agent's actions.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

Formally, an MDP is represented by a 5-tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma)$, where the elements indicate, respectively, the state space, the action space, the state-transition probability distribution, the reward mechanism, and the discount factor. This subsection introduces the meaning of the elements in the 5-tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma)$ and its specific implementation in TIGER, following the scheme in Figure 3.

*a) **State space**:* The state space $\mathscr{S}$ is the set of all the possible states or instantaneous system conditions that are realistically visible to the agent. Any state sample comprises two types of information: *exteroceptive* and *proprioceptive*. The former part concerns incoming raw traffic and traffic flow statistics, as defined in section III-A. The latter part is a fixed seven-channel vector: the count of predicted-known and predicted-anomalous flows in the current batch and their respective confidence levels, the fraction of this episode's zero-day pool already acquired, a flag indicating whether any CTI is still available for purchase, and the current system budget.

The state space elements are denoted by $\mathbf{s} \in \mathscr{S}$ and are dynamically packed in tensors to permit the implementation of RL agents based on neural networks. In the left panel of Figure 1, exteroceptive and proprioceptive states are schematised in the orange- and blue-bordered circles, respectively.

Figure 3 schematises instead the operations of the inference and decision modules. A set of input and support samples is fed to the IM to obtain clustered latent representations of traffic flows. Each cluster is equipped with the proprioceptive information to form a sample of the state space. Exteroceptive information consists of cluster-specific centroids, which are represented by the X marks in the grey circles in Figure 3. The proprioceptive vector, instead, is represented by a green square next to the cluster-specific or exteroceptive information.

Approximating the confidence level information. The decision module needs to know the confidence of the predictions made by the inference module to make informed decisions about enriching the CTI knowledge at a given time. In TIGER, the degree of *polarisation* of the logits in the output layer of the ASAP network is used to approximate confidence levels of the outcomes. While the default configuration relies on energy-based confidence estimates, alternative strategies such as entropy, margin, and MSP-like can also be used; a dedicated comparison and discussion are reported in Appendix B. More specifically:

- **Anomaly-detection confidence.** For each incoming sample, ASAP outputs one Sigmoid anomaly-prediction logit [69], which, if greater than a fixed threshold, associates the correspondent sample to an anomaly[5]. The confidence levels for anomaly detection are computed by averaging, over the cluster, the predicted anomaly probability assigned to the decided side of the boundary (the Maximum-Softmax-Probability criterion; see Appendix B.
- **Known traffic classification confidence.** ASAP also outputs a multi-class classification vector according to the set of currently known classes using the Softmax activation function. To approximate a confidence value, TIGER compares the average winning logit against the average of the non-selected logits, producing a log-ratio score (the *baseline* strategy; the default energy variant and the full family of strategies are detailed in Appendix B).

[5]A common practice in binary classification tasks, like anomaly-detection, is to use the Sigmoid function at the output layer to bias the network outputting values between 0 and 1 and polarising its outputs in regions near these bounds.

*b) **Action space**:* The action space $\mathscr{A}$ is the collection of all the possible actions an agent can perform for each state observation. Namely, these are: block the cluster of flows corresponding to the state observation, let it pass into the system, or buy training data to enrich the training curriculum (in which case the cluster is blocked). Within this action-space setting, the DM is said to implement the *action policy* function, typically denoted as π in the RL literature, which indicates what actions to take given a certain state: $\pi : \mathbf{s} \rightarrow \pi(\mathbf{s})$. Overall, the agents seek to maximise the *discounted future reward*, G_τ, which is defined as:

$$G_\tau = R_\tau + \gamma R_{\tau+1} + ... = \sum_{k=0}^{\infty} \gamma^k R_{\tau+k} \tag{1}$$

where R_τ is the reward the agent receives at time τ after executing an action, and γ is a fixed parameter that reduces the value of future action rewards concerning the immediate one and is known as the *discount factor*.

The objective is then to learn an action *policy* that maximises (1). Given a specific policy π, the *action value function*, also called *Q-value function* indicates how valuable it is in expectation to take a specific action $\mathbf{a}$ being at state $\mathbf{s}$ at time τ and using π to choose the subsequent actions:

$$Q_\pi(\mathbf{s}_\tau, \mathbf{a}_\tau) = \mathbb{E}_\pi[G_\tau \| \mathbf{s}_\tau, \mathbf{a}_\tau] \tag{2}$$

where $\mathbb{E}_\pi$ indicates the expectation of G_τ if acting with the policy π from instant $\tau+1$ on, $\mathbf{s}_\tau$ and $\mathbf{a}$ respectively indicate the state and the action taken at time τ.

Three actions can be taken at each step: blocking a cluster, accepting it, or buying data, where the "acquiring CTI" action in TIGER abstracts a real-world Security Operations Centre (SOC) workflow. In a real-world scenario, a SOC analyst might query a commercial threat intelligence platform, send a suspicious payload to a malware analysis sandbox, or escalate a ticket to a Tier-3 analyst. All of these actions incur a financial and temporal cost. TIGER mathematically abstracts this cost-benefit tradeoff: the agent pays a penalty (representing the cost of the CTI service) and, in return, receives labelled samples of the anomalous traffic (the actionable output of the CTI process), allowing the underlying NIDS to use these supervisory signals for further training.

In its default (*hard*) configuration, a single epistemic purchase is bundled: for one price it both delivers the class label and grants a persistent anomaly-detection oracle for that class, so that, for the remainder of the episode, the acquired class's flows are treated as known without re-consulting the inference module's anomaly detector. A *soft* variant, in which a purchase grants the label only, is also available.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

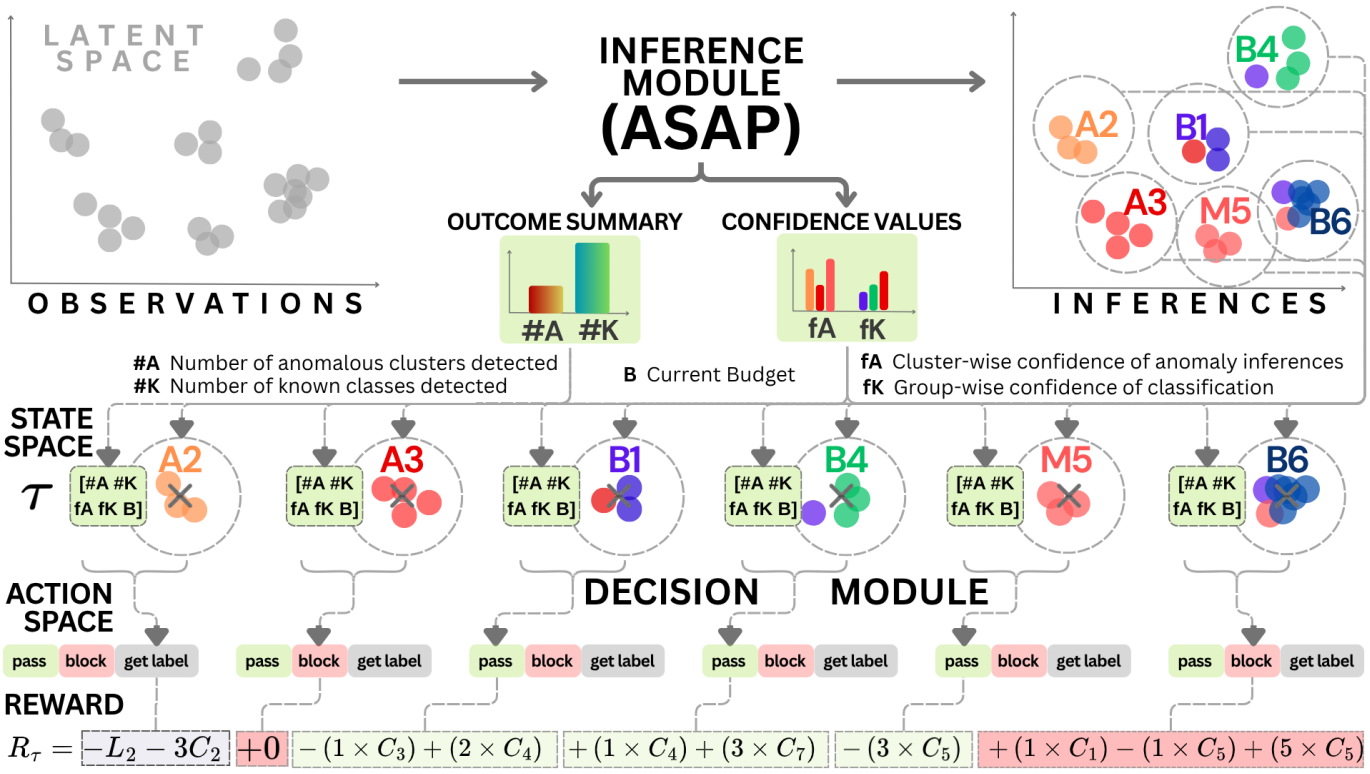


Figure 3. State-space, action-space and reward mechanism in TIGER. The decision module processes the inference module's outputs to decide whether acquiring or not CTI knowledge of class i. In case of acquisition, a cost of L_i is paid. The decision module may also block or accept traffic flow clusters. The reward equation shows that, for each benign flow that is blocked or malicious flow that is accepted, a class-specific cost C_j is incurred, where j denotes the flow's class. Conversely, accepting a benign flow or blocking a malicious flow is positively rewarded using C_j.

*c) **CTI delivery fidelity**:* In its default configuration, TIGER delivers acquired CTI as clean, correctly labelled, and complete traffic traces that become usable in the same time step of the purchase: taking the epistemic action moves the target class from the set of unseen zero-days directly into the known set, and its replay buffer — which already stores true-labelled online samples — becomes eligible for inference-module training immediately. This is a deliberate simplification. However, the quality of real-world CTI feeds can vary: a purchased capture may cover only part of a class's behaviour, and the intelligence often arrives with a delay. To let researchers study the impact of these imperfections — rather than assume them away — TIGER exposes an optional *CTI-delivery-fidelity* model that reintroduces them as three orthogonal, composable configuration channels:

(i) **Label quality**: a configurable fraction of the acquired class's training labels is flipped to a wrong known class, so that the newly-formed prototype is estimated from partially mislabelled evidence.
(ii) **Partial capture**: only a fraction of the acquired class's samples is admitted into its replay buffer, so the prototype is built from sparser, less representative data.
(iii) **Delivery delay** (optionally jittered): the class becomes trainable only a configurable number of decision steps after payment; during the lag it remains an unlearnable — and therefore still costly — zero-day, so that mistimed purchases are penalised.

Crucially, the severity of the label-noise and partial-capture channels can be *coupled* to the (hidden) quality of the cluster the CTI is bought for: purchasing intelligence for a messy, low-purity or low-confidence cluster yields noisier and sparser data than purchasing for a clean one, whereas the delivery delay models feed latency and is left independent of cluster quality. In terms of *policy transferability*, note that a controller trained under clean delivery might learn an over-eager acquisition policy that transfers poorly to a realistic, imperfect feed; conversely, the fidelity model allows a policy to be both trained and evaluated at a delivery quality that matches the target deployment. A first empirical characterisation of the environment under a representative subset of these regimes is provided in Section IV-C, leaving an exhaustive quantitative study to future work.

*d) **State-transition probabilities and reward mechanism**:* After an action is taken, state *transitions* occur in the environment following a state transition probability distribution $\mathscr{P}$ that allows a certain degree of stochasticity[6]. In the TIGER environment, these transitions are associated with fluctuations in the available budget, the predictions of the inference module, and its corresponding confidence levels. Incoming traffic follows the baseline IoT traffic simulation provided by SMARTVILLE, and is not influenced by the agent's actions.

In other words, benign and malicious nodes send traffic to the system without interruption, and the system is constantly

[6]Notice that such randomness may also subsume the incomplete knowledge of the environment dynamics that an agent realistically has.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

asked to accept or block it. Thus, TIGER treats the traffic-generating process as stationary and non-reactive, i.e., arriving flows are drawn from a fixed process that the agent's mitigation and acquisition decisions do not alter.

Note that the scope of our work does not include adversarial defence against co-adapting attackers either. It instead focuses on balancing the cost of learning novel attack patterns against detection effectiveness. While non-reactivity faithfully reflects a broad and practically relevant class of threats — automated and pre-programmed malware, botnets, worms, and scanners — future work could expand TIGER to model reactive and adaptive adversaries.

The budget is clearly influenced by the agent's actions, as explained in section III-A. Formally, the environment is said to deliver rewards following a reward policy $\mathscr{R}$, which is a function of the current state $\mathbf{s}$, and the action taken $\mathbf{a}$. At each time step τ, the action rewards will be summed to the current system's budget:

$$\mathcal{B}_{\tau+1} = \mathcal{B}_\tau + R_\tau, \quad \text{where} \quad R_\tau = \mathscr{R}(\mathbf{s}_\tau, \mathbf{a}_\tau) \tag{3}$$

As shown in the lower region of Figure 3, the influence of accepting/blocking traffic over the budget is class-specific. Namely, different attacks represent different costs to the network when they succeed, and different types of benign traffic produce different revenues if they are allowed to pass into the system. Moreover, the reward system not only assigns different costs to different cluster-related actions; it also computes a single reward for each incoming network flow, and the total pragmatic reward is the sum of the rewards for each input. Notice that classifying an unknown benign traffic sample as an anomaly can still be positively rewarded if the sample is allowed to pass through the system (pragmatic actions).

Additionally, each unclassified (collective-anomaly) cluster incurs a *cluster-impurity penalty* proportional to $1 - \text{purity}$, where purity is the fraction of the cluster's flows sharing its majority true class. Because acquiring CTI removes zero-day classes from the unlabelled pool, a policy that buys effectively leaves purer clusters and is penalised less; this term therefore credits improved perception rather than the mitigation action itself.

The CTI acquisition agent, on the other hand, has different prices depending on the attack/benign class. These per-class costs, revenues, and prices are specified as hyperparameters. Finally, note that the eventual epistemic actions taken by the agent at time τ, in addition to influencing the immediate reward, may also positively affect the predictions and the confidence level of the inference module in future states. Notice that even if pragmatic actions may influence system dynamics and rewards more quickly, epistemic actions can be game-changing in the long run.

*e) **On the theoretical guarantees of optimal policies**:* A fundamental characteristic of the MDP framework is the Markov property, which imposes a specific conditional independence on the system's dynamics. Namely, given the current state $\mathbf{s}_\tau$ and action $\mathbf{a}_\tau$, the probability distribution of the subsequent state $\mathbf{s}_{\tau+1}$ and the corresponding reward R_τ are entirely independent of the history of previously visited states, executed actions, or received rewards. Formally:

$$\begin{aligned}&\mathcal{P}(\mathbf{s}_{\tau+1}, R_{\tau+1} \mid \mathbf{s}_\tau, \mathbf{a}_\tau, \mathbf{s}_{\tau-1}, \mathbf{a}_{\tau-1}, \ldots, \mathbf{s}_0, \mathbf{a}_0) \\ &\quad = \mathcal{P}(\mathbf{s}_{\tau+1}, R_{\tau+1} \mid \mathbf{s}_\tau, \mathbf{a}_\tau)\end{aligned} \tag{4}$$

Under certain conditions, the Markovian property guarantees the existence of at least one deterministic optimal policy, denoted as π^* which yields an expected discounted future reward —as defined in (1)— that is greater than or equal to that of any other policy, from any arbitrary starting state. Similarly, the optimal state-action value function can be formally defined as $Q^*(\mathbf{s}_\tau, \mathbf{a}_\tau) = \max_\pi Q_\pi(\mathbf{s}_\tau, \mathbf{a}_\tau)$, which represents the maximum expected return achievable when taking action $\mathbf{a}_\tau$ in state $\mathbf{s}_\tau$ and subsequently following π^*:

$$Q_{\pi^*}(\mathbf{s}_\tau, \mathbf{a}_\tau) = \max Q_\pi(\mathbf{s}_\tau, \mathbf{a}_\tau)$$

The system dynamics of TIGER are more precisely modelled as a Partially Observable Markov Decision Process (POMDP). Formally, a POMDP extends the MDP tuple with an observation space Ω and an observation probability distribution (observation kernel) $\mathscr{O}$, yielding the 7-tuple $(\mathscr{S}, \mathscr{A}, \Omega, \mathscr{P}, \mathscr{O}, \mathscr{R}, \gamma)$.

Notice the IM is updated using stochastic gradient descent (SGD) sampling from a replay-buffer β_τ:

$$\theta_{\tau+1} = \theta_\tau - \eta \nabla_\theta \mathcal{L}(\theta_\tau; \xi_\tau), \qquad \xi_\tau \sim \text{Sample}(\beta_\tau), \tag{5}$$

Crucially, in SGD, $\theta_{\tau+1}$ depends not only on x_τ θ_τ, but also on which buffer configuration β_τ the training batch is sampled from. Moreover, because β_τ is a hidden variable that depends on the whole action/observation history, the minimum sufficient statistic for the observation and reward statistics is the triple $(x_\tau, \theta_\tau, \beta_\tau)$, which we denote as z_τ:

$$z_\tau = (x_\tau, \theta_\tau, \beta_\tau). \tag{6}$$

Notice that each component of $z_{\tau+1}$ is a function of (z_τ, a_τ) plus fresh, history-independent noise only: (i) $x_{\tau+1} \sim P_x(\cdot \mid x_\tau)$ is exogenous background traffic, unaffected by past actions; (ii) $\beta_{\tau+1} = U(\beta_\tau, x_\tau, a_\tau)$ deterministically appends the current observations to their class buffers and opens a new buffer for class c whenever a_τ is the epistemic action for class c; and (iii) $\theta_{\tau+1}$ follows (5), which is deterministic in $(\theta_\tau, \beta_\tau)$ up to the replay-sampling draw $\xi_\tau \sim \text{Sample}(\beta_\tau)$, itself a fresh (in our case, seeded) noise term. Since the reward R_τ is a function of $(x_\tau, \theta_\tau, a_\tau)$, hence of (z_τ, a_τ), it follows by construction that $P(z_{\tau+1}, R_\tau \mid z_\tau, a_\tau, \text{history}) = P(z_{\tau+1}, R_\tau \mid z_\tau, a_\tau)$, i.e., the augmented triple z_τ satisfies the Markov property [70].

Notice that partial observability is a direct consequence of this construction: The decision module never observes z_τ directly. Instead, it only observes the IM's outputs $o_\tau = f(x_\tau, \theta_\tau) \in \Omega$ which reveal neither θ_τ nor β_τ exactly. For this reason, TIGER is modelled as a POMDP with observation space Ω and a deterministic observation kernel $\mathscr{O} : o_\tau = f(x_\tau, \theta_\tau)$.

In the POMDP framework, decision-making can be grounded in the belief state b_τ, the posterior distribution over

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

the hidden state z_τ given the observation history, which serves as a sufficient statistic for optimal control [71], [72]. Under standard regularity conditions — bounded rewards, discounted infinite horizon with $\gamma \in [0, 1)$, and measurable transition kernels — the existence of an optimal stationary policy over the belief-state space is guaranteed [73]. This ensures that the sequential decision problem modelled in TIGER is, in principle, well-posed.

Computing the optimal belief-state policy is, however, PSPACE-complete in general [74], and is practically intractable for the high-dimensional hidden state considered here. The DQN and DDQN agents evaluated in Section IV act on single observations rather than the full belief state, which is suboptimal in the general POMDP case [75]. Nevertheless, memoryless policies on partial observations can approach optimality when single observations are sufficiently informative about the hidden state [76]. In TIGER, the confidence levels and clustering quality present in o_τ are direct reflections of the IM's training state, providing a useful proxy for the belief. The empirical convergence documented in Section IV is consistent with this approximation regime and demonstrates effective curriculum-shaping behaviour.

This interpretation is further supported by the ablation study in Section IV-B. When the Inference Module is replaced by a perfect oracle, the agent observes a much more informative signal and the partial-observability bottleneck is substantially reduced. In that regime, the consistently higher rewards and stable convergence are compatible with the theoretical picture above: although they do not constitute a proof of optimality, they indicate that the sequential decision problem implemented in TIGER is well posed and that, as the observation function becomes more faithful to the latent state, the learned policy can move closer to the optimal control law defined on the underlying Markov process. In other words, the *Perfect IM* setting empirically suggests that the main obstacle to optimality is not an ill-defined game, but rather the imperfect state information available to the controller.

IV. Benchmarking DRL agents in TIGER

To illustrate how TIGER can be used as a benchmarking environment, a heterogeneous set of baselines has been trained and compared under the same reward function, budget constraints, and traffic conditions. The benchmark includes standard value-based agents based on Deep Q-Networks (DQN) [77], their Double DQN (DDQN) counterparts [78], a duelling variant (DUELLING-DDQN), and two additional RL baselines: A "never buy CTI" agent that operates pure anomaly-based mitigation actions (BUY-NOTHING), and two different agents that perform epistemic actions based on fixed confidence-thresholds, that compare buying ability against a heuristic that directly operationalises confidence scores (THRESHOLD-LO, THRESHOLD-HI). Importantly, all the baselines use reinforcement learning for learning mitigation actions, which naturally leads to the optimisation of pragmatic gains. The ablation agents (*never-buy* and threshold-based) differ from the baselines only in the mechanism that drives the epistemic actions: in the case of the ablations, a fixed program schedules epistemic actions, while the baselines learn the CTI-acquisition policy automatically.

The aim of this section is not to argue that one particular controller is universally best, but rather to show that TIGER defines a coherent control problem in which qualitatively different agents can be evaluated under the same conditions and their learning dynamics can be meaningfully compared. The learning baselines cover complementary inductive biases. DQN and DDQN provide standard value-based references for sequential control under partial observability. Their duelling variants further decompose the action-value estimate into state-value and advantage components, which can be beneficial when actions have similar effects over a subset of states. More details on these agents can be found in appendix C-A.

A. *Result Summary*

Figure 4 reports running averages of four min-max scaled metrics across five-seeded experiments for each baseline. More specifically, for each metric, raw time-series data from individual experimental runs were first aligned to a common time index and linearly interpolated to handle discontinuities and missing observations. Subsequently, we aggregated these interpolated series across all five seeded runs within each algorithm group. This involved calculating the mean time series, along with the 95% confidence interval (CI) frontiers, at each time step. To reduce noise and highlight underlying trends, a 300-step simple moving average was applied to both the mean and the upper and lower 95% CI series (except for the episode counter plot in the right-bottom panel).

The upper-left line plot shows the unsupervised rewards associated with anomaly-based intrusion detection. These rewards increase when: a) the Inference Module correctly clusters anomalous observations, b) it correctly distinguishes anomalous-benign from anomalous-malicious clusters, and c) the Decision Module takes the corresponding blocking or forwarding action. The line plot of supervised rewards is instead proportional to the multi-class classification accuracy achieved by the IM on known traffic and to the quality of the associated management actions taken by the DM.

The sawtooth behaviour of these reward line plots reflects the inference module's reset at each episode (notice the episode counts in the bottom row of the grid). The fact that these curves exhibit an overall upward trend, however, indicates that the DM is learning a repeatable strategy for performing pragmatic and epistemic actions that lead to faster, more stable convergence of inference over time.

The total rewards line plot shows the sum of unsupervised and supervised rewards at each time step. These curves provide a compact view of the overall effectiveness of the coupled detection-and-control loop and reveal whether gains in one regime are offset by losses in the other, or whether the agent improves the global trade-off.

The mean budget-per-episode metric complements total reward by capturing whether the policy settles into a sustainable operating regime. Notice that two agents may obtain similar short-term rewards while differing substantially in how much resource slack they preserve. The panels in the

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

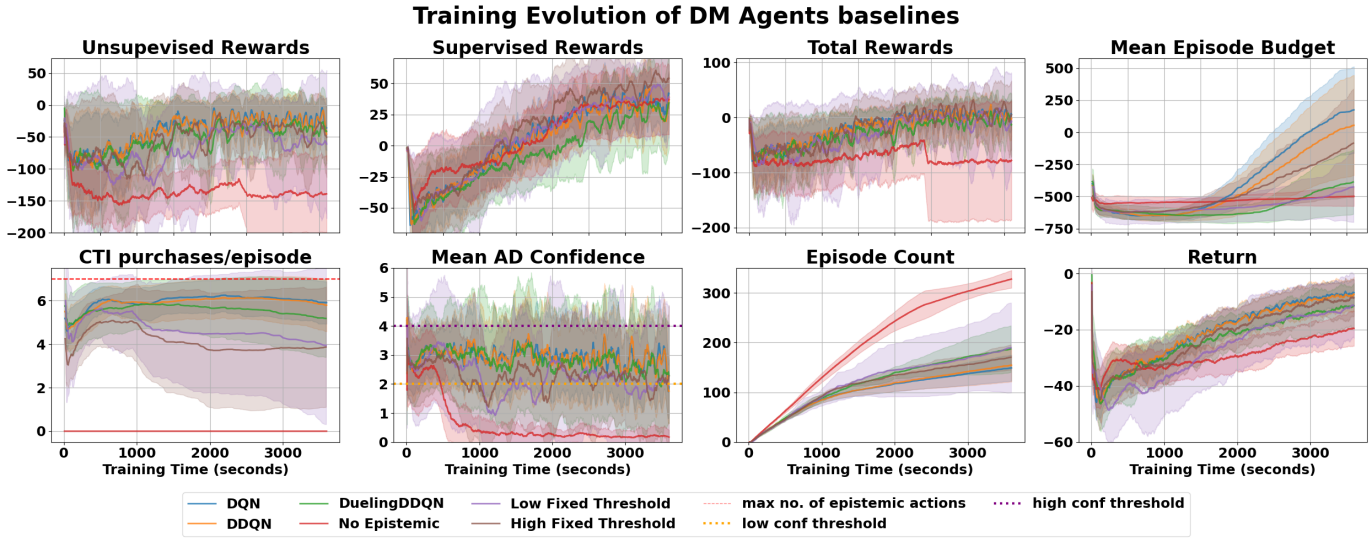


Figure 4. Transition between unsupervised anomaly-based intrusion detection and supervised multi-class cyber-attack classification. The goal of TIGER is not to overfit an inference module to a fixed closed set of attacks, but to provide a controlled environment in which one can evaluate how different decision agents manage the timing and cost of CTI acquisition, enabling the inference module to make progressively more informed decisions about zero-day cyber threats.

Table III
PERFORMANCE SUMMARY.

Metric	Group	Mean	Std	Median	Max	Metric	Group	Mean	Std	Median	Max
Unsup. Rewards	DQN	**−38.30**	23.37	−29.96	−4.22	Sup. Rewards	DQN	8.55	29.89	22.07	47.72
	DDQN	−41.48	22.63	−37.04	−6.38		DDQN	6.67	29.56	15.35	47.78
	DUELLING-DDQN	−52.54	20.12	−51.88	−3.80		DUELLING-DDQN	−1.60	26.35	−2.15	38.51
	BUY-NOTHING	−133.04	14.49	−134.86	−34.51		BUY-NOTHING	7.00	24.08	7.99	39.81
	THRESHOLD-LO	−59.90	29.18	−51.51	−8.37		THRESHOLD-LO	15.27	33.55	20.79	62.64
	THRESHOLD-HI	−49.44	25.14	−39.22	−11.06		THRESHOLD-HI	**20.88**	36.01	35.58	75.15
Total Rewards	DQN	−14.73	25.52	−4.88	20.38	Mean Ep. Budget	DQN	**−412.22**	345.43	−617.37	340.00
	DDQN	−17.41	24.26	−10.28	24.70		DDQN	−442.36	291.71	−614.01	164.85
	DUELLING-DDQN	−27.11	21.20	−23.42	14.33		DUELLING-DDQN	−558.37	136.18	−622.20	−181.03
	BUY-NOTHING	−73.04	12.51	−75.69	−3.20		BUY-NOTHING	−534.25	18.79	−543.94	−488.87
	THRESHOLD-LO	−21.45	29.04	−11.65	24.50		THRESHOLD-LO	−481.60	197.42	−594.25	−4.60
	THRESHOLD-HI	**−13.29**	29.80	−3.04	31.92		THRESHOLD-HI	−460.70	246.16	−599.97	202.09
CTI Purch./Ep.	DQN	5.84	0.44	5.95	6.36	Episode Count	DQN	**81.13**	47.31	81.15	162.61
	DDQN	5.75	0.44	5.95	6.24		DDQN	85.69	50.02	85.72	172.08
	DUELLING-DDQN	5.51	0.38	5.62	5.97		DUELLING-DDQN	102.89	60.22	102.90	206.97
	BUY-NOTHING	0.00	0.00	0.00	0.00		BUY-NOTHING	180.67	105.11	180.70	362.47
	THRESHOLD-LO	4.86	0.82	5.03	5.92		THRESHOLD-LO	95.09	55.71	95.05	191.19
	THRESHOLD-HI	4.50	0.54	4.39	5.39		THRESHOLD-HI	86.90	50.98	86.88	174.94
Mean AD Conf.	DQN	**2.89**	0.31	2.93	3.91	Return	DQN	**−18.01**	11.73	−14.81	−2.57
	DDQN	2.81	0.40	2.89	3.71		DDQN	−18.91	11.02	−16.13	−3.63
	DUELLING-DDQN	2.60	0.58	2.73	3.56		DUELLING-DDQN	−21.62	9.55	−20.06	−0.51
	BUY-NOTHING	0.64	0.76	0.30	3.63		BUY-NOTHING	−27.86	5.54	−29.68	−3.47
	THRESHOLD-LO	2.39	0.56	2.32	3.73		THRESHOLD-LO	−23.85	12.09	−21.28	−4.56
	THRESHOLD-HI	2.37	0.35	2.39	3.03		THRESHOLD-HI	−19.29	10.97	−16.76	−4.48

first row of Figure 4 are therefore especially informative for distinguishing between the effectiveness of learning optimal policies and that of heuristic epistemic exploration.

The episode count panel in Figure 4 helps visualise the pace of episode termination. The gradually decreasing positive slope of the episode counter indicates that, across episodes, the DMs converge toward better policies that keep the system operational longer. Notice there is no positive budget cap for episode termination in these experiments, and this detail helps convergence. See Appendix A-B for more experiments on different budget constraints and their influence on convergence.

Finally, the return panel in Figure 4 tracks the value of (1) across episodes. This plot shows a clear correlation between the reward policy and budget maximisation. However, it is worth stressing that TIGER aims to make visible the local interaction between resource consumption, reward fluctuations, and the timing of epistemic actions. In this respect, Figure 4 should be interpreted primarily as a diagnostic lens on how policies organise gameplay during training, rather than giving a final word on benchmark performance.

Table III contains descriptive statistics of the main evaluation metrics across the deep-RL agents (DQN, DDQN, DUELLING-DDQN) and the ablation baselines (BUY-NOTHING, THRESHOLD-LO, THRESHOLD-HI), aggregated over the five-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

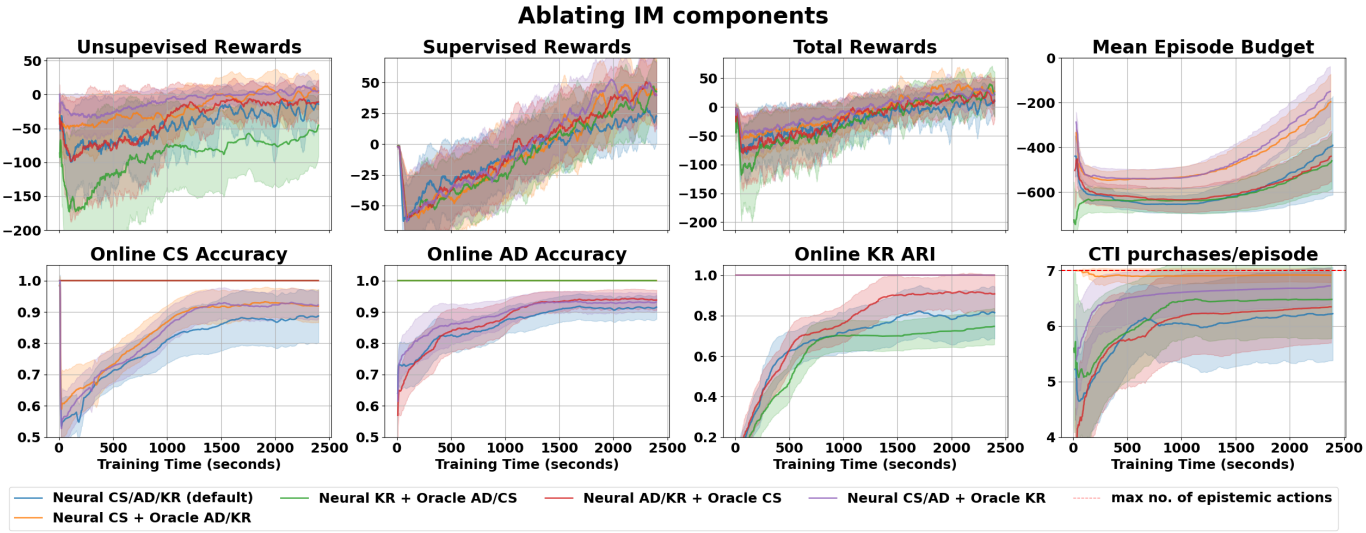


Figure 5. Running averages for performance metrics during 40 minutes of benchmarking in the TIGER environment ablating different outputs of the inference module and replacing by the ground-truth labels.

seeded experiments. For each metric, the best-performing configuration (by mean) is highlighted in **bold**. Higher values are better for all metrics except EPISODE COUNT, where fewer episodes indicate longer-lived, more stable runs; CTI PURCHASES/EPISODE is reported as a descriptive behavioural statistic and is therefore not ranked.

B. *Ablating the Components of the Inference Module*

Recall that a forward pass of the Inference Module (IM) in TIGER produces three outputs for each batch of network observations. First, it does Anomaly Detection (AD), i.e., it infers whether each sample belongs to a known class or is novel. Second, it performs Kernel Regression (KR), which means to cluster the unknown samples into groups that potentially correspond to a unique unknown traffic class. Third, it classifies the known samples (CS). As described in Section III, beyond an initial pretraining, these three outputs are learned online through gradient descent during each episode. To estimate the contribution of each subtask to the overall control loop, and to quantify the adverse effect of the partial observability as explained in paragraph III-C.

The results of an ablation of different IM modules is presented in Figure 5, where a subset of the IM outputs was gradually replaced by ground-truth labels. In all cases, the Decision Module (DM) still had to learn a DDQN policy for CTI acquisition and pragmatic traffic-management actions. These experiments were also repeated using 5 different seeds for each configuration. The reported line-plots including the 300-step running averages of mean values across runs and 95% confidence interval bands for each metric. The following configurations were tested during a 40-minute online training period:

1) **Fully-Neural IM** (NEURAL-CS-AD-KR): this is the default TIGER configuration, in which the IM is continuosly trained to solve the three subtasks jointly while the DM trains on the budget optimisation.
2) **Only Neural Classification** (NEURAL-CS_ORACLE-AD-KR): only the classification of known traffic is learned by the IM, while anomaly detection and clustering are provided by the environment.
3) **Only Neural Clustering** (NEURAL-KR_ORACLE-AD-CS): only the clustering of unknown traffic is learned by the IM, while anomaly detection and classification are replaced by the environment's perfect oracles.
4) **Not Neural Classification** (NEURAL-AD-KR_ORACLE-CS): anomaly detection and clustering remain learned components, whereas the labels of known traffic are provided as ground truth.
5) **Not Neural Clustering** (NEURAL-CS-AD_ORACLE-CS): anomaly detection and classification remain learned components, whereas the clustering of unknown traffic is provided as ground truth.

Figure 5 highlights that the learning curve of the inference modules materially limit downstream budget optimisation. The main trend in Figure 5 is that stronger inference configurations systematically accumulate larger episode budgets.

As the training progresses the learned policy buys more epistemic actions per episode, suggesting that better inference quality not only improves immediate detection performance but also influences the long-term exploration–exploitation trade-off.

Notice how the configurations in which either anomaly detection or classification is the only learned component still reach strong rewards, whereas NEURAL-KR_ORACLE-AD-CS remains the weakest setting despite having perfect anomaly/classification labels. In other words, the clustering task is the most critical in budget maximisation because it influences downstream efficacy of blocking/accepting decisions.

C. *Probing CTI Delivery Fidelity*

To illustrate the *CTI-delivery-fidelity* mode in paragraph III-C, we benchmarked the default TIGER agent while varying

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

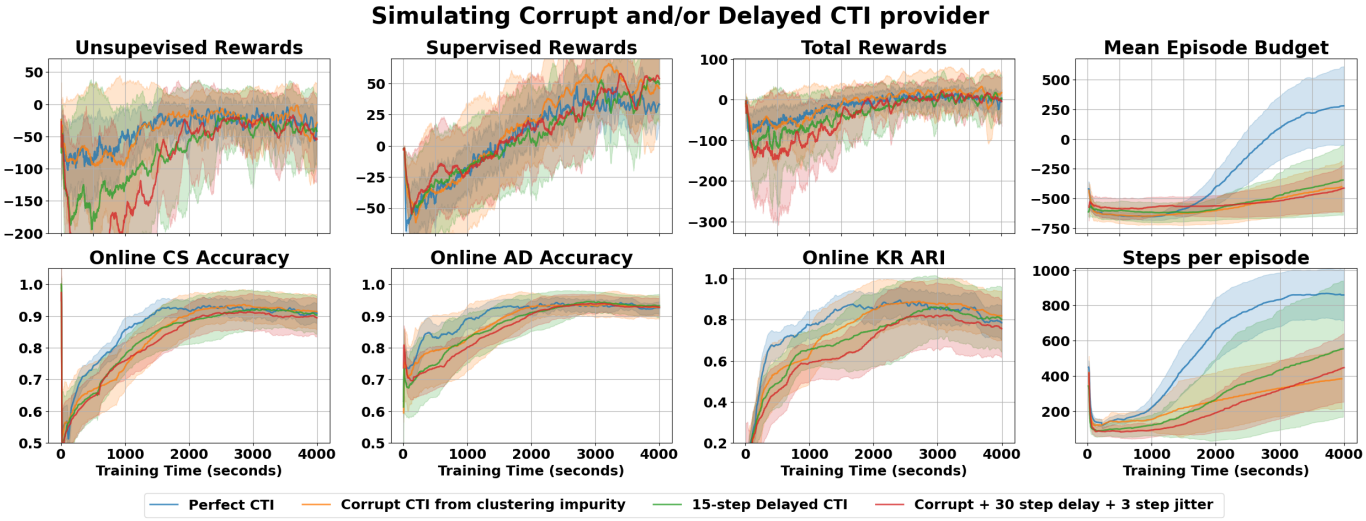


Figure 6. Running averages of the performance metrics during online benchmarking in the TIGER environment under four CTI-delivery-fidelity regimes: a perfect (clean) feed, a corrupt feed whose label-noise and partial-capture severity is coupled to clustering impurity, a 15-step delivery delay, and a compound regime combining corruption with a 30-step delay jittered by ± 3 steps. Lines report running averages of the mean across seeds, and the shaded bands the corresponding 95% confidence intervals.

the quality of the acquired intelligence. Beyond the ideal *Perfect CTI* baseline, we consider three degradations: a *corrupt* feed whose label-noise and partial-capture severity is coupled to the (hidden) impurity of the cluster the CTI is purchased for; a purely *delayed* feed that becomes trainable only 15 decision steps after payment; and a compound worst case that combines corruption with a longer, jittered 30-step delay. Figure 6 reports the resulting learning curves.

The regimes separate almost entirely along the *mean episode budget*: endpoint and rate measures — total episode reward, online CS/AD accuracy, and KR ARI — stay broadly comparable, with corruption and delay mostly slowing the early part of learning before the inference module recovers.

Notice the budget is reset to $\mathcal{B}_0$ each episode and accumulated step by step, $\mathcal{B}_\tau = \mathcal{B}_0 + \sum_{i \leq \tau} R_i$; the total episode reward reflects only the endpoint $\mathcal{B}_N$, whereas the mean episode budget averages the *running budget* $\mathcal{B}_\tau$ over all steps. By summation by parts this time-average weights early earnings more heavily than late ones, so two episodes with the same total but different timing yield different mean budgets. Imperfect CTI acts precisely on timing: a delayed feed leaves the purchased class an unlearnable, still-costly zero-day until it arrives, —because the DM continues to be charged if it attempts to buy it until labels arrive— and a noisy prototype breaks even more slowly. Consequently, the running budget stays depressed for longer even when it recovers to a similar endpoint. In this sense the mean episode budget surfaces the *working-capital cost* of low-fidelity intelligence that endpoint totals hide. We offer this as an illustrative observation, leaving a fuller quantitative study to future work.

V. Conclusions and Future Work

This work presented TIGER, an open-source training environment for researchers to build, train, and test RL-based network intrusion detection and mitigation agents. The main novelty of TIGER is the simulation of operating costs in an open-world scenario, where new types of cyber-attacks may arise after pre-training and deployment of the detection system. Given that the detection system has not been trained on these attacks, a management agent must learn to balance the timely collection of the necessary training data with conservative anomaly-based mitigation strategies. TIGER permits training and evaluating these kinds of management agents in their capacity to maximise the overall detection and mitigation effectiveness within limited resource availability.

TIGER is an extension of the SMARTVILLE network environment, which replays real cyber-attack traces and is built on top of the GNS3 network simulation software. The intrusion detection modules in TIGER inherit the ASAP architecture that combines supervised and unsupervised intrusion detection. By combining these design choices, the RL agents trained in TIGER prove effective in managing the post-deploy continual learning strategies of ML-based network cyber-defence instruments. More specifically, the agents learn to balance the cost-benefit trade-off between using pre-trained intrusion detectors and fine-tuning them over time to detect new attack types.

By considering the cost of adapting statistical learning algorithms to post-deploy attack distributions, TIGER proposes adopting more realistic modelling conditions for research on ML-based cyber-defensive instruments. In this respect, the main future work directions include augmenting the heterogeneity of the attack set, developing agents based on other RL algorithms, scaling the underlying network infrastructure beyond what the SMARTVILLE environment offers, and training agents to manage the federated learning strategy of multiple detection modules. A further important direction is to relax the assumption that network traffic is stationary and non-reactive, extending TIGER to model adaptive adversaries whose traffic reacts to the defender's mitigation decisions — for example, by evading detection, mimicking benign flows, or deliberately targeting the agent's

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

budget.

On realism, generalisation, and the relational bottleneck inductive bias. From a more theoretical point of view, an objection to the generalisability of TIGER is that the Decision Module's state space contains instance-based representations of traffic —namely, the hidden-space prototypes—, breaking the relational bottleneck inductive bias used by the Inference Module (based on the ASAP architecture). For this reason, further work should rigorously assess the extend to which the current setting overlaps with RL-based traffic classification in the latent space. A relational state space in which the DM decides to adopt blocking/accepting/buying actions based only in relational information about the current cluster and the acquired knowledge would certainly go in the direction of generalisation and deserves future research.

While TIGER currently scopes CTI acquisition strictly to the retrieval of labelled traffic traces for ML retraining, real-world CTI also includes structural data such as Tactics, Techniques, and Procedures (TTPs), YARA rules, and STIX/TAXII indicators. Future extensions of the TIGER environment could expand the agent's observation and action spaces to ingest these heterogeneous formats of threat intelligence, requiring the RL agent to decide not only *when* to acquire CTI, but *which type* of intelligence is most cost-effective for the current threat context.

REFERENCES

[1] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[2] Oluwadamilare Harazeem Abdulganiyu, Taha Ait Tchakoucht, and Yakub Kayode Saheed. A systematic literature review for network intrusion detection system (ids). *International journal of information security*, 22(5):1125–1162, 2023.

[3] Martin Mundt, Yongwon Hong, Iuliia Pliushch, and Visvanathan Ramesh. A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning. *Neural Networks*, 160:306–336, 2023.

[4] Nan Sun, Ming Ding, Jiaojiao Jiang, Weikang Xu, Xiaoxing Mo, Yonghang Tai, and Jun Zhang. Cyber threat intelligence mining for proactive cybersecurity defense: a survey and new perspectives. *IEEE Communications Surveys & Tutorials*, 25(3):1748–1774, 2023.

[5] Zhongkun Yu, JunFeng Wang, BinHui Tang, and Li Lu. Tactics and techniques classification in cyber threat intelligence. *The Computer Journal*, 66(8):1870–1881, 2023.

[6] Sai Prasath, Kamalakanta Sethi, Dinesh Mohanty, Padmalochan Bera, and Subhransu Ranjan Samantaray. Analysis of continual learning models for intrusion detection system. *IEEE Access*, 10:121444–121464, 2022.

[7] Saqib Saeed, Sarah A Suayyid, Manal S Al-Ghamdi, Hayfa Al-Muhaisen, and Abdullah M Almuhaideb. A systematic literature review on cyber threat intelligence for organizational cybersecurity resilience. *Sensors*, 23(16):7273, 2023.

[8] Tom SF Haines and Tao Xiang. Active learning using dirichlet processes for rare class discovery and classification. In *BMVC*, pages 1–11, 2011.

[9] Xiwen Huang and Pierre Pinson. Querymarket: Cost-aware online active learning in data markets. *arXiv preprint arXiv:2606.17805*, 2026.

[10] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

[11] Keerthana Sivamayil, Elakkiya Rajasekar, Belqasem Aljafari, Srete Nikolovski, Subramaniyaswamy Vairavasundaram, and Indragandhi Vairavasundaram. A systematic study on reinforcement learning based applications. *Energies*, 16(3):1512, 2023.

[12] Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem Beyah. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 755–766, 2016.

[13] Shengping Zhou, Zi Long, Lianzhi Tan, and Hao Guo. Automatic identification of indicators of compromise using neural-based sequence labelling. In Stephen Politzer-Ahles, Yu-Yin Hsu, Chu-Ren Huang, and Yao Yao, editors, *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong, 1–3 December 2018. Association for Computational Linguistics.

[14] Zi Long, Lianzhi Tan, Shengping Zhou, Chaoyang He, and Xin Liu. Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling. In *2019 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[15] Chen Gao, Xuan Zhang, Mengting Han, and Hui Liu. A review on cyber security named entity recognition. *Frontiers of Information Technology & Electronic Engineering*, 22(9):1153–1168, 2021.

[16] Leslie F Sikos. Cybersecurity knowledge graphs. *Knowledge and Information Systems*, 65(9):3511–3531, 2023.

[17] Kiavash Satvat, Rigel Gjomemo, and VN Venkatakrishnan. Extractor: Extracting attack behavior from threat reports. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 598–615. IEEE, 2021.

[18] Xiaojuan Zhao, Rong Jiang, Yue Han, Aiping Li, and Zhichao Peng. A survey on cybersecurity knowledge graph construction. *Computers & Security*, 136:103524, 2024.

[19] Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145:103999, 2024.

[20] Hanxiang Xu, Shenao Wang, Ningke Li, Kailong Wang, Yanjie Zhao, Kai Chen, Ting Yu, Yang Liu, and Haoyu Wang. Large language models for cyber security: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, September 2025.

[21] Jesus F. Cevallos-Moreno, Alessandra Rizzardi, Sabrina Sicari, and Alberto Coen-Porisini. Asap: Automatic synthesis of attack prototypes, an online-learning, end-to-end approach. *Computer Networks*, page 110828, 2024.

[22] Chaomeng Lu, Xufeng Wang, Aimin Yang, Yikai Liu, and Ziao Dong. A few-shot-based model-agnostic meta-learning for intrusion detection in security of internet of things. *IEEE Internet of Things Journal*, 10(24):21309–21321, 2023.

[23] Jesus F. Cevallos-Moreno, Alessandra Rizzardi, Sabrina Sicari, and Alberto Coen-Porisini. Nero: Neural algorithmic reasoning for zero-day attack detection in the iot: A hybrid approach. *Computers & Security*, 142:103898, 2024.

[24] Zhen Yang, Xiaodong Liu, Tong Li, Di Wu, Jinjiang Wang, Yunwei Zhao, and Han Han. A systematic literature review of methods and datasets for anomaly-based network intrusion detection. *Computers & Security*, 116:102675, 2022.

[25] Dylan Chou and Meng Jiang. A survey on data-driven network intrusion detection. *ACM Computing Surveys (CSUR)*, 54(9):1–36, 2021.

[26] Song Wang, Juan Fernando Balarezo, Sithamparanathan Kandeepan, Akram Al-Hourani, Karina Gomez Chavez, and Benjamin Rubinstein. Machine learning in network anomaly detection: A survey. *IEEE Access*, 9:152379–152396, 2021.

[27] Giampaolo Bovenzi, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. A hierarchical hybrid intrusion detection approach in iot scenarios. In *GLOBECOM 2020-2020 IEEE global communications conference*, pages 1–7. IEEE, 2020.

[28] Jian Yang, Xiang Chen, Shuangwu Chen, Xiaofeng Jiang, and Xiaobin Tan. Conditional variational auto-encoder and extreme value theory aided two-stage learning approach for intelligent fine-grained known/unknown intrusion detection. *IEEE Transactions on Information Forensics and Security*, 16:3538–3553, 2021.

[29] Rasheed Ahmad, Izzat Alsmadi, Wasim Alhamdani, and Lo'ai Tawalbeh. Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10):10733–10811, 2023.

[30] Yang Guo. A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer communications*, 198:175–185, 2023.

[31] Alice Bizzarri, Chung-En Yu, Brian Jalaian, Fabrizio Riguzzi, and Nathaniel D Bastian. A synergistic approach in network intrusion detection by neurosymbolic ai. *arXiv preprint arXiv:2406.00938*, 2024.

[32] Alice Bizzarri, Brian Jalaian, Fabrizio Riguzzi, and Nathaniel D Bastian. A neuro-symbolic artificial intelligence network intrusion

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

detection system. In *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2024.

[33] Chathuranga Sampath Kalutharage, Xiaodong Liu, Christos Chrysoulas, and Oluwaseun Bamgboye. Neurosymbolic learning in the xai framework for enhanced cyberattack detection with expert knowledge integration. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 236–249. Springer, 2024.

[34] Gudmund Grov, Jonas Halvorsen, Magnus Wiik Eckhoff, Bjørn Jervell Hansen, Martin Eian, and Vasileios Mavroeidis. On the use of neurosymbolic ai for defending against cyber attacks. In *International Conference on Neural-Symbolic Learning and Reasoning*, pages 119–140. Springer, 2024.

[35] Tommaso Zoppi, Mohamad Gharib, Muhammad Atif, and Andrea Bondavalli. Meta-learning to improve unsupervised intrusion detection in cyber-physical systems. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(4):1–27, 2021.

[36] Mohiuddin Ahmed and Al-Sakib Khan Pathan. Deep learning for collective anomaly detection. *International Journal of Computational Science and Engineering*, 21(1):137–145, 2020.

[37] Jesús F. Cevallos-Moreno, Alessandra Rizzardi, Sabrina Sicari, and Alberto Coen-Porisini. Hero: From high-dimensional network traffic to zero-day attack detection. *Computer Networks*, 265:111264, 2025.

[38] Jesus F. Cevallos-Moreno, Alessandra Rizzardi, Sabrina Sicari, and Alberto Coen-Porisini. Deep reinforcement learning for intrusion detection in internet of things: Best practices, lessons learnt, and open challenges. *Computer Networks*, 236:110016, 2023.

[39] Richard Elderman, Leon JJ Pater, Albert S Thie, Madalina M Drugan, and Marco A Wiering. 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.

[40] 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.

[41] Alex Andrew, Sam Spillard, Joshua Collyer, and Neil Dhir. Developing optimal causal cyber-defence agents via cyber security simulation. In *Workshop on Machine Learning for Cybersecurity (ML4Cyber), Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, Baltimore, Maryland, USA, July 2022. PMLR.

[42] José Álvaro Fernández Carrasco, Iñigo Amonarriz Pagola, Raúl Orduna Urrutia, and Rodrigo Román. Cybershield: A competitive simulation environment for training ai in cybersecurity. In *2024 11th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 11–18, 2024.

[43] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisỳ. 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.

[44] Huynh Phuong Thanh Nguyen, Zhi Chen, Kento Hasegawa, Kazuhide Fukushima, and Razvan Beuran. Pengym: Pentesting training framework for reinforcement learning agents. In *ICISSP*, pages 498–509, 2024.

[45] Sean Oesch, Amul Chaulagain, Brian Weber, Matthew Dixson, Amir Sadovnik, Benjamin Roberson, Cory Watson, and Phillipe Austria. Towards a high fidelity training environment for autonomous cyber defense agents. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, CSET '24, page 91–99, New York, NY, USA, 2024. Association for Computing Machinery.

[46] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

[47] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. 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.

[48] Amrutha Machireddy, Ranganath Krishnan, Nilesh Ahuja, and Omesh Tickoo. Continual active adaptation to evolving distributional shifts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3444–3450, 2022.

[49] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao Tao, Dongyan Zhao, Jinwen Ma, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International conference on learning representations*, 2019.

[50] Haiyan Zhao, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Does continual learning equally forget all parameters? In *International Conference on Machine Learning*, pages 42280–42303. PMLR, 2023.

[51] Suresh Kumar Amalapuram, Akash Tadwai, Reethu Vinta, Sumohana S Channappayya, and Bheemarjuna Reddy Tamma. Continual learning for anomaly based network intrusion detection. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 497–505. IEEE, 2022.

[52] Abdelaziz Amara Korba, Souad Sebaa, Malik Mabrouki, Yacine Ghamri-Doudane, and Karima Benatchba. A life-long learning intrusion detection system for 6g-enabled iov. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1773–1778. IEEE, 2024.

[53] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

[54] Suresh Kumar Amalapuram, Bheemarjuna Reddy Tamma, and Sumohana S Channappayya. Spider: A semi-supervised continual learning-based network intrusion detection system. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 571–580. IEEE, 2024.

[55] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

[56] Thin Tharaphe Thein, Yoshiaki Shiraishi, and Masakatu Morii. Few-shot learning-based malicious IoT traffic detection with prototypical graph neural networks. *IEICE TRANSACTIONS on Information and Systems*, 106(9):1480–1489, 2023. Publisher: The Institute of Electronics, Information and Communication Engineers.

[57] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[58] Meng Fang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Active class discovery and learning for networked data. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 315–323. SIAM, 2013.

[59] Tian Xie, Jifan Zhang, Haoyue Bai, and Robert Nowak. Deep active learning in the open world. *arXiv preprint arXiv:2411.06353*, 2024.

[60] Davide Cacciarelli and Murat Kulahci. Active learning for data streams: a survey. *Machine Learning*, 113(1):185–239, 2024.

[61] Xiwen Huang and Pierre Pinson. How to purchase labels? a cost-effective approach using active learning markets. *INFORMS Journal on Data Science*, 2026.

[62] Sarah Wassermann, Thibaut Cuvelier, and Pedro Casas. Ral-improving stream-based active learning by reinforcement learning. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) Workshop on Interactive Adaptive Learning (IAL)*, 2019.

[63] Jesus F. Cevallos-Moreno, Alessandra Rizzardi, Sabrina Sicari, and Alberto Coen-Porisini. Smartville: A framework for realistic deep learning-based online network intrusion detection, 2026. Journal of Network and Systems Management, In-press.

[64] Taylor W Webb, Steven M Frankland, Awni Altabaa, Simon Segert, Kamesh Krishnamurthy, Declan Campbell, Jacob Russin, Tyler Giallanza, Randall O'Reilly, John Lafferty, et al. The relational bottleneck as an inductive bias for efficient abstraction. *Trends in Cognitive Sciences*, 2024.

[65] Steinar Laenen and Luca Bertinetto. On episodes, prototypical networks, and few-shot learning. *Advances in Neural Information Processing Systems*, 34:24581–24592, 2021.

[66] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc.

[67] Dino Ienco, Albert Bifet, Indrė Žliobaitė, and Bernhard Pfahringer. Clustering based active learning for evolving data streams. In *International Conference on Discovery Science*, pages 79–93. Springer, 2013.

[68] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[69] Anil Menon, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. Characterization of a Class of Sigmoid Functions with Applications to Neural Networks. *Neural Networks*, 9(5):819–835, July 1996.

[70] Karl J Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

[71] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.

For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

[72] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.
[73] Detlef Rhenius. Incomplete information in markovian decision models. *The Annals of Statistics*, pages 1327–1334, 1974.
[74] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.
[75] Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, page 141, 2015.
[76] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine learning proceedings 1995*, pages 362–370. Elsevier, 1995.
[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
[78] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
[79] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pages 1–30. Lille, 2015.
[80] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.
[81] Raia Hadsell, Sumit Chopra, and Yann LeCun. 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.
[82] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.
[83] GNS3 Community. GNS3: The software that empowers network professionals. https://www.gns3.com/, 2026.

APPENDIX A
ADDITIONAL EXPERIMENTS

Please note that two main parameter settings have been used in this work: the main results described in Section IV were obtained under a high traffic volume with more heterogenenous traffic setting in which traffic captures of a same class exhibited high variance. Instead, the results reported in the following ablations come from single runs, used a reduced drastically the heterogeneity of within-class packet features —-favouring the overall class separability—— . Finally, the rewards and the prices for CTI acquisition are rescaled in the following experiments. See Appendix C-B for more information.

A. Kernel regression architectures

The inference module in TIGER relies on the ASAP framework to learn anomaly detection, classification, and clustering online in an end-to-end differentiable manner. At a high level, ASAP combines metric-based meta-learning to separate known from novel traffic with a relational component that estimates whether two novel observations should belong to the same cluster.

An important design choice in this framework is the architecture used to build the latent representation space where these metric-based decisions are made. In the main configuration of this paper, we use Multi-Layer Perceptrons and Gated Recurrent Units to encode time sequences of network records. However, the framework is not tied to these choices, and alternative encoders—for example, temporal convolutional architectures—could be integrated as well.

To study this design space, we benchmarked different relational operators and loss functions for the anomaly-clustering stage:

- The default operator is a Similarity Network [79] that takes the L1 distance between two latent representations and outputs an association score. We refer to this operator as the Distance Kernel Regressor (DistanceKR).
- A lighter alternative applies 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 [80]. 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 [81].

Figure 7 reports the running metrics over 2 hours of training for the full inference-and-decision pipeline, comparing DistanceKR and DotProdKR under both the UMAP-style and BCE-style objectives. (In every case the IM has been pretrained for 30 minutes).

By inspecting Figure 7, one can see that the default architecture converges faster and more consistently than the alternatives. This trend is particularly visible in the *Mean Episode Budget* metric (upper right), where the default configuration stabilises earlier, while the other variants remain noisier for longer and approach their steady regime more slowly. Since the episode budget reflects the long-term balance between CTI acquisition costs and defensive returns, this behaviour suggests that the default architecture provides the Decision Module with a more reliable signal early in training.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

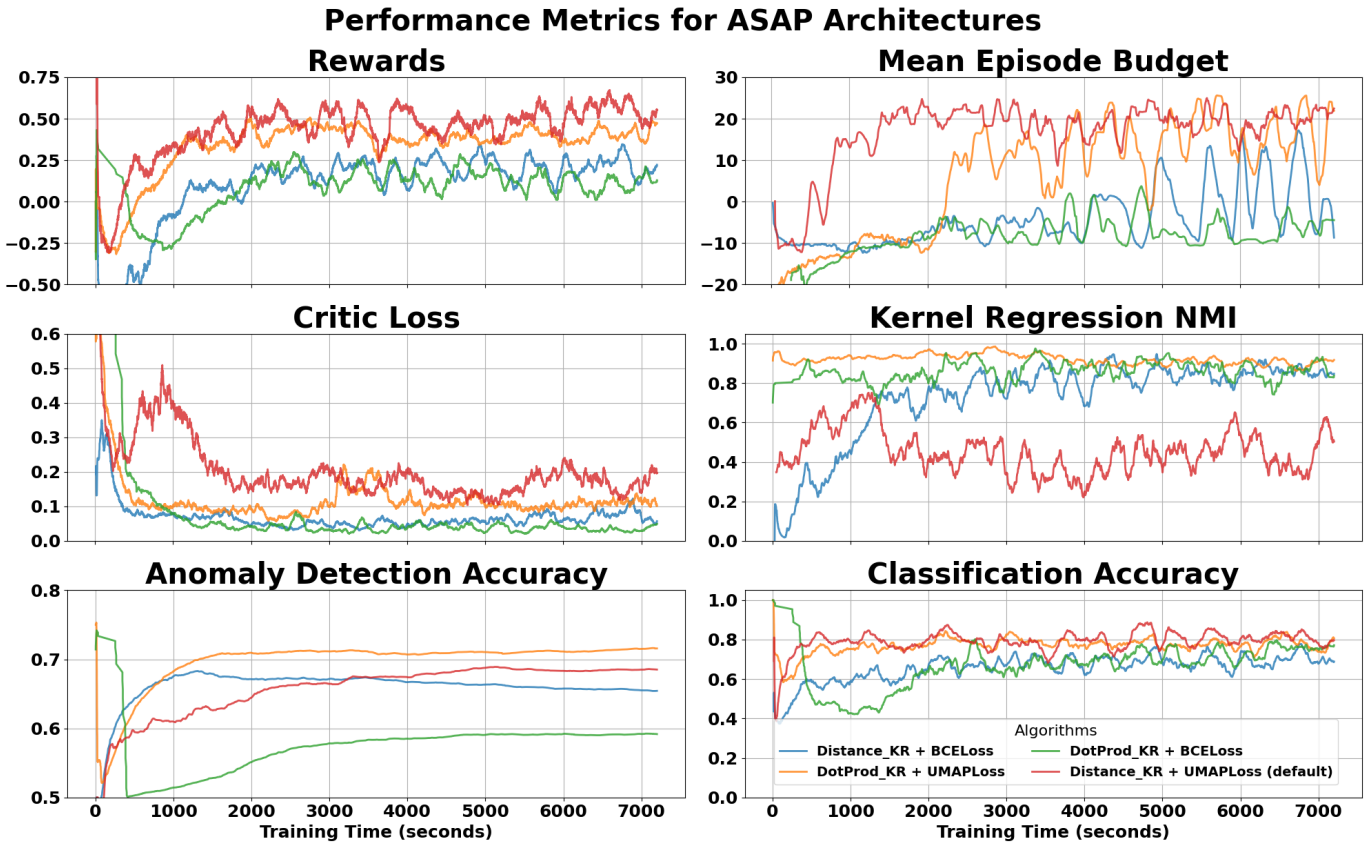


Figure 7. Running averages for performance metrics during 120 minutes of benchmarking in the TIGER environment using different combinations of Kernel Regression Neural Operators and Loss functions.

These findings motivate several directions for future work. A first direction is to investigate hybrid relational operators that combine the efficiency of dot-product interactions with the stronger inductive bias of distance-based similarity networks. A second direction is to test more expressive temporal encoders, such as temporal convolutions or attention-based models, to determine whether better sequence modelling further accelerates budget convergence. More broadly, Figure 7 suggests that architectural choices in the kernel-regression stage affect not only clustering quality, but also the stability of the downstream control policy.

B. Modifying Budget Constraints

Figure 8 shows that the choice of budget bounds changes the learning dynamics of TIGER in a structural way, rather than merely rescaling the rewards. The upper-left plot is particularly informative because, besides the mean episode budget, it also reports the first and third quartiles. These dashed lines reveal how much the budget achieved at the end of an episode varies across runs and over time. In more permissive settings, the interquartile range remains visibly wider during the transient phase, indicating that the agent explores a broader family of strategies before settling. Under tighter constraints, the quartile band shrinks earlier, suggesting a more compressed but also more restrictive learning regime.

This effect is coherent with the relation between budget range and episode horizon. When the admissible interval is wider, the agent can survive suboptimal decisions for longer and therefore execute more steps within each episode before reaching a terminal condition. In practice, this may give the Decision Module more opportunities to interleave pragmatic actions with costly epistemic ones and to observe the delayed benefit of retraining the Inference Module after a CTI purchase. By contrast, narrow budget ranges force earlier terminations, reduce the number of effective decisions per episode, and make the control problem more short-sighted because the price of epistemic actions is paid immediately whereas their return is realised only later.

The figure also suggests that different budget intervals induce different notions of convergence. Settings with shorter episodes may generate a larger number of episodes in the same amount of training time, but each episode conveys less within-episode evidence about whether an epistemic action was ultimately worthwhile. More permissive settings do the opposite: they expose the agent to longer temporal dependencies, which can improve the quality of the learned policy, although often at the cost of a slower and noisier transient phase. For this reason, convergence in Figure 8 should be interpreted jointly through the number of steps per episode and the number of episodes needed to stabilise, not only through the final average budget.

Overall, Figure 8 indicates that budget constraints in TIGER act as a form of curriculum shaping for the Decision Module. By controlling how quickly the agent is punished for poor

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

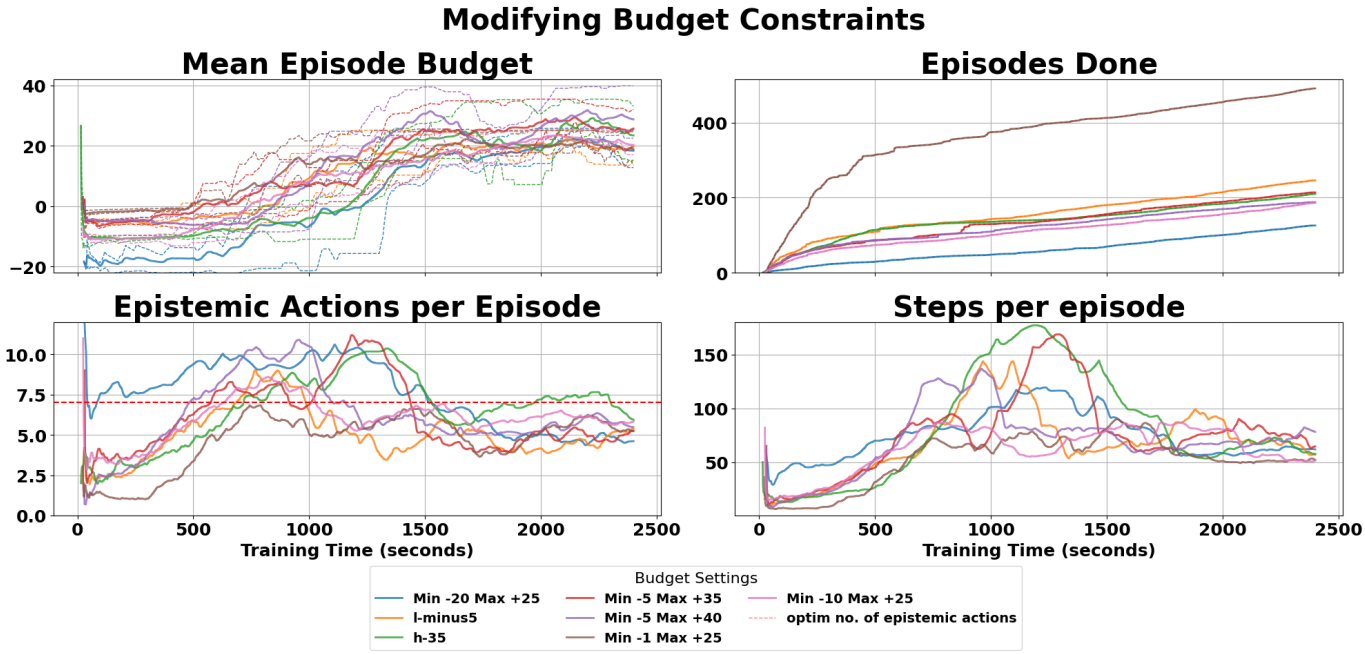


Figure 8. Utilising different budget constraints in TIGER (releasing the constraint of a maximum number of steps per episode). In the upper-left plot, the solid line reports the mean episode budget, while the dashed lines indicate the first and third quartiles, so that the spread of episode budgets can be compared directly across constraint settings.

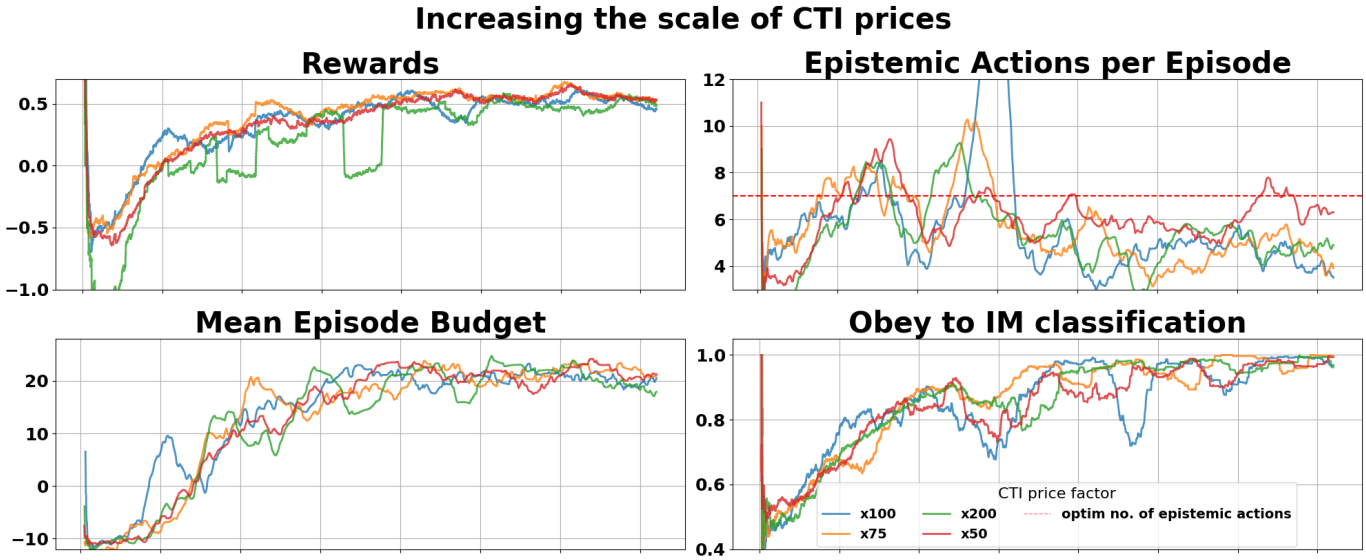


Figure 9. Main performance metrics of DDQN agents under increasing scale factors for CTI acquisition prices. A stochastic multiplicative price-decay mechanism was used in this experiment to show that rescaling the base price preserves the qualitative behaviour of the agents.

actions and how much slack it has to invest in new intelligence, the budget range directly influences sample efficiency, stability, and the visibility of long-term gains from CTI acquisition.

C. Varying the cost of CTI

Figure 9 shows the results of increasing increasing the cost of purchasing CTI. In these experiments, the cost of CTI-acquisition was computed using a multiplicative factor of the correspondent traffic rewards. Moreover, a stochastic multiplicative price-decay mechanism was used in this experiment to show that rescaling the base price preserves the qualitative behaviour of the agents.

The relevant point is that this decay is multiplicative. Consequently, increasing the base CTI price factor also increases the magnitude of subsequent discounted prices, so that the temporal structure of the acquisition problem is preserved to a large extent. In other words, the environment does not simply move from a "cheap CTI" regime to an "expensive CTI" regime in a static way. Rather, it rescales a dynamic pricing process in which the relative advantage of waiting for a better buying opportunity is maintained. This helps explain why the performance curves remain comparatively stable: agents are not facing a completely different problem, but a homothetic variant of the same

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

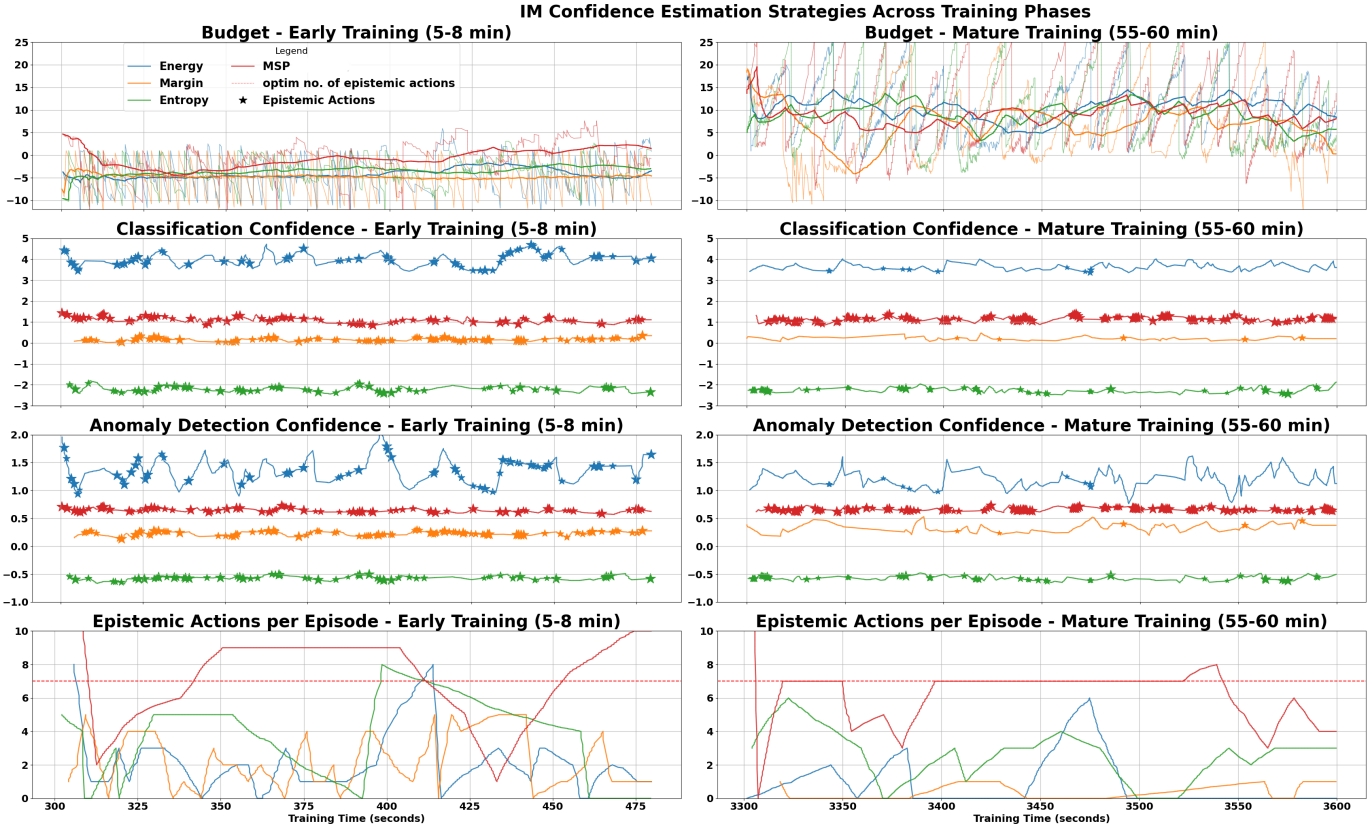


Figure 10. Instantaneous budget over short time intervals during early and mature training under different confidence-estimation strategies. Unlike the previous plots, no running average is applied here, with the exception made for the budget, which shows both a running average and the original series in transparency, so that the trajectories expose the exact per-batch budget fluctuations. Star markers denote epistemic actions; multiple epistemic actions in the same batch increase the marker size.

sequential trade-off.

This observation is important for two reasons. First, a stochastic price-decay process is more realistic than a deterministic or constant price schedule. In real cyber-defence settings, the effort required to obtain actionable intelligence is rarely fixed: the availability of relevant indicators, the maturity of an attack campaign, and the workload of the intelligence provider all introduce fluctuations in acquisition costs over time. In TIGER, these fluctuations can be activated and are generated by a pseudorandom process with a settable seed, so the environment remains fully reproducible while still exposing the controller to a non-trivial and time-varying market for CTI.

Second, the stochasticity of price decay is essential to prevent the benchmark from collapsing into a trivial synchronisation problem. If CTI prices followed a perfectly predictable periodic pattern, then a hand-crafted agent that simply buys every n steps in phase with that schedule could become artificially optimal, independently of the actual uncertainty of the Inference Module or the current attack mix. By contrast, the stochastic decay used in TIGER breaks this possibility: although prices still tend to become more favourable over time, the exact timing of attractive purchase opportunities is not rigidly locked to the episode clock.

APPENDIX B
CONFIDENCE ESTIMATION STRATEGIES

In this appendix we analyse the confidence signals used by the Inference Module (IM) to drive epistemic actions in the Decision Module (DM). Differently from the previous benchmarks, Figure 10 reports *instantaneous* budget values apart from running averages. This visualisation is useful because it reveals how the budget moves within individual episodes, how epistemic actions modify the local trajectory, and how these effects become more visible once training matures and the agent learns to preserve and accumulate budget more consistently.

Both the supervised classification head and the anomaly-detection head can estimate confidence through four alternative strategies. For the classification head, let $\mathbf{z}$ denote the logits over the known classes. The *entropy* strategy computes the average predictive entropy from the softmax distribution, so lower-entropy predictions correspond to higher certainty. The *energy* strategy—the default used throughout the benchmarks reported in this paper—instead uses $\log \sum_i e^{z_i}$, namely a log-sum-exp score over the logits, which is sensitive to the global activation pattern and not only to the winning class. The *margin* strategy measures the gap between the two largest softmax probabilities, thus rewarding clear separation between the best and second-best class. Finally, the *baseline* (MSP-like) strategy compares the average winning logit against

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

the average non-selected logits, producing a confidence score that is conceptually close to a log-ratio between chosen and discarded evidence.

For anomaly detection, the same family of strategies is adapted to the binary known-vs-unknown decision. Entropy is computed over the two-class distribution formed by anomaly and non-anomaly probabilities. Margin reduces to the average distance from the undecided point, i.e., $|2p-1|$. The baseline strategy coincides with the Maximum Softmax Probability (MSP) criterion in the binary case, because it averages the confidence assigned to the predicted side of the decision boundary. Energy is obtained by reconstructing the pre-sigmoid logit from the anomaly probability and then applying a symmetric log-sum-exp score to the pair of opposite logits. In this sense, the energy-based variant is the most sensitive to changes in the full shape of the detector response, while MSP mainly tracks the largest predicted probability.

Figure 10 suggests three main considerations. First, once running averages are removed, the budget dynamics become markedly more irregular at early training time, confirming that the controller is still exploring and that individual epistemic actions can induce sudden local jumps or drops in available resources. At mature training time, all strategies tend to operate at a higher budget regime, which is consistent with a better calibrated exploration–exploitation trade-off and a more profitable use of CTI acquisitions.

Second, the energy-based strategy appears to produce the highest variance in the instantaneous confidence trajectories, especially in the mature stage. We interpret this as evidence that energy—the default strategy—is more informative than the MSP-like baseline for signalling shifts in model uncertainty, because it reacts more strongly to changes in the entire output profile rather than to the top probability alone. In practical terms, this larger variance seems to make the timing of epistemic actions more expressive: star markers are associated with more visible changes in the local confidence path, suggesting that energy yields a sharper coupling between uncertainty spikes and CTI-purchase decisions.

Third, the effect of epistemic actions appears to be stronger on classification confidence than on anomaly-detection confidence. This is coherent with the role of CTI in TIGER: when labelled samples of novel attacks are acquired, the most immediate gain is a better allocation of traffic among known attack classes and newly incorporated categories, whereas the binary separation between known and unknown traffic is already a coarser task. Nevertheless, Figure 10 indicates that epistemic actions influence both components, and this joint effect is again more evident under the energy strategy than under MSP. Overall, these observations support the view that confidence estimation is not a mere auxiliary design choice: it shapes how uncertainty is translated into action, how visibly the budget reacts across episodes, and how efficiently the agent learns to invest in new intelligence.

APPENDIX C
EXPERIMENTAL CONFIGURATION

A. Agent setting

a) ***DQN and DDQN****:* The DQN algorithm is an extension of Q-learning [82] that uses a neural network to approximate (2) and choose the actions that correspond to the maximum estimated Q-value. The ground-truth value for the estimates of the neural approximations is obtained through bootstrapping, using the Bellman equation:

$$Q(\mathbf{s}_\tau, \mathbf{a}_\tau; \theta) \leftarrow R_\tau + \gamma \max_{\mathbf{a}_{\tau+1}} Q(\mathbf{s}_{\tau+1}, \mathbf{a}_{\tau+1}; \theta), \tag{7}$$

where $Q(\mathbf{s}, \mathbf{a}; \theta)$ is the predicted Q-value for state $\mathbf{s}$ and action $\mathbf{a}$, using an *online* neural network whose parameters are denoted by θ.

To improve stability during training, DQN employs two key techniques. First, the *experience replay* technique, by which the agent stores its experiences $(\mathbf{s}_\tau s, \mathbf{a}_\tau, R_\tau, \mathbf{s}_{\tau+1})$ in a replay buffer and randomly samples from this buffer for re-training purposes. By doing so, the correlation between consecutive experiences is broken, and convergence is accelerated. Second, the instantiation of the *target* neural network, whose architecture is identical to the online network. The target network is used to compute the ground-truth Q-values and is updated at a slower pace with respect to the online network.

More specifically, the loss function minimized by the DQN is the mean squared error between the predicted Q-values and the target Q-values. Namely, if the target parameters are denoted by θ^-, the target Q-value is $y = R_\tau + \gamma \max_{\mathbf{a}_{\tau+1}} Q(\mathbf{s}_{\tau+1}, \mathbf{a}_{\tau+1}; \theta^-)$, and the loss function is computed as:

$$L(\theta) = \mathbb{E}_{(\mathbf{s}_\tau s, \mathbf{a}_\tau, R_\tau, \mathbf{s}_{\tau+1}) \sim D} \left[\left(y - Q(\mathbf{s}_\tau, \mathbf{a}_\tau; \theta) \right)^2 \right], \tag{8}$$

On the other hand, the DDQN algorithm is an extension of DQN that aims to mitigate the known issue of overestimation of Q-values by decoupling the action *selection* from the *evaluation* process [78], the key difference between DDQN and DQN lies in the target computation:

$$y = R_\tau + \gamma Q(s', \arg\max_a Q(s', a; \theta); \theta^-) \tag{9}$$

DDQN does not use the target network to select the next action for bootstrapping the ground-truth values. It instead uses the online network θ for action selection and the target network θ^- for action evaluation, thus reducing overestimation bias. This improvement helps the algorithm converge more reliably and efficiently, particularly in environments with noisy or high-dimensional state spaces.

b) ***Exploitation/exploration balance****:* Both the DQN and DDQN implementations in the experiments utilize an ϵ-greedy exploration strategy with an exponential decay factor to balance exploration and exploitation. Specifically, at each time step, the agent selects random actions (exploration) with probability ϵ and the action with the highest Q-value (exploitation) with probability $1-\epsilon$. The exploration rate ϵ is initialized at 1.0 and decays by a factor of $\mathcal{D}$ after each

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

time step, following the rule $\epsilon_{t+1} = \epsilon_t \times \mathcal{D}$. This decay schedule allows for extensive exploration in early steps while gradually transitioning to exploitation as the agent learns optimal policies. ϵ is decreased up to a minimum value $0 < \epsilon_{min} \ll 1$ to maintain a small probability of exploration even in later episodes.

*c) **CTI-affordability guard**:* In the main experiments the epistemic action is subject to an affordability guard, which is the setting denoted as "guarded" in Table IV. Whenever the agent selects a CTI purchase (action 2) whose worst-case price would drive the budget below the floor $\mathcal{B}_{\min}$, the action is remapped to a block before it is scored and stored, so that the agent can never bankrupt itself on a single epistemic action. This amounts to invalid-action masking on the acquisition channel: it removes self-inflicted bankruptcies from CTI buys and thereby lowers the variance of the learned policy, at the cost of no longer measuring the agent's ability to discover the budget constraint on its own.

*d) **Rainbow DQN settings**:* The Rainbow-style variants extend the baseline DQN and DDQN agents with a compact set of stabilisation and sample-efficiency enhancements that are all consistent with the implementation defaults reported in Table IV. First, Prioritised Experience Replay (PER) replaces uniform replay with a sampling strategy that revisits transitions having larger temporal-difference errors more often, so that the DM focuses updates on the most informative budget-management and CTI-acquisition decisions. In our implementation, PER is enabled with standard coefficients α=0.6 and β=0.4, while the Huber loss is weighted by the corresponding importance-sampling factors.

Second, the agent stores transitions using 3-step returns (n=3) rather than purely one-step targets. Concretely, the reward attached to each replayed sample aggregates up to three consecutive discounted rewards before bootstrapping, which helps propagate the delayed effect of epistemic actions—such as purchasing CTI now to improve future mitigation decisions—more quickly through the value function. This is especially useful in TIGER, where the benefit of acquiring intelligence is often not immediate but materialises after the IM has adapted and later traffic is handled more effectively.

Third, the Rainbow configurations use soft target-network updates with τ=0.005. Instead of copying the online network to the target network only through abrupt hard synchronisations, each update interpolates the target parameters toward the online ones, which smooths the learning dynamics and reduces oscillations in the bootstrapped targets. This mechanism is combined with the DDQN target computation when the underlying agent is DDQN-based, thereby preserving the usual decoupling between action selection and action evaluation.

Finally, the dueling enhancements used in the Dueling DQN and Dueling DDQN variants replace the single Q-head with separate estimators for the state value and the per-action advantage, which are recombined to obtain the final action values. This decomposition is useful in our setting because several actions can have similar consequences in many states, while the overall desirability of the state—e.g., whether the current budget, uncertainty, and attack mix make any action risky—still needs to be estimated accurately. Relative to the plain DQN/DDQN baselines, the dueling variants therefore provide a more structured value representation, whereas PER, 3-step returns, and soft target updates mainly improve how efficiently and stably that representation is learned.

B. Parameters and Hyper-parameters

Full configuration files, result-reproduction scripts, and plotting utilities—which retrieve logged metrics directly from our public *Weights&Biases* dashboards[7] — are available in our public GitHub repository at github.com/DISTA-IoT/tiger. Tables IV and V summarise the parameters most relevant to the experiments in Sections IV and the preceding appendices.

The IM is a two-stream recurrent network (one stream for flow statistics, one for sampled packet payloads) with a prototypical classifier and a kernel-regression head, as described in Section III. The DM is a DDQN agent with prioritised experience replay (PER) and soft target-network updates; n-step returns are used to reduce variance in the bootstrapped Q-value targets.

The traffic-class curriculum is reported in Table V. *Known* classes are those available to the IM at pre-training. *G1* classes are *fake* ZdAs: they are present in the network from the start but are deliberately withheld from the IM's closed-set labels and used instead as unlabelled anomalies to supervise the kernel-regression clustering head during online training. *G2* classes are *real* ZdAs: they are entirely absent from the IM's initial knowledge and can only be incorporated through an epistemic action, i.e., by the DM purchasing the corresponding labelled traces from the CTI authority.

C. Sensitivity analysis

To complement the qualitative discussion provided in this appendix, Table VI reports summary statistics of the main episodic metrics obtained under the different Rainbow DQN ablations considered in this work. For compactness, we denote `mean_episode_reward`, `sum_episode_rewards`, and `mean_episode_budget` as MER, SER, and MEB, respectively. The comparison helps clarify how PER and soft target updates affect not only average returns, but also the stability of the learned policy.

Table VI suggests that the best overall trade-off is obtained when Prioritized Experience Replay (PER) and soft target updates are enabled jointly. This configuration yields the highest mean episodic reward (0.429), the highest median reward (0.436), and the largest maximum reward (0.910), while also keeping the standard deviation comparatively low. These statistics indicate not only better peak performance, but also a more stable learning process across episodes.

By contrast, enabling only one of the two enhancements appears insufficient to match the robustness of the combined setting. In particular, the *Soft Update* variant shows the weakest aggregate performance, with the lowest mean episodic

[7]Our public dashboards containing a complete interactive result set of the experiments done in this project are available at https://wandb.ai/jfcevallos/TIGER3.0 (main results) and https://wandb.ai/jfcevallos/TIGER2.0 (appendixes ablations).

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

Table IV
HYPER-PARAMETERS USED IN THE TIGER EXPERIMENTS.

Parameter	Value
Inference Module (IM)	
Hidden size	200
GRU layers	2
Dropout	0.2
Kernel-regressor heads	2
Flow feature dimension	4
Packet feature dimension	64
Flow time-window (T)	10
Packets per sample	10
Support shots (k)	4
Batch size	16
Learning rate	10^{-3}
Replay buffer capacity	10 000
Attractive / repulsive weights	1 / 1
Decision Module (DM) – DQN variants	
Replay batch size	32
Agent memory size	50 000
Discount factor γ	0.99
n-step returns	3
Target-network update frequency	50 steps
Soft-update τ	0.005
PER α / β	0.6 / 0.4
ϵ init / min / decay	1.0 / 0.01 / 0.999

Game / environment	*Main experiments*	*Ablations (reduced traffic)*
Initial budget $\mathcal{B}_0$	1	1
Budget ceiling $\mathcal{B}_{\max}$	-	25
Budget floor $\mathcal{B}_{\min}$	-1000	-10
Max episode steps	1000	50
CTI price factor	*class-specific* 10-150	50
CTI price decay	no	yes
Useless-epistemic penalty	0 (guarded)	7
Cluster impurity penalty	100	8
Flow-stats polling interval (s)	5	5
Random seed	1,2,3,4,5	777

Table V
PER-CLASS REWARDS AND CURRICULUM ROLES (DISTA_TIGER PROFILE).

Traffic class	Reward C_j		Curriculum role
	Main exp.	Ablations (reduced traffic)	
hue	+1	+0.05	Known (benign)
echo	+1	+0.02	G2 (benign)
doorlock	+1	+0.05	G2 (benign)
hakai	−1	−0.10	Known
torii	−1	−0.21	Known
okiru	−1	−0.48	G1
cc_heartbeat	−1	−0.31	G1
generic_ddos	−1	−0.29	G1
mirai	−1	−0.20	G2
gafgyt	−1	−0.25	G2
hajime	−1	−0.13	G2
h_scan	−1	−0.17	G2
muhstik	−1	−0.36	G2

Table VI
SUMMARY STATISTICS OF THE MAIN EPISODIC METRICS FOR THE RAINBOW DQN CONFIGURATIONS USED IN THIS PAPER.

Algorithm	Metric	Mean	Std Dev	Min	25th Pctl	Median	75th Pctl	Max
PER	MER	0.17	0.61	-3.30	0.19	0.30	0.38	0.55
PER	SER	17.46	10.94	-13.20	10.02	24.33	24.89	26.17
PER	MEB	18.30	10.53	-10.51	10.82	24.95	25.53	26.94
Soft Upd.	MER	0.13	0.45	-3.17	0.05	0.20	0.35	0.59
Soft Upd.	SER	13.67	12.28	-13.63	3.01	18.75	24.57	26.20
Soft Upd.	MEB	14.55	11.98	-12.12	3.98	19.53	25.29	26.80
Base.	MER	0.37	0.22	-1.47	0.31	0.38	0.47	0.69
Base.	SER	19.38	8.57	-11.78	15.40	24.39	24.82	26.03
Base.	MEB	20.03	8.40	-9.18	16.11	24.96	25.46	26.69
PER + SU	MER	0.43	0.18	-0.42	0.37	0.44	0.54	0.91
PER + SU	SER	19.77	8.16	-3.38	17.32	24.20	24.87	25.99
PER + SU	MEB	20.37	8.03	-1.73	17.92	24.79	25.42	26.45

reward and the lowest mean episodic budget among the tested configurations. The *PER*-only configuration performs slightly better than *Soft Update* alone, but still remains clearly below the *Baseline* in both average reward and variability. Overall, the ablation confirms that the two mechanisms are complementary in this task: PER improves the efficiency of learning from informative transitions, while soft target updates help smooth the optimisation dynamics, and their combination translates into the most reliable policy among the tested alternatives.

APPENDIX D
RUNTIME METRICS

The TIGER testbed runs on GNS3 v2.2.55 [83]. The GNS3 server operates on Ubuntu 22.04.3 LTS inside a KVM virtual machine equipped with 50 vCPUs (x86_64 architecture). This configuration provides a single socket with one thread per core, resulting in 50 logical processors in total. The current CPU frequency reported by the guest is approximately 2.19 GHz. Due to the virtualised nature of the environment, this value reflects the effective operating frequency observed inside the KVM guest rather than a fixed base frequency of the underlying physical host CPU. The CPU utilisation percentages presented in Fig. 11 were measured as relative usage on one allocated vCPU.

The upper-left panel reports the latency of one training step of the Inference Module (IM), including a single-batch forward and backpropagation pass; the upper-right panel reports IM inference-only latency, i.e., forward passes without backpropagation. The lower-left panel reports the processing time including acting, one forward/backward update, and one batch of experience replay for the Decision Module (DM). The lower-right panel reports CPU utilisation at the controller node hosting both the IM and DM. For consistency, note that even configurations labelled as "perfect IM" or using oracle information still keep the neural modules active in the background, since their confidence estimates are required to assemble the DM input space. The plots are intended to show stable runtime behaviour across configurations rather than to support a detailed scalability analysis, which is left as future work.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2026.3732249

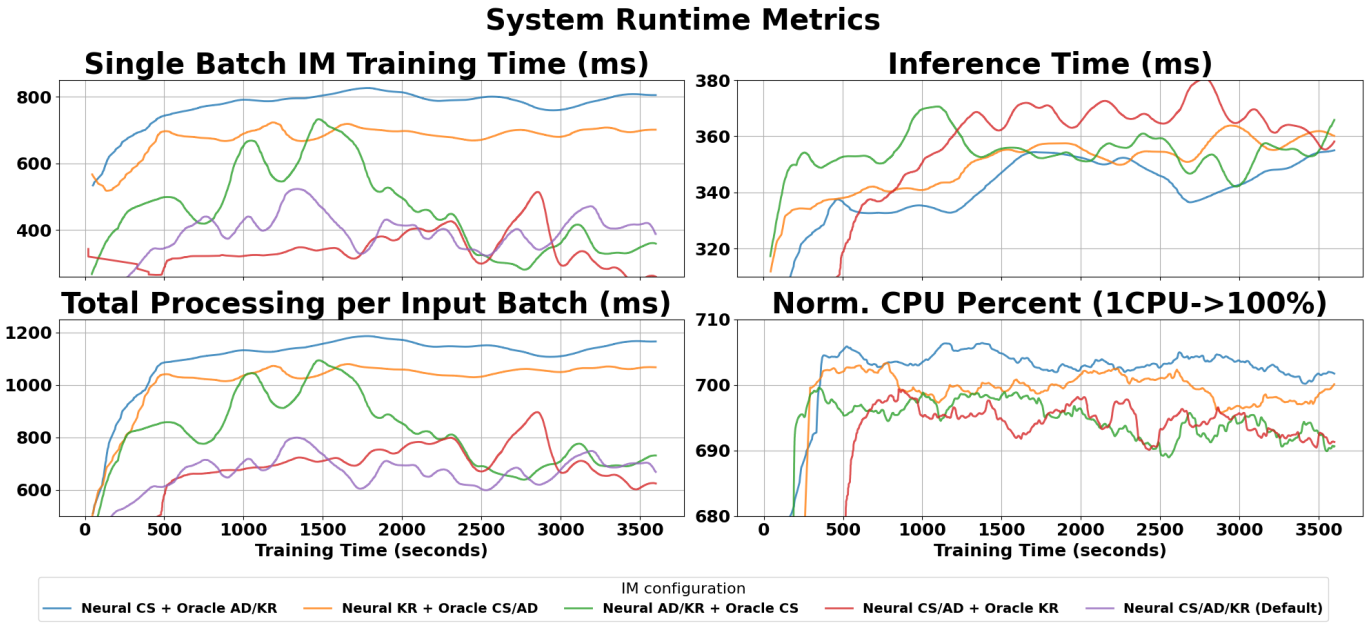


Figure 11. Runtime metrics collected during training under different degrees of reliance on learned modules versus oracle signals.



Jesús F. Cevallos M. received a Ph.D. in Computer Science Engineering from Sapienza University (Rome) in 2022. He now covers 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.



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



Alessandra Rizzardi received her BS and MS degrees in Computer Science 110/110 cum laude at the University of Insubria in Varese (Italy), in October 2011 and July 2013, respectively. In December 2016, she got her Ph.D. in Computer Science and Computational Mathematics at DiSTA (Department of Theoretical and Applied Science). Her research activity is focused on cybersecurity. She is now an Associate Professor of Software Engineering at the University of Insubria. She has been the author of more than 40 scientific papers, which have been published in international journals and conference proceedings. She is an 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.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/